Цифрлық әдіскер
ақпараттық-білім беру ортасы

ДӘРІСТЕР ЖИНАҒЫ

Python бағдарламалау тілі

Болашақ информатика мұғалімдеріне арналған Python негіздері: синтаксис, деректер құрылымдары, функциялар және алгоритмдік есептер.

- Синтаксис және деректер түрлері
- Шартты операторлар мен циклдер
- Тізімдер, сөздіктер, жиындар
- Функциялар және модульдер
- Файлдармен жұмыс
- Алгоритмдік есептер

ДӘРІС ЖИНАҒЫ

КІРІСПЕ

Оқу құралында заманауи технологиялар мен бағдарламалау тәсілдерінің теориялық негіздерімен қатар, практикалық бағдарлама құру мәселелері қамтылған. Python жоғары деңгейлі программалау тілі аясында негізгі алгоритмдік құрылымдар мен олардың жүзеге асырылуы кеңінен қарастырылған. Мұнда компьютерлік бағдарламаларды жасау ерекшеліктері, Python тілінің негізгі синтаксистік құралдары - жолдар, тізімдер, сөздіктер, файлдар - және оларды тиімді пайдалану тәсілдері көрсетілген. Сонымен қатар, бағдарламалау барысында жиі кездесетін есептер мен оларды шешудің жолдары, функциялар арқылы бағдарламаны құрылымдау мен жетілдіру мәселелері жан-жақты сипатталған. Python тілінде графикалық интерфейсі бар қосымшаларды дайындау әдістері бірізді әрі жүйелі түрде баяндалады. Теориялық бөлімдер түрлі мысалдармен және бағдарламалау процесін кезең-кезеңмен көрсететін иллюстрациялармен толықтырылып, оқушылардың өздігінен орындауына арналған практикалық тапсырмалар арқылы олардың нақты дағдыларын қалыптастыруға жол ашады. Пәнді оқытудың негізгі мақсаты оқушылардың логикасын дамытып құрылымдық бағдарламалаудың базалық түсінігін қалыптастыру, 2d ойынына программа құруға үйрету, бағдарламалу тілінде PyGame, Tkinter кітапханасын іске қосуды үйрету, ойын терезесін ашу үшін PyGame, Tkinter кітапханасының дайын модульдерін пайдалануға, GUI арқылы программаларды басқару кодын жазуға дағдыландыру, бағдарламалық кодты трансляциялау әдістерін үйрету. Оқу құралында бағдарламалау тілдерінің PyGame, Tkinter кітапханасы жайлы түсініктер мен Python бағдарламалау тілінде PyGame, GUI, Tkinter кітапханасын орнату әдісі және қолдану жолдары, GUI арқылы программаларды басқару, батырма арқылы деректерді енгізу, т.б. мәліметтер берілген. Бағдарламаны меңгеру және бағдарлама құруды үйрену нәтижесінде білім алушы, білуі керек:

- Python бағдарламалау тілінің ерекшеліктерін;
- Python ide бағдарламалау ортасында жұмыс жасау принциптерін;
- Python бағдарламалау тілінің синтаксисінің негіздерін;
- Python тілінің негізгі объектілернің құрылымын және типтілігін;
- Python тілінің құрылымын басқару және олардың жұмыс принципін;
- Python программалау тілінде 2d ойынын құру, PyGame, Tkinter кітапханасының дайын модульдерін пайдалану;
- GUI арқылы программаларды басқару.

жасай алуы керек:

- Python ide бағдарламалау ортасын орнатып баптай алуы;
- Python тілінің басқарушы конструкцияларын жаза алуы;
- Тұтынушылық функцияларын құрып және қолдана алуы;
- Python модульдерін жүктеп және осы модульдің функцияны шақыру, модульдің анықтамалық ақпараттарымен жұмыс жасай алуы;
- Python тілінде 2d ойынын құра алады.

Оқу құралында Python-да ойын құруға және кодтау дағдыларын жетілдіруге қызығушылық танытатын жаңадан бастаушылар мен бағдарламашыларға, білім алушыларға арналған. Оқу құралы болашақ ақпараттық технологиялар мамандарының әдістемелік және кәсіби құзыреттілігін қалыптастыруға, сондай-ақ ғылыми-зерттеу дағдыларын дамытуға үлес қосады деп сенеміз.

1 Python бағдарламалау жүйесі. Python тілінің негізгі элементтері

1.1 Python бағдарламалау тілі

Python тілі - тиімділігі жоғары бағдарламалау тілі. Басқа тілдермен салыстырғанда Python-да код қысқа әрі ықшам жазылады, бұл өз кезегінде кодтың құрылымын жеңіл әрі түсінікті етеді. Python синтаксисі анық әрі таза код жазуға жол ашады, ал бұл кодпен жұмыс істеуді - яғни оны түзету мен жаңартуды - жеңілдетеді. Python тілі көптеген салаларда

қолданыс табады: күрделі ойындар әзірлеуден бастап, веб-қосымшалар мен бизнеске бағытталған шешімдерге дейін. Сонымен қатар, ол түрлі жобаларда ішкі құралдар жасауға және ғылыми зерттеулерде, математикалық есептерді шешуде белсенді қолданылады. Бүгінгі таңда Python-ның екі негізгі нұсқасы бар: Python 2 және Python 3. Бағдарламалау тілдері уақыт өте келе жаңа идеялар мен технологиялардың ықпалымен дамып отырады. Python тілі де бұл үрдістен шет қалмай, оны жетілдіруге бағытталған үздіксіз жұмыстар жүргізілуде. Кейбір жаңартулар үлкен болмағанымен, Python 2-де жазылған кодтар Python 3 жүйесінде жұмыс істемеуі мүмкін жағдайлар кездеседі (1.1 кесте).

1.1 КЕСТЕ - PYTHON ТІЛІНІҢ НҰСҚАЛАРЫ

Python нұсқасы	Шығарылған күні
Python 1.0	1994 жылдың қаңтары
Python 1.5	1997 жыл 31 желтоқсан
Python 1.6	2000 жыл 5 қыркүйек
Python 2.0	2000 жыл 16 қазан
Python 2.1	2001 жыл 17 сәуір
Python 2.2	2001 жыл 21 желтоқсан
Python 2.3	2003 жыл 29 шілде
Python 2.4	2004 жыл 30 қараша
Python 2.5	2006 жыл 19 қыркүйек
Python 2.6	2008 жыл 1 қазаны
Python 2.7	2010 жыл 3 шілде
Python 3.0	2008 жыл 3 желтоқсан
Python 3.1	2009 жыл 27 маусым
Python 3.2	2011 жыл 20 ақпан
Python 3.3	2012 жыл 29 қыркүйек
Python 3.4	2014 жыл 16 наурызы
Python 3.5	2015 жыл 13 қыркүйек
Python 3.6	2016 жыл 23 желтоқсан
Python 3.7	2018 жыл 27 маусым
Python 3.8	2019 жыл 14 қазан
Python 3.9	2020 жыл 5 қазан
Python 3.10	2021 жыл 4 қазан
Python 3.11	2022 жыл 3 қазан
Python 3.12	2023 жыл 3 қазан
Python 3.13	2025 жыл 12 наурыз

Python - кросс-платформалық тіл, яғни ол барлық танымал операциялық жүйелерде қолданыла алады. Дегенмен, әр операциялық жүйеге орнату жолдары сәл ерекшеленеді. Python тілі келесі мүмкіндіктерімен ерекшеленеді:

- серверлік веб-қосымшаларды жасауға мүмкіндік береді;
- бағдарламалық жүйелермен бірігіп жұмыс процесін автоматтандыруға жарамды;

- дерекқорлармен байланыс орнатып, файлдарды оқу және өңдеу қабілетіне ие;
- күрделі есептеулер мен үлкен деректерді өңдеуге бейімделген;
- бағдарламалық өнімдерді жасау үшін кеңінен қолданылады. Python 3 нұсқасында мықты және тиімді бағдарлама жазу үшін түрлі модульдер мен кітапханаларды дұрыс үйлестіру маңызды. Бұл пайдаланушыға күрделі процестерді жеңілдетіп, дайын шешімдер арқылы бағдарламаны тез құрастыруға мүмкіндік береді. Егер Python тілін жеке компьютерде қолданғыңыз келсе, оның ең соңғы нұсқасын https://www.python.org/downloads/ ресми сайтынан жүктеуге болады. Орнату үшін келесі қадамдарды орындау қажет:
- Жоғарыда көрсетілген сайтқа кіріп, Python орнату файлын жүктеу;
- Орнату шебері нұсқауларын орындау;
- Python жұмысын тексеру үшін қарапайым бір-екі бағдарлама жазып көру.

Python-ды орнату (жүктеу). Python - тегін таралатын бағдарлама, оны 1.1-суретте көрсетілген ресми веб-сайттан компьютерге оңай жүктеуге болады. Кодты іске қосудың екі түрлі тәсілі бар:

- Интерактивті режим;

1.1 СУРЕТ - PYTHON БАҒДАРЛАМАСЫН ЖҮКТЕУ

Интерактивті режим деп аталуының себебі - бұл режимде «Пуск» батырмасы арқылы IDLE деп іздеу нәтижесінде IDLE (Python 3.7 32-bit) бағдарламасы ашылады. Онда жазылған командалар тікелей жұмыс аймағында орындалады. Бұл әдіс жеке жолдармен кодты орындап көруге және сынауға өте қолайлы (1.2-суретке қараңыз).

1.2 СУРЕТ - PYTHON-НЫҢ ИНТЕРАКТИВТІ НҰСҚА КӨРІНІСІ

Сондай-ақ, а айнымалысына белгілі бір мән беріп, сол мән арқылы түрлі операцияларды орындауға мүмкіндік бар (1.3 сурет).

1.3 СУРЕТ - АЙНЫМАЛЫ МӘНІН ЕСЕПТЕУ

Алайда, интерактивті режимнің бір кемшілігі - алдыңғы енгізілген нұсқаулар сақталмайды. Мысалы, Shell мәзірінен Restart Shell (Қайта іске қосу) пәрмені таңдалған жағдайда, бұрын берілген а айнымалысы жойылып, жүйе мұндай мәннің жоқ екенін көрсетеді (1.4 және 1.5-суреттерді қараңыз).

1.4 СУРЕТ - ҚАЙТА ЖҮКТЕУ (RESTART SHELL) КӨРІНІСІ

1.5 СУРЕТ - ҚАЙТА ЖҮКТЕУ НӘТИЖЕСІНІҢ КӨРІНІСІ

Келесі маңызды ерекшелік - санды енгізген кезде оны экранға шығару үшін print сөзін жазу міндетті емес, сан автоматты түрде көрсетіледі. Ал мәтін шығару үшін print командасынан кейін жақша ішіне тырнақша арқылы мәтінді жазу қажет (1.6-сурет).

1.6 СУРЕТ - PRINT СӨЗІН ҚОЛДАНУ КӨРІНІСІ

1. Файлдық режим. Файлдық нұсқаны пайдалану үшін File → New File (Ctrl+N) командасын таңдау қажет. Осы әрекеттен кейін 1.7-суретте көрсетілген жаңа терезе ашылады.

1.7 СУРЕТ - ЖАҢА ФАЙЛ ҚҰРУ

Ашылған жаңа терезеге есептелетін мән print командасы арқылы енгізіледі. Бұл файлға атау беріліп, ол сақталады. Кейіннен Run → Run Module (F5) пәрмені таңдалады. Нәтижесінде нәтиже интерактивті режим терезесінде көрсетіледі (1.8-сурет).

1.8 СУРЕТ - ФАЙЛДЫҚ ЖӘНЕ ИНТЕРАКТИВТІ НҰСҚАЛАР КӨРІНІСІ

Назар аударатын жайт - егер print командасының алдында бос орын (пробел) жазылса, бұл синтаксистік қате ретінде қабылданады. Мәтінді экранда көрсету үшін міндетті түрде print пәрмені қолданылады (1.9-сурет). Сонымен қатар, мәтін жақша ішіндегі тырнақшада жазылуы қажет. Егер тырнақша қойылмаса, бағдарлама қате туралы хабарлама шығарады.

1.9 СУРЕТ - PRINT СӨЗІНІҢ ЖАЗЫЛУ КӨРІНІСІ

Енді объектілер, сандар және оларға қолданылатын операцияларға тоқталайық. Бағдарлама - бұл компьютерге белгілі бір әрекеттер тізбегін орындауды анықтайтын нұсқаулар жиынтығы. Объектілерді түсіну - бағдарламада қолданылатын сандар мен мәтінді анықтау үшін маңызды. 1-мысалда (1.9-сурет) бес қатарда объектілер көрсетілген. Алғашқы үш қатар сандарды есептеуге арналған болса, соңғы екі қатар мәтінді экранға шығару үшін қолданылады. Сондықтан объектілердің алдын ала анықталған түрлеріне назар аудару керек.

1.2 Сандар (Numbers)

Python программалау тілінде сандар - деректердің маңызды типтерінің бірі. Олар әртүрлі математикалық есептеулер мен логикалық операцияларда қолданылады. Python тіліндегі сандар бірнеше типте болады: қарапайым бүтін сандар (int типі), ұзын бүтін сандар (long типі), және нақты сандар (float типі). Мектептегі есептерде көбінесе бүтін және нақты сандар қолданылады. Сандарға қатысты негізгі операциялар 1.2-кестеде көрсетілген.

1.2 КЕСТЕ - САНДАРМЕН ОРЫНДАЛАТЫН НЕГІЗГІ ОПЕРАЦИЯЛАРДЫҢ ЖАЗЫЛУЫ

Операциялар	Сипаттамасы
x + y	Қосу
x – y	Азайту
x * y	Көбейту

x/y Бөлу. Ескерту: Python 3 нұсұасынан бастап / операторы әрдайым нақты сан (float) түрінде нәтиже береді. Мысалы, 100 / 8 нәтижесі 12.5 ал 100 / 4 операциясының нәтижесі 25.0 болады. x**y Дәрежелеу x//y Қалдықсыз бөлу. Бөлінді мәнінің бүтін санын алады. x%y Қалдықты бөлу. Бөлінген мәннің қалдық санын алады

Сандарды нақты типтен бүтін типке және керісінше түрлендіру үшін Python тілінде int() және float() функциялары бар. Мысалы, int(12.6) нәтижесі 12 болады, ал float(12) - 12.0. Бүтін және нақты сандарға қолданылатын операцияларда 1.3-кестеде келтірілген ережелерді ескеру қажет. Мысалы:

```
2 + 5 = 7
2.5 + 5 = 7.5
5 – 6 = –1
4.5 – 3 = 1.5
2 * 3 = 6
2 – 3.0 = –1.0
```

1.3 КЕСТЕ - БҮТІН ЖӘНЕ НАҚТЫ САНДАРҒА АРНАЛҒАН НЕГІЗГІ ОПЕРАЦИЯЛАР

Операциялардың басымдылығы	Мысалдар
Жақша іші	2*(5+2)=14
**** дәрежелеу**	3*2**3=24
***///%**	6/3*5=10
+-	4-5+2=1

Бүтін сандармен операция жүргізуге арналған Python-ның кірістірілген математикалық функциялары (1.4-кестеде көрсетілген).

1.4 КЕСТЕ - КІРІСТІРІЛГЕН МАТЕМАТИКАЛЫҚ ФУНКЦИЯЛАР ЖАЗЫЛУЫ

Функция	Функция нәтижесі
abs (value)	abs (-7)→7, abs (7)→7
min (x1,x2, ...xn)	min (-43, 90, 832, -78)→-78
max (x1,x2, ...xn)	max (-43,90,832,-78)→832
pow (x,y)	pow (2,5) →32
round ()	round (3.5)→4

```
math модулі (қосымша функциялар)
Python-да күрделі есептер үшін math модулі қолданылады. Оны пайдалану үшін бағдарламаның басында import math
командасын жазу қажет.
```

Функция Сипаттамасы Мысал

```
math.sqrt(x)
```

Квадрат түбір

```
math.sqrt(25) → 5.0
math.ceil(x)
```

Жоғары қарай дөңгелектеу

```
math.ceil(3.2) → 4
math.floor(x)
```

Төмен қарай дөңгелектеу

```
math.floor(3.8) → 3
```

math.pi Пи саны math.pi → 3.1415...

```
math.sin(x)
Синус (радиан)
math.sin(math.pi/2) → 1
math.factorial(n)
```

Факториал

```
math.factorial(5) → 120
```

Мысал 1.1: Сандар түрін тексеру

```
x = 3.5
print(type(x))  # float
```

Мысал 1.2: Пайдаланушыдан сандар сұрап, амалдар орындау

```
a = float(input("Бірінші сан: "))
b = float(input("Екінші сан: "))
print("Қосу:", a + b)
print("Бөлінді:", a / b)
```

1.3 Логикалық мәндер (Boolean values)

Python тіліндегі логикалық мәндер екі түрден тұрады: True (Ақиқат) және False (Жалған). Бұл мәндер - bool типіне жатады.

```
a = True
b = False
print(type(a))  # <class 'bool'>
```

Python-да бас әріппен жазылады: True, False

Логикалық шамалар логикалық операциялар мен өрнектердің нәтижесінде анықталады (1.5 - кестеге қараңыз).

1.5 КЕСТЕ - ЛОГИКАЛЫҚ АМАЛДАРДЫҢ СИПАТТАЛУЫ

Өрнек	Сиаттамасы
>	"Үлкен" шарт (мысалы, a > b)
<	"Кіші" шарт (мысалы, a < b)
==	Теңдік шарты (А жәнеВ тең екені тексеріледі)
!=	Теңсіздіктің шарты (А және В тең емес екені тексеріледі)
not x	Терістеу (x шарты орындалмайды)
x and y	

«және» логикалық өрнегі. x және y шартын орындау үшін x және y шарттарын бір уақытта орындау қажет. x or y «немесе» логикалық өрнегі. x немесе y шартын орындау үшін шарттардың бірін орындау қажет. x in a X элементінің a жиынына тиесілігін тексеру

```
a < x < b
(x > a) және (x < b) ге тең

1.6 кесте – Логикалық операторлар (AND, OR, NOT)
```

Оператор Сипаттамасы Мысал Нәтиже and Екі шарт та True болса ғана True True and True True or Ең болмағанда бірі True болса True False or True True not Қарама-қарсы мән not True False

```
x = 5
print(x > 3 and x < 10)  # True
print(x < 3 or x > 10)   # False
print(not (x == 5))      # False
```

Логикалық мәндер мен бүтін сандар Python-да True - 1-ге, ал False - 0-ге тең.

```
print(True + True)   # 2
print(False + 5)     # 5
print(True * 10)     # 10

bool() функциясы
Кез келген мәнді логикалық мәнге түрлендіру үшін bool() функциясы қолданылады.
print(bool(0))        # False
print(bool(5))        # True
print(bool(""))       # False
print(bool("Python")) # True
print(bool([]))       # False (бос тізім)
print(bool([1, 2]))   # True

Python–да "бос мәндер" (0, '', [], None) – әрқашан False.
```

Логикалық мәндер if шарттарында

```
age = 18
if age >= 18:
    print("Сіз кәмелетке толдыңыз!")
```

```
else:
    print("Кешіріңіз, сіз әлі жассыз.")
```

1.4 Айнымалы (Variables)

Айнымалы - белгілі бір мәнді сақтау үшін қолданылатын атауы бар жад ұяшығы. Айнымалы атауы қысқа болуы мүмкін (мысалы, x немесе y), не болмаса мағынасы кеңінен сипатталған атаумен де белгіленуі мүмкін. Python тілінде айнымалы атауларын жазу барысында белгілі бір ережелерді сақтау қажет. Айнымалы атауы санмен басталмайды және тек латын әріптерімен, сандармен және төменгі сызықшамен жазылады (A-Z, a-z, 0-9 және _). 1.10-суретте меншіктеу операторына қатысты мысал көрсетілген. Бұл оператор '=' таңбасы арқылы беріледі. Егер айнымалы алдын ала анықталмаса, бағдарлама «белгісіз мән» туралы хабарлама береді.

1.10 СУРЕТ - АЙНЫМАЛЫЛАРДЫҢ МӘНІН АУЫСТЫРУ МЫСАЛЫ КӨРІНІСІ

Бірнеше айнымалыны бір уақытта бір айнымалыға немесе бір қатарда бірнеше айнымалыға мән меншіктеуге болады (1.11-сурет).

1.11 СУРЕТ - БІРНЕШЕ АЙНЫМАЛЫҒА БІР МӘНДІ МЕНШІКТЕУ КӨРІНІСІ

Айнымалылар үш негізгі бөліктен тұрады: айнымалы атауы (идентификатор), оның мәні және типі. 1.12-суретте age айнымалысы - бүтін сан ретінде, ал money айнымалысы - нақты сан ретінде көрсетілген. Айнымалы мәнін беру кезінде бас және кіші әріптердің айырмашылығы маңызды, бұл типті анықтауға әсер етуі мүмкін. Айнымалыны анықтағанда ең алдымен оның алғашқы символы қарастырылады. Егер бұл бірінші таңба сан болса, бағдарлама оны қате ретінде қабылдайды.

1.12 СУРЕТ - АЙНЫМАЛЫ МӘНІН АНЫҚТАУ КӨРІНІСІ

Сондай-ақ, айнымалы атауы ретінде true, false және басқа да кілттік сөздерді пайдалануға болмайды. Себебі, бұл сөздер Python тілінде арнайы мағынаға ие. Кілттік сөздердің тізімі 1.6-кестеде берілген. Оларды жазу кезінде әдетте түсі өзгеше болып тұрады, бұл арқылы басқа сөздерден оңай ажыратуға болады.

1.6 КЕСТЕ - PYTHON ТІЛІНДЕГІ КІЛТТІК СӨЗДЕР

False	Class	Finally	is	return	None	Continue	for	
lambda	try	True	def	from	nonlocal	while	and	del
global	not	with	as	elif	if	or	yield	assert
else	import	pass						

break except in raise

Мысал ретінде max функциясын алуға болады - ол нақты бір қызмет атқарады. Егер бұл атауды айнымалы ретінде қолдансақ, онда бастапқы функция өз жұмысын тоқтатады (1.13-сурет).

1.13 СУРЕТ - АЙНЫМАЛЫҒА КІЛТТІК СӨЗДЕРДІ ҚОЛДАНБАУ КӨРІНІСІ

```
input() функциясы
```

Python тілінде дерек енгізу үшін input() функциясы қолданылады. Бұл функция шақырылған кезде бағдарлама уақытша орындалуын тоқтатады және пайдаланушының дерек енгізуін күтеді. Пайдаланушы қажетті мәтінді енгізіп, Enter пернесін басқаннан кейін, input() функциясы сол мәнді қабылдап, оны әрі қарай өңдеу үшін бағдарламаға береді. Мысалдар:

```
    •    a = input() – енгізілген мән a айнымалысына мәтін түрінде (жол ретінде) жазылады.
```

- a = int(input()) - пайдаланушы енгізген мән бүтін санға түрлендіріліп, a айнымалысына сақталады.
- a = float(input()) - енгізілген дерек нақты сан ретінде қабылданып, a айнымалысына жазылады.

1.14 СУРЕТ - INPUT КОМАНДАСЫ КӨРІНІСІ

Егер input() командасы интерактивті режимде орындалса, бастапқыда экранда ештеңе көрінбейді. Компьютер пайдаланушыдан дерек енгізуін күтеді және Enter пернесі басылғаннан кейін ғана енгізілген ақпарат экранда көрсетіледі (1.14-сурет).

Шартты операторды қолдану Мысалы, x саны берілсін, және оның абсолюттік мәнін (модулін) анықтау қажет болсын. Бағдарлама x > 0 болған жағдайда - x мәнін, ал кері жағдайда - оның теріс мәнін (қарама-қарсы шамасын) көрсетуі керек (1.15-сурет).

1.15 СУРЕТ - ШАРТТЫ КОМАНДАСЫ КӨРІНІСІ

1.5 Деректер құрылымы (Data Structures)

Деректер құрылымы - деректерді белгілі бір ретпен сақтап, оларға ыңғайлы түрде қол жеткізуге мүмкіндік беретін Python-дағы контейнерлер. Python тіліндегі ең жиі қолданылатын деректер құрылымдары:

- Тізім (List)
- Кортеж (Tuple)
- Жиын (Set)
- Сөздік (Dictionary)

Python тілінде деректер құрылымдарының екі негізгі түрі қолданылады: бірізділік (тізбектер) және бейнелеу түріндегі құрылымдар, соңғылары көбіне сөздіктер деп аталады. Сөздік құрылымы «кілт - мән» жұбын сақтау үшін қолданылады (мысалы, «Тегі - Мекенжайы»), осылайша олар ассоциативті массивтерді жасауға мүмкіндік береді. Тізбектер екіге бөлінеді: өзгеретін және өзгермейтін. Тізбектің өзгергіштігі - оның элементтерін қосуға немесе жоюға болатындығын білдіреді, яғни тізбек элементтерінің саны өзгеруі мүмкін. Python тілінде деректер құрылымдарымен жұмыс істеуге арналған түрлі функциялар мен әдістер қарастырылған. Олар арасындағы басты айырмашылық - синтаксисінде, яғни жазылу тәсілінде байқалады.

```
Тізім (List)
```

Көптеген бағдарламаларда жеке айнымалылармен емес, біртұтас айнымалылар жиынтығымен жұмыс жасау қажет болады. Мысалы, бағдарлама файлдан немесе пернетақтадан білім алушылардың тізімін оқып, сол деректерді өңдеуі мүмкін. Бұл жағдайда сыныптағы оқушылар саны өзгерсе де, бағдарлама кодын қайта жазудың қажеті болмауы тиіс. Осындай мәліметтерді сақтау үшін Python тілінде тізім (list) құрылымы пайдаланылады. Көптеген басқа бағдарламалау тілдерінде бұл ұғым массив деп аталады. Тізім - 0-ден басталатын индекстері бар элементтердің реттелген жиыны. Тізімді құру үшін элементтер шаршы жақша ішіне жазылады. Мысалы:

```
Primes = [2, 3, 5, 7, 11, 13]
```

Бұл тізім 6 элементтен тұрады, яғни:

- Primes[0] == 2
- Primes[1] == 3

- ...
- Primes[5] == 13

Сол сияқты, Rainbow деген тізім 7 жолдық элементтен тұруы мүмкін (мысалы, түстердің атаулары). Python тілінде тізім элементтеріне теріс индекстер арқылы да қол жеткізуге болады, яғни санау соңынан басталады:

- Primes[-1] == 13
- Primes[-6] == 2

Тізімнің ұзындығын анықтау үшін len() функциясы қолданылады. Мысалы, len(Primes) нәтижесі - 6.

```
Кортеж (Tuple)
Өзгермейтін (immutable), тізім сияқты жұмыс істейді, бірақ өзгертуге болмайды.
colors = ("red", "green", "blue")
print(colors[0])  # red
```

Артықшылықтары: Жылдам жұмыс істейді

```
Қауіпсіз (өзгермейді)
```

Кілттер ретінде қолдануға болады

```
Жиын (Set)
Қайталанбайтын элементтер жиыны, ретсіз (unordered).
nums = {1, 2, 3, 2, 1}
print(nums)  # {1, 2, 3}
```

Функция Түсініктеме

```
add(x)
```

Қосу

```
remove(x)
```

Өшіру

```
union(set2)
```

Біріктіру

```
intersection(set2)
```

Қиылысу

```
difference(set2)
```

Айырма

```
Сөздік (Dictionary)
Кілт–мән (key–value) жұбы түрінде сақтайды.
student = {
```

"name": "Ainur", "age": 17, "grade": "A" }

```
print(student["name"])  # Ainur
```

Функция Түсініктеме

```
keys()
```

Барлық кілттер

```
values()
```

Барлық мәндер

```
items()
```

Кілт-мән жұптары

```
get("key")
```

Кілт арқылы мәнді алу

1.7 КЕСТЕ -ДЕРЕКТЕР ҚҰРЫЛЫМДАРЫНЫҢ САЛЫСТЫРМАСЫ

Құрылым	Өзгеруі	Реттілігі	Қайталану	Пішімі	List	Иә	Ретті
Иә	[1, 2, 3]	Tuple	Жоқ	Ретті	Иә	(1, 2, 3)	Set	Иә
Ретсіз	Жоқ	{1, 2, 3}	Dict	Иә	Ретсіз	Кілт қайталанбайды	{"name": "Ali"}

Мысалдар:

1. Тізімге 5 зат қосып, сұрыптап шығарыңыз python КопироватьРедактировать items = [] for i in range(5): item = input("Элемент енгізіңіз: ") items.append(item) items.sort() print(items)

1. Tuple арқылы айлар тізімін шығару python КопироватьРедактировать months = ("Қаңтар", "Ақпан", "Наурыз") for m in months: print(m)

1. Жиын арқылы қайталанатын сандарды алып тастау python КопироватьРедактировать numbers = [1, 2, 2, 3, 3, 4] unique = set(numbers) print(unique)

1. Сөздікпен оқушы мәліметін енгізу python КопироватьРедактировать student = {} student["name"] = input("Аты: ") student["age"] = int(input("Жасы: ")) print(student)

```
Жолдар (Strings)
```

Жол - бұл символдар тізбегі. Python-да жолдар да тізім сияқты өңделеді. Мысалы:

```
S = 'Білім'
print(len(S))  # Нәтиже: 5
```

Жолдағы әрбір символ индекс арқылы алынады, және индекстеу 0-ден басталады. Python жолдары тек ағылшын әріптерінен емес, кез келген алфавиттің әріптерінен құралуы мүмкін. Дегенмен, айнымалылар мен синтаксис тұрғысынан жиі ағылшын әріптерін қолдану ұсынылады. Жолдар қос немесе жалғыз тырнақшада жазылады: 'text' немесе "text". Python тілінде жолдар ұзындығына шектеу қойылмайды - шектеу тек компьютер жады көлемімен анықталады. Яғни, бірнеше мың беттік мәтінді де бір жол ретінде сақтауға болады.

Түрлендіру функциялары Сандарды жолға және керісінше түрлендіру үшін арнайы функциялар қолданылады: Мысал Түсіндірме

```
str(123)
```

Бүтін санды жолға айналдырады: '123'

```
int('123')
```

Жолды бүтін санға түрлендіреді: 123

```
float('12.34')
```

Жолды нақты санға түрлендіреді: 12.34

1.8-КЕСТЕ. ТҮРЛЕНДІРУ ФУНКЦИЯЛАРЫ МЕН ОПЕРАЦИЯЛАРЫНЫҢ СИПАТТАМАСЫ

Функция	немесе операция	Сипаттамасы мен нәтижесі	len(s)
S жолының ұзындығы таңбалар саны ретінде есептеледі			

s1 + s2 Конкатенация. s1 жолының соңында s2 жолы қосылады, нәтижесінде жаңа жол шығады, - мысал, ' сіз ' + ' жыл ' → 'сіз жыл'

```
s *n (немесе n *s)
s жолының N есе қайталануы, нәтижесінде–жаңа жол бар, мысалы('мен' *2 → 'меннен')
```

```
s [ i ]
```

S элементін I нөмірінен таңдау, нөмірлеу 0-ден басталады. Нәтижесі- символ болады. Егер i < 0 болса, есеп соңынан жүреді (жолдың бірінші символы 0 нөмірінен басталып, соңғысы -1 нөмірі болады). Мысалы: 's= 'кітап'

```
s [2] → 'т'
s[–2] → 'а'
```

Тізімдерді құру және элементтерін санау тәсілдері Тізімдерді жасау мен оларға әртүрлі операциялар қолданудың бірнеше әдістері бар. Мысалы, тізімдерді біріктіру - екі немесе одан да көп тізімдерді бір бүтінге қосу (конкатенациялау), сондай-ақ тізімді белгілі бір санға көбейту арқылы оның элементтерін қайталау сияқты әрекеттер орындалады (1.16-сурет).

1.16 СУРЕТ - ТІЗІМДЕРДІ ҚОСУ КӨРІНІСІ

```
Қырқу (slice) ұғымы
```

Қырқу - жолдан немесе басқа деректер құрылымдарынан (мысалы, тізімдер, кортеждер) белгілі бір таңбаларды немесе элементтерді бөліп алу операциясы. Қырқудың ең қарапайым түрі - қатардағы бір символды таңдау. Мысалы, t[i] - бұл t жолындағы i-ші символды білдіреді, мұндағы индекстеу 0-ден басталады. Егер

S = 'СӘЛЕМ' ДЕГЕН ЖОЛ БЕРІЛСЕ, ОНДА:

```
•   S[0] – 'С'
•   S[1] – 'ә'
•   S[2] – 'л'
```

- S[3] – 'Е'
- S[4] – 'М'

Жолдағы (және басқа деректер құрылымдарындағы) таңбаларға сәйкесінше индекс беріледі.

Егер индекстеу нөмірі жол ұзындығынан (яғни, len(S)) үлкен немесе тең болса, немесе теріс мән дұрыс көрсетілмесе, онда IndexError: string index out of range қатесі пайда болады. Индекстер теріс мәнді алса, санау жолдың соңынан жүргізіледі, яғни:

- S[–1] – соңғы символ, яғни 'М'
- S[–2] – соңғыдан екінші символ, 'Е'
- S[–3] – 'Л'
- S[–4] – 'Ә'
- S[–5] – 'С'

1.9 КЕСТЕ - ИНДЕКСТІҢ ТЕРІС МӘНІ

Қатар	С
Ә	Л
Е	М
Индекс	S(0)
S(1)	S(2)
S(3)	S(4)
Индекс	S(-5)
S(-4)	S(-3)
S(-2)	S(-1)

find() әдісі

find() әдісі - жол ішінде берілген қосымша жолды іздеу үшін қолданылады. Бұл әдіске ізделетін мәтін жолы параметр ретінде беріледі. Егер қосымша жол табылса, онда оның алғашқы кездесетін орнындағы индекс қайтарылады. Ал егер ізделген мәтін жолда жоқ болса, функция -1 мәнін қайтарады (1.17-сурет).

1.17 СУРЕТ - FIND ӘДІСІ АРҚЫЛЫ ЖОЛДЫ ТАБУ КӨРІНІСІ

rfind әдісі осы қатардың соңғы индекснен («оң жақтан іздеу») табады (1.18 –сурет).

1.18 СУРЕТ - RFIND ӘДІСІ КӨРІНІСІ

Қайталау әдісі (Replace)

replace() әдісі - бір жолдағы барлық белгілі бір символдарды немесе таңбалар тізбегін басқа таңбалармен ауыстыруға арналған. Яғни, жолдағы барлық сәйкес элементтер жаңа мәнге алмастырылады (1.19-сурет).

1.19 СУРЕТ - REPLACE ӘДІСІ КӨРІНІСІ

Егер replace әдісі былай қолданылса: s.replace(old, new, count), онда жолдағы барлық таңбалар емес, тек алғашқы көрсетілген count санына сәйкес келетін элементтер ғана ауыстырылады. Мысалы, егер count=2 болса, онда бірінші екі табылған орын алмастырылады (1.20-сурет).

1.20 СУРЕТ - REPLACE ӘДІСІНЕ САНДЫ БЕРУ КӨРІНІСІ

```
Count() әдісі
```

count() әдісі - жол ішінде берілген таңбаның немесе таңбалар тізбегінің қанша рет кездесетінін есептейді (1.21-сурет).

1.21 СУРЕТ - ЖОЛДА БЕРІЛГЕН ӘРІПТІ ЕСЕПТЕУ КӨРІНІСІ

Әдіс - бұл объектінің нақты типіне тән функция болып табылады. Яғни, әр түрлі типтегі объектілерге арналған әдістер әртүрлі болады. Әдісті шақыру форматы: объект (мысалы, жол) нүктемен әдіс атауы және қажет болған жағдайда аргументтер беріледі (1.22-сурет).

1.22 СУРЕТ - ЖОЛДАРҒА ОРЫНДАЛАТЫН ӘДІСТЕР КӨРІНІСІ

Нақты сандарды енгізу Пернетақтадан нақты сандарды алу үшін input() функциясының алдынан float түрлендіргішін қою қажет (1.23-сурет):

```
a = float(input())
```

1.23 СУРЕТ - НАҚТЫ САНДАРДЫ ЕСЕПТЕУ МЫСАЛЫ КӨРІНІСІ

1.6 Math кітапханасы

Python тілінде нақты сандармен күрделі есептеулер жүргізу үшін арнайы қосымша функциялардан тұратын math модулі қолданылады. Бұл кітапханада математикалық амалдарды орындауға арналған көптеген функциялар бар. Егер бұл модуль алдын ала импорттанса, әр жол сайын math деп көрсетудің қажеті болмайды (1.24-сурет).

1.24 СУРЕТ - MATH КІТАПХАНАСЫН ПАЙДАЛАНУ КӨРІНІСІ

Мысалы, берілген санды ең жақын үлкен бүтін санға дейін дөңгелектеу қажет болса, math модуліндегі ceil() функциясын қолдануға болады. Бұл функция бір аргумент қабылдайды және былай жазылады:

```
math.ceil(x)
```

Мұндағы x - нақты сан, айнымалы немесе арифметикалық өрнек болуы мүмкін. Бұл тәсіл арқылы кез келген нақты санды жоғары жаққа қарай дөңгелектеуге болады. Айта кету керек, кейбір функциялар - мысалы, int(), round(), және abs() - Python-да стандартты түрде берілген, сондықтан оларды пайдалану үшін math модулін импорттау қажет емес (1.10-кесте).

1.10 КЕСТЕ - СТАНДАРТТЫ ФУНКЦИЯЛАР

Функция	Сипаттамасы
Дөңгелектеу	int(x)

Санды нөлге қарай дөңгелектейді. Бұл стандартты функция, оны пайдалану үшін math модулін қосудың қажеті жоқ.

```
round(x)
```

Санды ең жақын бүтін санға дейін дөңгелектейді. Егер санның бөлшек бөлігі 0.5 тең болса, онда сан ең жақын жұп санға дейін дөңгелектенеді.

```
round(x, n)
```

X санын нүктеден кейін n белгіге дейін дөңгелектейді. Бұл стандартты функция, оны пайдалану үшін math модулін қосудың қажеті жоқ.

```
floor(x)
Floor (1.5) = = 1, floor(–1.5) = = –2 санын төмен санға қарай дөңгелектейді
```

```
ceil(x)
Floor (1.5) = = 1, floor (-1.5) = = -2жоғарғы санға қарай дөңгелектейді.
trunc(x)
Нөлге қарай дөңгелектеу (int функциясы сияқты).
abs(x)
Нөлге қарай дөңгелектеу (int функциясы сияқты).
fabs(x)
Модуль (абсолюттік шама). Бұл функция әрқашан float түрін қайтарады.
```

Түбір, дәрежелер, логарифмдер

```
sqrt(x)
Квадратты түбір. Қолданылуы: sqrt(x)
pow(a, b)
Дәрежелеу ab. Қолданылуы: pow(a,b)
exp(x)
Экспонента ex қайтарады. Қолданылуы: exp(x)
log(x)
Логарифм. Log(x, b) шақыру кезінде b негізі бойынша логарифмді қайтарады.
log10(x)
```

Ондық логарифм e Негізгі логарифм e271828. Тригонометрия

```
sin(x)
```

Радиандарда берілген бұрыштың синусы

```
cos(x)
```

Радиандарда берілген бұрыштың косинусы

```
tan(x)
```

Радиандарда берілген бұрыштың тангенсі

```
asin(x)
```

Арксинус, радианның мәнін қайтарады

```
acos(x)
```

Арккосинус, радианның мәнін қайтарады

```
atan(x)
```

Арктангенс, радианның мәнін қайтарады

```
atan2(y, x)
 (x, y) координаттары бар нүктенің полярлық бұрышы (радиандарда)
hypot(a, b)
```

a және b катеттері бар тікбұрышты үшбұрыштың гипотенузасының ұзындығы.

```
degrees(x)
```

Радианмен көрсетілген бұрышты градусқа өзгертеді.

```
radians(x)
```

Градуспен көрсетілген бұрышты радиандарға түрлендіреді.

Мысалдар:

```
import math
print("pi =", math.pi)
print("Квадрат түбірі 25 =", math.sqrt(25))
print("Екі санының 5 дәрежесі =", math.pow(2, 5))
print("5.3-тің төбесі =", math.ceil(5.3))
print("5.3-тің төменгі мәні =", math.floor(5.3))
```

math модулі - күрделі математикалық операцияларды оңайлатуға арналған. Ол арқылы түбір табу, дәреже, логарифм, тригонометрия, т.б. амалдар орындалады. Мектеп математикасы мен олимпиадалар үшін өте пайдалы модуль.

Бақылау сұрақтары:

- Python бағдарламалау тілі қай жылы құрылды?
- Python тілінің авторы кім?
- Python қандай программалау тілдерінің үлгісінде жасалды?
- Python тілінің негізгі артықшылықтары қандай?
- Python тілінің қандай негізгі қолдану аймақтары бар?
- Python-ды қалай компьютерге жүктеп алуға болады?
- Python орнату кезінде қандай нұсқаларды таңдауға болады?
- Python тілінің REPL деген не?
- Python бағдарламасын іске қосу үшін қандай команданы теремiз?
- Python тілінде комментарийлер қалай жазылады?
- Python тілін орнатқаннан кейін қандай негізгі файлдар пайда болады?
- Python тілінің негізгі IDE-лері қандай?
- Python тілінің қандай интерпретаторлары бар?
- Python тілінің қандай дербес кітапханалары бар?
- Python-ның ашық коды (open source) деген не?
- Python тілінде қандай сандық типтер бар?
- int және float типтерінің айырмашылығы неде?
- Python-да бүтін санды қалай жазамыз?
- Python-да нақты сан қалай көрсетіледі?
- Компьютерде сандардың сақталуы қандай форматта болады?
- Арифметикалық операциялардың Python синтаксисі қандай?
- Қай бөлу операциясы нақты нәтижені қайтарады?
- Целочисленный бөлу (//) деген не?
- Пайдаланушыдан санды қалай енгізуге болады?
- Санның абсолют мәнін есептеу үшін қандай функция қолданылады?
- Python тілінде рандомдық санды қалай жасауға болады?
- Санның дәрежесін қалай есептейміз?
- Сандарды түрлендіру деген не?

- Комплекс сандар дегеніміз не және Python-да қалай қолданылады?
- Сандармен жұмыс істеу кезінде қандай қателіктер кездесуі мүмкін?
- Логикалық типтің екі мәні қандай?
- Python тілінде True және False қалай жазылады?
- Логикалық операция AND қалай жұмыс істейді?
- Логикалық операция OR қалай жұмыс істейді?
- Логикалық NOT операциясының мағынасы қандай?
- Логикалық мәндерді қайда қолданамыз?
- Логикалық өрнектер қандай жағдайда True болады?
- Python-да шартты оператор қалай жұмыс істейді?
- Пайдаланушы енгізген мәнді логикалық тексеру қалай жүргізіледі?
- Логикалық операциялардың нәтижесін қалай басып шығаруға болады?
- Айнымалы деген не?
- Python тілінде айнымалыны қалай жариялаймыз?
- Айнымалы атауына қандай талаптар қойылады?
- Айнымалыға мәнді қалай меншіктейміз?
- Айнымалының мәнін қалай өзгертуге болады?
- Айнымалылардың типі қалай анықталады?
- Айнымалыларды атау кезінде қандай қателіктер болмауы тиіс?
- Бір айнымалыға бірнеше мәнді қалай меншіктеуге болады?
- Айнымалы мәнін экранға қалай шығаруға болады?
- Айнымалылардың жадтағы рөлі қандай?
- Деректер құрылымы дегеніміз не?
- Python-да қандай негізгі деректер құрылымдары бар?
- Тізім (list) деген не?
- Кортеж (tuple) деген не?
- Сөздік (dictionary) деген не?
- Массив пен тізімнің айырмашылығы неде?
- Тізім элементтерін қалай қосамыз?
- Сөздікке элементтерді қалай енгіземіз?
- Тізім элементтерінің индекстері қалай саналады?
- Тізімнен элементті қалай жоямыз?
- Тізімдегі элементтерді қалай ауыстырамыз?
- Кортеждің негізгі ерекшелігі неде?
- Сөздіктің кілттері мен мәндері қалай ұйымдастырылған?
- Тізімдер мен сөздіктердің қолдану салалары қандай?
- Деректер құрылымдары бағдарламалауда қандай маңызы бар?
- Тізім дегеніміз не?
- Тізімді қалай құрамыз?
- Тізім элементтеріне қалай қол жеткіземіз?
- Тізімді қалай өзгертуге болады?
- Тізімге элемент қосу әдістері қандай?
- Тізімнен элементті қалай жоюға болады?
- Тізімді қалай сұрыптауға болады?
- Тізімнің ұзындығын қалай білеміз?
- Тізімді қайталанатын элементтерден қалай тазартамыз?

- Тізімнің элементтерін қалай цикл арқылы шығарамыз?
- Тізімді басқа тізіммен қалай біріктіруге болады?
- Тізімді терең көшіру мен жазық көшіру арасындағы айырмашылық неде?
- Тізімдегі элементтерді индекстеу қалай жүзеге асады?
- Тізімнің бос екенін қалай тексереміз?
- Тізімдердің қолдану салалары қандай?
- Math кітапханасы деген не?
- Math кітапханасын қалай импорттаймыз?
- Math кітапханасындағы негізгі функцияларды атаңыз.
- Пи (π) саны қалай шақырылады?
- Квадрат түбірді қалай есептейміз?
- Дәрежеге көтеру функциясы қалай жұмыс істейді?
- Логарифм дегеніміз не және оны қалай есептейміз?
- Math модуліндегі тригонометриялық функцияларды атаңыз.
- Math.ceil() және math.floor() функциялары не істейді?
- Math.fabs() функциясының мақсаты қандай?
- Math модулінде кездейсоқ сандарды генерациялай аламыз ба?
- Math модулін қолданғанда қандай мүмкін болатын қателіктер болады?
- Math.factorial() функциясы не істейді?
- Math.gcd() функциясы не үшін қолданылады?
- Math модуліндегі radians() және degrees() функциялары не істейді?
- Math.isclose() функциясы қандай жағдайларда қажет?
- Math модулінің басқа Python модулдерінен айырмашылығы неде?
- Math модулін қолдану арқылы есептеулердің дәлдігі қалай қамтамасыз етіледі?
- Math модулінің ең танымал қолдану мысалдары қандай?
- Math кітапханасын өзіңіз жазған бағдарламада қалай қолданған болар едіңіз?

2 PYTHON тілінің ГРАФИКАЛЫҚ негізі

2.1 Python тілінің графикалық мүмкіндіктері. Turtle кітапханасы

Қазіргі заманғы технологиялардың дамуымен күнделікті өмірде қолданылатын көптеген құрылғылар - мысалы, ұялы телефондар, автокөліктер, қол сағаттары, ойын құрылғылары, тренажерлар, медицина жабдықтары, өндірістік станоктар, тіпті құттықтау хаттары мен роботтар - бәрі де өз ішіне компьютерлік компоненттерді қамтиды. Python тілі арқылы графикалық элементтер мен пішіндер салып, оларды қозғалысқа келтіріп, анимациялар жасауға болады. Python графикамен жұмыс істеуге арналған бірнеше кітапханаларды қолдайды, солардың көмегімен күрделі графикалық интерфейсі бар қолданбалар жасауға мүмкіндік береді. Бұл бөлімде біз Python тіліндегі графикамен жұмыс істеудің негіздерімен танысамыз: Turtle модулінің көмегімен қарапайым графикалық элементтер салып, оларды экранда қозғалту және математикалық немесе физикалық модельдерді жасау үшін Tkinter кітапханасын пайдалану жолдарын үйренеміз (2.1 -сурет).

2.1 СУРЕТ - АЛТЫ БҰРЫШТЫ ОЮ-ӨРНЕК

Кейбір бағдарламалар мыңдаған жолдан тұрса, енді біреулері небәрі бірнеше жолмен шектеледі. Мысалы, 2.2-суретте көрсетілген NiceHexSpiral.py бағдарламасы осындай қысқа кодтарға жатады.

2.2 СУРЕТ - NICEHEXSPIRAL.PY БАҒДАРЛАМАСЫНЫҢ КӨРІНІСІ

Бұл шағын бағдарлама 2.2-суретте көрсетілгендей түрлі-түсті спираль үлгісін салады. Turtle (немесе "Тасбақа") - Python тіліне кірістірілген, графикалық сызбалар жасауға арналған кітапхана. Ол экранда сурет салатын қалам секілді жұмыс істейді. Бұл кітапханамен жұмыс істеу қағазға сурет салу үдерісіне ұқсайды: алдымен «қағаз» ретінде экран дайындалады, кейін қалам параметрлері орнатылады және соңында сурет салынады. Сурет салу кезінде экрандағы тасбақа тәрізді объект қозғалады және оның қозғалыс ізі көрінетін болғандықтан, бұл кітапхана turtle деп аталған. Бұл

идея алғаш рет 1967 жылы Уолли Ферцайг, Сеймур Пейперт және Синтия Соломон тарапынан жасалған және ол бастапқыда Logo бағдарламалау тілі құрамында пайда болған. Logo тілі өз заманында балалар арасында кең танымал болды, себебі ол арқылы олар экранда оңай әрі қызықты графикалар сала алды. Сол тәсілді Python-дағы Turtle кітапханасы жалғастырады: онда экранда қозғалатын кішкене объект арқылы әртүрлі сызбалар мен пішіндер жасауға болады. Turtle модулі әсіресе жаңадан бастаушыларға арналған. Ол Python тілін үйренуді көңілді әрі визуалды түрде ұғынуға мүмкіндік береді. Бұл кітапхана арқылы оқушылар ғана емес, тәжірибелі бағдарламашылар да түрлі күрделі фигуралар, көрнекі суреттер және графикалық ойындар жасап шығара алады. Бұл бөлімде қарапайым әрі қысқа бағдарламалар жазу арқылы күрделі және әдемі ою-өрнектерді құру тәсілдері қарастырылады. Бұл үшін Python тіліндегі turtle (тасбақа) графикасы пайдаланылады. Turtle графикасымен жұмыс істегенде, бағдарламалаушы виртуалды тасбақаға нақты экранда қалай қозғалуы керек екенін сипаттайтын командалар жиынтығын жазады. Тасбақа өзімен бірге қалам ұстайды, және бағдарлама барысында оған қай бағытта қозғалу керек екенін және қозғалу кезінде қаламды түсіріп, із қалдыруын нұсқауға болады. Түзу жолмен қозғалу командаларын пайдалана отырып, тасбақа арқылы күрделі суреттерді жасауға болатынына көз жеткізуге болады. Turtle графикасы арқылы небәрі бірнеше жол код жазу арқылы тартымды өрнектер мен суреттерді салуға болады. Сонымен қатар, тасбақаның қалай қозғалатынын қадағалап, әр жазылған команданың оның бағыты мен қозғалыс траекториясына қалай әсер ететінін түсінуге болады. Бұл өз кезегінде бағдарлама логикасын тереңірек меңгеруге көмектеседі. Мысал ретінде, координаталық жазықтықтың (0, 0) нүктесінен басталатын виртуалды робот-тасбақаны көз алдыңызға елестетіңіз. Бағдарламада turtle.forward(15) командасын бергенде, тасбақа экранда алға қарай 15 пикселге жылжиды және артында сызық қалдырады. Ал turtle.right(25) командасы оған сағат тілі бағытымен 25 градусқа бұрылуды бұйырады. Осындай қарапайым командаларды біріктіре отырып, күрделі суреттер мен геометриялық фигураларды оңай жасауға болады. Turtle модулі Python тілінде графикалық примитивтермен жұмыс істеуді объектіге бағытталған және процедуралық тәсілдер арқылы жүзеге асыруға мүмкіндік береді. Бұл модуль негізгі графикалық мүмкіндіктер үшін tkinter кітапханасына сүйенеді, сондықтан оны пайдалану үшін Python-ның tkinter-мен жабдықталған нұсқасы қажет. Объектіге бағытталған интерфейс негізінен екі негізгі класпен жұмыс істейді:

- TurtleScreen класы - графикалық терезелерді тасбақа сурет салатын аймақ ретінде анықтайды. Бұл классты қолдану үшін tkinter элементі аргумент ретінде берілуі тиіс. Егер Turtle модулі басқа қосымшаның құрамында қолданылса, онда Canvas немесе ScrolledCanvas элементтері пайдаланылады. Ал Python ортасында жеке тасбақа графикасын пайдалану кезінде, Screen() функциясы осы кластың ішкі бір экземплярын жасап береді. Бұл объект бір реттік болғандықтан, оны қайтадан мұрагерлікке беруге болмайды.
- TurtleScreen немесе Screen интерфейстерінде барлық әдістер процедуралық тәсілде функция ретінде де қолжетімді. RawTurtle (кейде "Raven" деп аталады) - бұл TurtleScreen бетіне сурет салатын нақты тасбақа нысаны. Оны құру үшін міндетті түрде сурет салынатын аймақ - Canvas, ScrolledCanvas немесе TurtleScreen - аргумент ретінде берілуі керек. RawTurtle класының Turtle ішкі класы (бейресми түрде "қалам" деп аталады) автоматты түрде screen нысанын қолданады, егер ол әлі жасалмаған болса, онда оны өзі құрып алады. Бұл кластағы барлық әдістер де процедуралық интерфейс арқылы функциялар ретінде қолданыла алады. Процедуралық интерфейс жалпы Screen және Turtle кластарының әдістеріне негізделген функциялар жиынтығын ұсынады. Бұл функциялар әдістермен бірдей атауға ие. Егер қандай да бір функция screen әдісі арқылы шақырылса, онда экран нысаны автоматты түрде пайда болады. Сол сияқты, Turtle әдісінен алынған функцияны алғаш қолданғанда тасбақа нысаны да автоматты түрде құрылады. Экранда бірнеше тасбақамен жұмыс істеу қажет болған жағдайда, объектіге бағытталған тәсіл қолданылуы тиіс. Ал процедуралық интерфейс - бұл Screen және Turtle кластарындағы әдістер негізінде құрылған функциялар жиынтығы. Бұл функциялардың атаулары сәйкес кластағы әдістермен бірдей болады. Әр жолы Screen функциясы шақырылған кезде, экран объектісі автоматты түрде жасалады. Сол сияқты, Turtle функциясы орындалғанда, аты жоқ тасбақа нысаны бірден құрылады. Бір уақытта бірнеше тасбақаны пайдалану үшін объектіге бағытталған интерфейсті қолдану міндетті екенін есте сақтау қажет. Python тіліндегі Turtle кітапханасын қолдану барысында біз оның негізгі ұғымдарымен танысамыз, тасбақаны экранда қалай пайдалану керектігін үйренеміз, негізгі командаларды меңгереміз және қысқа, бірақ әдемі графикалық сызбалар жасаймыз. Turtle модулі Python-ға кіріктірілген, яғни оны бөлек жүктеп орнату қажет емес. Бағдарламалауды бастамас бұрын басты екі әрекетті орындау керек: Python ортасын дайындау және turtle модулін бағдарламаға импорттау. Осыдан кейін тасбақа графикасымен жұмыс істеуге толық мүмкіндік ашылады. Turtle (Тасбақа) - Python тіліндегі графикамен жұмыс істеуге арналған кітапхана, ол сурет салу мен дизайн жасау үшін қажетті негізгі әдістер мен функцияларды қамтиды. Бұл кітапхананы қолдану үшін ең алдымен оны Python бағдарламасына қосу керек:

```
import turtle
```

Осы команданы орындағаннан кейін Turtle модуліндегі барлық функциялар мен әдістерді қолдануға мүмкіндік ашылады. Ең алдымен, сурет салуға арналған арнайы графикалық терезе жасау қажет. Бұл терезені айнымалы арқылы инициализациялау арқылы іске асыруға болады:

```
s = turtle.getscreen()
```

Осылайша, біз экранда сызбалар орындауға дайын интерфейс құрамыз.

2.3 СУРЕТ -TURTLEГРАФИКАСЫНЫҢ КӨРІНІСІ

(2.3-суретте) көрсетілгендей, экранның ортасында кішкентай үшбұрыш түріндегі объект орналасқан - бұл тасбақа. Бұл тасбақаны экран бетінде әртүрлі бағытта қозғалта отырып, түрлі пішіндер мен дизайндар салуға болады. Оның бірнеше өзгермелі қасиеттері бар: түсін, өлшемін, қозғалыс жылдамдығын баптауға болады. Егер нақты бағыт көрсетілмесе, тасбақа ағымдағы бағыты бойынша қозғалады.

```
Python тіліндегі Turtle (Тасбақа) кітапханасымен бағдарламалау негіздерін қарастырайық.
```

Алдымен, тасбақаның экранда қалай қозғалатынын меңгеру қажет. Сонымен қатар, қалам арқылы оның қозғалысын және сызу қасиеттерін баптауға болады. Төменде тасбақаны экранда басқаруға арналған негізгі қозғалыс командалары берілген. Бұл командалар оның төрт бағытта қозғалуы үшін қолданылады:

```
    •    forward() – Алға жылжытады
    •    backward() – Артқа қозғайды
    •    left() – Солға бұрылады
    •    right() – Оңға бұрылады
```

Тасбақаның қозғалысы Turtle (тасбақа) экранында қозғалу кезінде өзі қарап тұрған бағытта алға немесе артқа жылжи алады. Бұл үшін арнайы функциялар қолданылады. Мысалы: forward(distance) немесе қысқаша turtle.fd(distance) - тасбақаны алға қарай берілген қашықтыққа жылжытады. Мұндағы distance - тасбақа жүруі тиіс арақашықтық, ол бүтін сан немесе нақты сан түрінде беріледі (қосымша үшін 2.4-суретке қараңыз). Бұл командалар экранда графикалық сызық сыза отырып, қозғалысты визуалды түрде көрсетуге мүмкіндік береді.

```
    •    import turtle  # Turtle модулін импорттау
```

- # Тасбақа объектісін жасау
- t = turtle.Turtle()
- # Тасбақаны алға 100 пиксельге жылжыту
- t.forward(100)
- # Графикалық терезені ашық ұстау
- turtle.mainloop()

2.4 СУРЕТ - TURTLE КӨРІНІСІ

Түсініктеме: 1-жолда turtle кітапханасы жүктеледі.

```
3–жолда тасбақа (Turtle) объектісі құрылады.
```

5-жолда тасбақа экранда өзінің ағымдағы бағытында 100 пиксель алға қозғалады. 7-жол экранды жабылудан сақтап, графика терезесін ашық қалдырады.

back(distance), turtle.bk(distance) немесе turtle.backward(distance) - бұл функциялар тасбақаны өзінің қарама-қарсы бағытына қарай жылжытады. Яғни, тасбақа артқа қозғалады, бірақ бағыты (немесе бұрылған бұрышы) өзгеріссіз қалады. Бұл оны тек кейін шегіндіру үшін қолданылады (қосымша үшін 2.5-суретке қараңыз).

```
    •    import turtle  # Turtle модулін импорттау
```

- # Тасбақа объектісін жасау t = turtle.Turtle()
- # Тасбақаны артқа қарай 100 пиксельге жылжыту t.backward(100)
- # Графикалық терезені ашық ұстау үшін turtle.mainloop()

2.5 СУРЕТ - TURTLE ҚАРАМА - ҚАРСЫ БАҒЫТҚА ЖЫЛЖЫТУ КӨРІНІСІ

Тасбақаны бұру әдістері right(angle) немесе turtle.rt(angle) - бұл функция тасбақаны көрсетілген градус мөлшерінде оңға бұрады. Мұндағы angle - бұру бұрышының мәні градуспен беріледі.

```
import turtle
```

Тасбақа экранын құру

```
t = turtle.Turtle()
```

Тасбақаның ағымдағы бағытын алу

```
print(t.heading())
```

Тасбақаны сағат тілі бағытымен 25 градусқа бұру

```
t.right(25)
```

Тасбақаның жаңа бағытын шығару

```
print(t.heading())
```

Экранды ашық ұстау үшін

```
turtle.mainloop()

left(angle) немесе turtle.lt(angle) – тасбақаны солға көрсетілген градусқа бұрады.
Төменде (2.6-суреттегі) мысал ретінде бұру командасын қолдану көрсетілген.
    •    import turtle  # Turtle кітапханасын импорттау
```

- # Тасбақа объектісін құру t = turtle.Turtle()
- # Тасбақаның ағымдағы бағытын алу және шығару print("Бастапқы бағыт:", t.heading())
- # Тасбақаны солға қарай 100 градусқа бұру t.left(100)
- # Жаңа бағытын шығару print("Бұрылыс кейінгі бағыт:", t.heading())
- # Терезені ашық ұстау turtle.mainloop() t.heading() - тасбақаның ағымдағы бағыт бұрышын градуспен қайтарады.
- t.left(100) - тасбақаны солға 100 градусқа бұрады.
- print() арқылы бұрылыс алдындағы және кейінгі бағыттарды көрсетеді.

2.6 СУРЕТ -TURTLECОЛ ЖАҚҚА ЖЫЛЖЫТУ КӨРІНІСІ

Экран бастапқыда төрт квадрантқа бөлінеді, және тасбақа бағдарлама басталғанда координаталық жазықтықтағы нөлдік нүкте - Home позициясында, яғни (0, 0) нүктесінде орналасады. goto(x, y), немесе баламалы түрде turtle.setpos(x, y) және turtle.setposition(x, y) әдістері тасбақаны экрандағы белгілі бір орынға жылжытуға мүмкіндік береді. Бұл функцияларға екі аргумент - x және y координаталары беріледі. Төменде (2.7-суретте көрсетілгендей) тасбақаны қажетті орынға көшіру мысалы қарастырылған.

```
    •    import turtle  # Turtle кітапханасын импорттау
```

- # Тасбақа объектісін жасау t = turtle.Turtle()
- # Тасбақаны экрандағы (100, 80) координаталарына жылжыту t.goto(100, 80)
- # Графикалық терезені ашық ұстау turtle.mainloop()

2.7 СУРЕТ - TURTLE КООРДИНАТТАРЫН ЖЫЛЖЫТУ

Бұл код тасбақаны экрандағы 100 пиксель бойынша көлденең және 80 пиксель бойынша тік бағытта жылжытады.

Пішін салу Алдыңғы бөлімде тасбақаның қозғалыс негіздері қарастырылды. Енді нақты геометриялық пішіндерді құруға көшейік. Әсіресе, көпбұрыштарды салу маңызды, себебі олар белгілі бұрыштармен қосылған түзу сызықтардан тұрады. Төменде көпбұрыш салуға арналған мысал берілген.

```
Бұл мысал (2.8–суретте көрсетілген кескінге) ұқсас келеді.

t.fd(100)
t.rt(90)
t.fd(100)
t.rt(90)
t.fd(100)
t.rt(90)
t.fd(100)
```

2.8 СУРЕТ - TURTLE КӨПБҰРЫШТЫ СЫЗУ КӨРІНІСІ

Тасбақа көмегімен кез келген пішінді салуға болады: тіктөртбұрыш, үшбұрыш, квадрат және тағы басқа. Алайда, тең емес төртбұрышты салғанда оның әрбір бұрышын және координаталарын нақты есептеу қажет болады. Тіктөртбұрыш салғаннан кейін, оның қабырғаларын өзгертіп немесе көбейтіп, басқа да көпбұрыш түрлерін жасауға болады. Егер шеңбер салу керек болса, оны тіктөртбұрыш салады деп ұзақ уақыт жұмсау тиімсіз болады. Бұл жағдайда Python-ның тасбақа кітапханасында арнайы шеңбер салу функциясы бар, ол бұл тапсырманы оңайлатады. Шеңберді радиус бойынша салады, және оның қай бөлігі салынатыны анықталуы мүмкін. Егер бөлшек көрсетілмесе, онда толық дөңгелек салынады. Төменде (2.9-суреттегі) осы функцияны пайдалану мысалы көрсетілген.

```
import turtle  # Turtle кітапханасын импорттау
```

Тасбақа объектісін құру

```
t = turtle.Turtle()
```

Радиусы 50 пиксель болатын шеңбер салу

```
t.circle(50)
```

Графикалық терезені ашық ұстау

```
turtle.mainloop()
```

2.9 СУРЕТ - TURTLE ШЕҢБЕРДІ СЫЗУ КӨРІНІСІ

Бұл бағдарлама тасбақа арқылы радиусы 50 пиксель болатын толық шеңберді салады. Шеңберді салу үшін бір ғана команда жеткілікті: circle(radius, extent=None, steps=None) - бұл функция экранға шеңбер немесе оның бөлігі ретінде қисық сызық сызады. Үш параметрді қабылдайды:

```
    •    radius (радиус) – шеңбердің радиусы, сандық мән болуы тиіс.
```

- extent (дәреже) - қисықтың қанша бөлігі салынады, градуспен өлшенеді; параметр берілмесе, толық шеңбер салынады.
- steps (қадамдар) - қисықты түзу сызықтардан құрау үшін бөлінетін бөліктер саны, бүтін сан. Сонымен қатар, толық толтырылған нүкте (толтырылған шеңбер) салуға да болады. Бұл әдістің нәтижесі 2.10-суретте көрсетілген.

```
import turtle  # Turtle кітапханасын импорттау
```

Тасбақа объектісін жасау

```
t = turtle.Turtle()
# Диаметрі 50 пиксель болатын нүкте (толтырылған шеңбер) салу
t.dot(50)
```

Графикалық терезені ашық ұстау

```
turtle.mainloop()
```

2.10 СУРЕТ - TURTLE ТОЛТЫРЫЛҒАН ШЕҢБЕР

Түсініктеме: t.dot(50) - бұл команда экранға диаметрі 50 пиксель болатын толтырылған шеңбер, яғни нүкте салады. Бұл нүкте экрандағы тасбақаның ағымдағы позициясында пайда болады. Егер dot() командасына екінші аргумент ретінде түс атауын берсеңіз, нүкте сол түспен салынады. Мысалы: t.dot(50, "red") dot() функциясына берілген сан - салынатын нүктенің диаметрін білдіреді. Осы мәнді өзгерту арқылы нүктенің өлшемін үлкейтуге немесе кішірейтуге болады. Осы уақытқа дейін біз тасбақаның экрандағы қозғалысын және әртүрлі геометриялық пішіндерді қалай сызуға болатынын қарастырдық. Енді алдағы бөлімдерде тасбақаның өзін және оның қоршаған ортасын баптау, яғни оның сыртқы түрін, түсін, жылдамдығын және басқа да сипаттарын өзгерту тәсілдерін үйренеміз.

2.2 Экран фонының түсін өзгерту

Әдетте, Turtle модуліндегі экран ақ түспен ашылады. Алайда, арнайы функция арқылы бұл фонның түсін өзгертуге болады.

```
import turtle  # Turtle кітапханасын импорттау
```

Тасбақа объектісін жасау

```
t = turtle.Turtle()
```

Экран фонын қызыл түске орнату

```
turtle.bgcolor("red")
```

Экранды ашық күйде қалдыру

```
turtle.mainloop()
```

Жоғарыдағы код экран фонын қызыл түске өзгертеді. Сонымен қатар, "blue", "yellow", "green" сияқты стандартты түстерді қолдануға немесе HTML-дегі сияқты алтылық (hex) түстер кодын беруге болады.

Мысалы:

```
turtle.bgcolor("#87CEEB")  # Аспан көк түс
```

Экран фонына сурет орнату Экранның фон түсін өзгерткен секілді, фонға сурет қоюға да болады. Бұл үшін bgpic() функциясы пайдаланылады. Жалпы синтаксис:

```
turtle.bgpic(picname)
```

- picname - сурет файлының атауы немесе жолы. Бұл .gif форматындағы файл болуы керек.
- Егер "none" немесе "nopic" деп берілсе, фондағы сурет өшіріледі.
- Параметр көрсетілмесе, ағымдағы фон суретінің аты қайтарылады.

Мысалы:

```
import turtle
```

Тасбақа объектісін жасау

```
t = turtle.Turtle()
```

Ағымдағы фон суретінің атауын шығару

```
print(turtle.bgpic())
# Фондық суретті орнату (тек .gif файлдары ғана жұмыс істейді!)
turtle.bgpic(r"C:\Users\DEVANSH SHARMA\Downloads\person.gif")
```

Қайтадан суреттің атауын шығару

```
print(turtle.bgpic())
```

Экранды ашық ұстау

```
turtle.mainloop()
```

Turtle тек .gif форматындағы кескін файлдарын қабылдайды. Егер .jpg немесе .png форматын қолданғыңыз келсе, оны алдын ала .gif форматына түрлендіру керек.

Экран өлшемін өзгерту Turtle кітапханасында экран өлшемін өзгерту үшін screensize() функциясы қолданылады. Бұл функция арқылы кескін салу аймағының ені мен биіктігін, сондай-ақ оның фон түсін реттеуге болады.

```
Жалпы жазылу тәсілі (синтаксис):
turtle.screensize(canvwidth=None, canvheight=None, bg=None)
```

-

- Параметрлер сипаттамасы:
- canvwidth - кескін салынатын кенептің ені (пиксельмен), оң сан болуы тиіс.
- canvheight - кенептің биіктігі (пиксельмен), оң санмен беріледі.

- bg - фон түсін анықтайтын жол немесе түстер тізбегі. Бұл параметр берілсе, фон түсі автоматты түрде өзгереді.

```
Мысалы (2.11- сурет негізінде):
import turtle
```

Тасбақа экранының өлшемін өзгерту

```
turtle.screensize(canvwidth=800, canvheight=600, bg="lightblue")
```

Тасбақа объектісін жасау

```
t = turtle.Turtle()
```

Терезені ашық күйде қалдыру

```
turtle.mainloop()
```

Бұл өзгеріс кескін салу аймағына (Canvas), яғни логикалық экран өлшеміне әсер етеді. Егер нақты терезе өлшемін өзгерткіңіз келсе, setup() функциясын қолдануға болады.

```
import turtle  # Turtle кітапханасын импорттау
```

Тасбақа объектісін жасау

```
t = turtle.Turtle()
```

Ағымдағы экран өлшемін шығару

```
print("Бастапқы экран өлшемі:")
print(turtle.screensize())
```

Экран өлшемін 1500x1000 пиксельге өзгерту

```
turtle.screensize(1500, 1000)
```

Жаңа экран өлшемін шығару

```
print("Жаңа экран өлшемі:")
print(turtle.screensize())
```

Графикалық терезені ашық ұстау

```
turtle.mainloop()
```

2.11 СУРЕТ - TURTLE КЕСКІН ӨЛШЕМІН СЫЗУ

Не істейді:

```
Алдымен экран өлшемін шығарады (әдетте 400x300 немесе жүйе бойынша).
```

Кейін экран өлшемі 1500x1000 пиксельге орнатылады. Содан кейін жаңа өлшем тағы да экранға шығарылады.

Ескерту:

```
screensize() тек кенептің логикалық өлшемін өзгертеді, яғни сіздің салатын аймағыңыз.
```

Егер нақты терезенің (window) өлшемін өзгерткіңіз келсе, turtle.setup(width, height) функциясын қолдану қажет.

```
setup() арқылы терезе өлшемін орнату
import turtle  # Turtle кітапханасын импорттау
# Терезе өлшемін орнату: ен = 800 пиксель, биіктік = 600 пиксель
turtle.setup(width=800, height=600)
```

Тасбақа объектісін жасау

```
t = turtle.Turtle()
```

Мысалы, бір шеңбер салып көрейік

```
t.circle(100)
```

Терезені ашық ұстау

```
turtle.mainloop()
```

Негізгі айырмашылық: Функция Не өзгертеді

```
screensize()
Кенептің (Canvas) логикалық өлшемі
setup()
Терезе өлшемі (нақты экран терезесі)
```

Яғни, setup() - сыртқы терезенің өлшемін орнатады, ал screensize() - сызуға болатын ішкі аймақтың логикалық көлемін ұлғайтады.

Қалам мен тасбақаның түсін өзгерту Әдепкіде, жаңа Turtle терезесі ашылғанда, тасбақа да, сызатын сызық та қара түспен болады. Алайда, бұл түстерді өз қалауыңызға қарай өзгертуге болады: Қалам түсі - бұл сызылатын сызықтың немесе контурдың түсі.

```
Тасбақаның өзі – бұл тұтас пішіннің (толтыру) түсі.
```

Егер қажет болса, екі түсін де өзгертуге болады: тасбақаның түсін де, сызықтың түсін де бірдей немесе әртүрлі етіп баптауға болады. Түстерді нақты көріп, өзгертулердің әсерін жақсы байқау үшін, алдымен тасбақаның өлшемін үлкейткен дұрыс. Мысалы:

```
import turtle
```

Тасбақа объектісін жасау

```
t = turtle.Turtle()
```

Тасбақаның мөлшерін үлкейту

```
t.turtlesize(3)
# Қалам түсін (сызық түсін) қызылға, ал тасбақаның толтыру түсін көкке өзгерту
t.color("red", "blue")  # Бірінші аргумент – қалам түсі, екінші – толтыру түсі
```

Бір шеңбер салу

```
t.begin_fill()
t.circle(50)
t.end_fill()
```

Терезені ашық күйде қалдыру

```
turtle.mainloop()
```

Түсініктеме:

```
t.color(pencolor, fillcolor) – екі түсті бірден орнатады.
t.turtlesize() – тасбақаның экрандағы көрінетін өлшемін өзгертеді.
begin_fill() және end_fill() – пішінді толтыру үшін қолданылады.

import turtle  # Turtle кітапханасын импорттау
```

Тасбақа объектісін жасау

```
t = turtle.Turtle()
```

Тасбақаның өлшемін үлкейту: ұзындығы, ені, қалам қалыңдығы

```
t.shapesize(3, 3, 3)
# Тасбақаның ішін (толтыру) көк түспен бояу          t.fillcolor("blue")
# Қаламның (сызықтың) түсін сары түске орнату       t.pencolor("yellow")
# Экранды ашық қалдыру   turtle.mainloop()
```

2.12 СУРЕТ - TURTLE КОНТУР СЫЗУ

Қосымша түсініктеме:

```
t.shapesize(stretch_wid, stretch_len, outline)
```

- Тасбақаның биіктігін, енін және контур қалыңдығын көрсетеді. Мұндағы 3, 3, 3 - әдеттегі өлшемнен 3 есе үлкен. fillcolor() - тасбақаның ішкі түсін орнатады (егер фигураны толтырсаңыз, сол түс қолданылады). pencolor() - сызықтың немесе контурдың түсін белгілейді. Екеуінің де - яғни қаламның түсі мен тасбақаның толтыру түсін бір уақытта өзгерту үшін color() функциясын қолдануға болады.

```
import turtle
```

Тасбақа объектісін жасау

```
t = turtle.Turtle()
```

Тасбақаның өлшемін үлкейту

```
t.shapesize(3, 3, 3)
```

Қалам түсін сары, ал ішкі бояу түсін көк етіп орнату

```
t.color("yellow", "blue")
```

Толтырылған шеңбер салу

```
t.begin_fill()
t.circle(60)
t.end_fill()
# Терезені ашық қалдыру  turtle.mainloop()
```

2.13 СУРЕТ - TURTLE ЕКЕУІНІҢ ТҮСІН ӨЗГЕРТУ

```
    •    t.shapesize(3,3,3) – тасбақаның пішінін үлкейтеді.
    •    t.color("green", "red") – контурын жасылға, толтыру түсін қызылға қояды.
    •    t.forward(100) – тасбақаны 100 пиксель алға жылжытады.
    •    turtle.mainloop() – графикалық терезені ашық ұстайды.
```

Қалам мен толтыру түсін бірге орнату Тасбақа сызатын сызықтың түсі мен ішкі бояу түсін бір функция арқылы бірден өзгертуге болады. Ол үшін color() әдісін қолданамыз. Бұл әдіс екі түрлі параметр қабылдайды:

```
t.color(pencolor, fillcolor)

pencolor – сызықтың (қаламның) түсі;
fillcolor – фигураның ішкі бояуы (толтыру түсі).
```

Жоғарыдағы кодта екі түрлі түс көрсетілген: бірінші - қаламның түсі (контурдың түсі), екіншісі - толтыру түсі. Тасбақа сол екі түсті қолданып, пішінді сызады да, оны ішінен бояйды. Түстер суреттің тартымдылығын арттырады және пішіндерді түрлі-түсті етіп безендіруге мүмкіндік береді. Осындай түрлі түсті фигуралар салу үшін келесі мысалды қарастыруға болады (2.14-сурет).

import turtle # Тасбақа тасбақасын жасау

```
t = turtle.Turtle()
t.shapesize(3,3,3)
t.begin_fill()
t.fd(100)
t.lt(120)
t.fd(100)
t.lt(120)
t.fd(100)
t.end_fill()
turtle.mainloop()
```

2.14 СУРЕТ - TURTLE СУРЕТКЕ ТҮС ҚОСУ

Бұл бағдарламада тасбақа көмегімен толтырылған үшбұрыш салынады.

```
    •    Алдымен, тасбақа пішіні үлкейтіледі (t.shapesize(3,3,3)).
```

- t.begin_fill() пен t.end_fill() аралығындағы командалар фигураның ішін бояуға мүмкіндік береді.
- Тасбақа 100 пиксельге алға жылжып, 120 градусқа солға бұрылады - бұл үшбұрыштың қабырғалары мен бұрыштарын қалыптастырады.
- turtle.mainloop() терезені ашық ұстайды.

БАҒДАРЛАМА ІСКЕ ҚОСЫЛҒАНДА АЛДЫМЕН ҮШБҰРЫШ СЫЗЫЛЫП, СОДАН КЕЙІН КӨРСЕТІЛГЕНДЕЙ ҚАРА ТҮСПЕН ТОЛТЫРЫЛАДЫ. ТОЛТЫРУ ПРОЦЕСІН БАСТАУ ҮШІН ЖАБЫҚ ФИГУРАНЫ САЛУДЫ БІЛДІРЕТІН BEGIN_FILL() ӘДІСІ ҚОЛДАНЫЛАДЫ. ОЛ АЯҚТАЛҒАН СОҢ, END_FILL() ӘДІСІ ШАҚЫРЫЛЫП, ФИГУРАНЫҢ ТОЛТЫРЫЛАТЫНЫН БІЛДІРЕДІ. ОСЫ АРҚЫЛЫ ПІШІН ТҮРЛІ ТҮСТЕРМЕН БОЯЛУҒА ДАЙЫН БОЛАДЫ.

Тасбақаның бастапқы формасы үшбұрыш болып табылады. Дегенмен, тасбақаның пішінін өзгертуге болады, өйткені бұл модульде тасбақа үшін әртүрлі пішіндер жиынтығы бар. Мысал ретінде келесі кодты қарастырайық (2.15-сурет).

import turtle # Creating turtle turtle

```
t = turtle.Turtle()
t.shape("turtle")
```

Change to arrow

```
t.shape("arrow")
```

Chnage to circle

```
t.shape("circle")
turtle.mainloop()
```

2.15 СУРЕТ - TURTLE ФОРМАСЫН ӨЗГЕРТУ

Бұл кодта тасбақаның пішіні бірнеше рет өзгереді:

```
    •    Алдымен, t.shape("turtle") арқылы тасбақа классикалық тасбақа формасына айналады.
    •    Кейін t.shape("arrow") арқылы пішіні көрсеткіге ауысады.
    •    Соңында t.shape("circle") арқылы тасбақа дөңгелек формаға өзгереді.
```

Экранда тек соңғы өзгеріс - дөңгелек пішін көрінеді, өйткені әр жаңа shape() шақыруы бұрынғысын ауыстырады. Егер әр пішінді бөлек көру керек болса, олардың арасында кідіріс қосу қажет. Тасбақаның пішінін қажетті талапқа қарай өзгертуге болады. Ол квадрат, үшбұрыш, классикалық тасбақа, жебе немесе шеңбер сияқты әртүрлі формада болуы мүмкін. Классикалық пішін - бұл тасбақаның стандартты көрінісі. Сонымен қатар, тасбақаның қозғалыс жылдамдығын да басқаруға болады. Әдетте, тасбақа экранда орташа жылдамдықпен қозғалады, бірақ қажет болса, оны тездетіп немесе баяулатуға мүмкіндік бар. Төменде тасбақаның жылдамдығын өзгерту тәсілі көрсетілген (2.16-сурет).

import turtle # Тасбақа жасау

```
t = turtle.Turtle()
t.speed(3)
t.forward(100)
```

```
t.speed(7)
t.forward(100)
turtle.mainloop()
```

2.16 СУРЕТ - TURTLE ЖЫЛДАМДЫҒЫН ӨЗГЕРТУ

- t.speed(3) - тасбақаның қозғалыс жылдамдығын 3 деңгейіне қояды (төмен немесе орташа жылдамдық).
- t.forward(100) - тасбақа 100 пиксель алға жылжиды.
- t.speed(7) - жылдамдықты 7 деңгейіне көтереді (жылдамырақ).
- t.forward(100) - тағы 100 пиксель алға қозғалады. speed() әдісі 0-ден 10-ға дейін мәндерді қабылдайды, мұндағы 0 - ең жылдам қозғалыс (секундтық көрсетілімсіз), ал 1 - ең баяу. Тасбақа жылдамдығы 0-ден 10-ға дейін бүтін сандар арасында өзгеруі мүмкін. Егер speed() функциясы параметрсіз шақырылса, ол тасбақаның ағымдағы қозғалыс жылдамдығын көрсетеді. Төмендегі кестеде жылдамдық мәндері мен олардың мағыналары көрсетілген (2.1 -кесте).

2.1 КЕСТЕ - ЖЫЛДАМДЫҚ ЖОЛДАРЫ

0	Fastest	Ең жылдам	10	Fast	Жылдам	6	Normal
Қалыпты	3	Slow	Баяу	1	Slowest	Ең баяу	

Ескерту: Егер жылдамдық 0 деп орнатылса, тасбақа қозғалысы анимациясыз, дереу жүзеге асады. Бір жолда баптау. Тасбақа параметрлерін бірнеше рет өзгертудің орнына, оларды бір жолда бірден орнатуға болады. Мысалы, төмендегідей сипаттамаларды тағайындауға болады:

- Қалам түсі - қызыл
- Толтыру түсі - қызғылт сары
- Қаламның қалыңдығы - 10
- Қозғалыс жылдамдығы - 7
- Экранның фоны - көк Бұл параметрлерді қолданатын мысалды қарастырайық (2.17-сурет).

import turtle # Тасбақа жасау

```
t = turtle.Turtle()
t.pencolor("red")
t.fillcolor("orange")
t.pensize(10)
t.speed(7)
t.begin_fill()
t.circle(75)
turtle.bgcolor("blue")
t.end_fill()
turtle.mainloop()
```

2.17 СУРЕТ - TURTLE ФОНЫН ӨЗГЕРТУ

Бұл код тасбақаның қалам түсін қызылға, толтыру түсін қызғылт сарыға, қаламның қалыңдығын 10-ға, жылдамдығын 7-ге орнатады және көк фонды экранда шеңбер салады. Тасбақаны (Turtle) көшіру Кейде күрделі дизайн жасау үшін бірнеше тасбақаларды пайдалану керек болады. Мұндай жағдайда ағымдағы тасбақаның көшірмесін жасап, экранда бірнеше тасбақаны бір уақытта басқаруға болады. Төменде осындай мысал көрсетілген (2.18-сурет).

import turtle # Тасбақа жасау

```
t = turtle.Turtle()
c = t.clone()
t.color("blue")
c.color("red")
t.circle(20)
c.circle(30)
for i in range(40, 100, 10):
```

```
c.circle(i)
turtle.mainloop()
```

2.18 СУРЕТ - TURTLE ШЕҢБЕР СЫЗУ

Кодта for цикліндегі код дұрыс индентацияланбаған және циклдің ішінде қай тасбақамен жұмыс істейтіні анық емес. Түзетілген нұсқасы төменде көрсетілген:

```
import turtle
```

Тасбақа жасау

```
t = turtle.Turtle()
c = t.clone()
t.color("blue")
c.color("red")
t.circle(20)
c.circle(30)
for i in range(40, 100, 10):
c.circle(i)  # цикл ішінде c тасбақасын қолдану
turtle.mainloop()
```

Бұл кодта t тасбақасы көк түспен 20 радиусты шеңбер салады, ал c тасбақасы қызыл түспен 30 радиусты және цикл арқылы 40-тан 90-ға дейінгі радиустардағы шеңберлерді салады. Жоғарыда көрсетілген кодта тасбақа c деген жаңа айнымалыға көшіріледі. Алдымен t тасбақасы көк түсті шеңбер сызады, одан кейін c тасбақасы for циклі арқылы берілген шарттарға сәйкес әртүрлі радиустары бар сыртқы шеңберлерді салады.

2.3 Циклдар мен шартты операторлар

Turtle кітапханасының негізгі және күрделі ұғымдарын меңгергеннен кейін, келесі маңызды кезең - Python тіліндегі циклдар мен шартты операторларды пайдаланып оларды зерттеу. Циклдар (Loops) - белгілі бір шарт орындалғанша кодтың белгілі бір бөлігін қайталап орындауға мүмкіндік береді. Шартты операторлар (Conditional Statements) - бағдарламаның орындалуын белгілі бір шарттарға байланысты өзгерту үшін қолданылады. Шегініс (Indentation) - кодтың құрылымын анықтайтын және циклдар мен шартты операторлар кезінде өте маңызды рөл атқаратын бос орындар жиынтығы. Бір деңгейдегі операторлар бір код блогына жатады. Тақырыпты жақсы түсіну үшін төмендегі мысалдарға назар аударайық.

For циклі Циклдер - бірнеше рет қайталанатын әрекеттерді орындайды. Алдыңғы мысалдарда бірнеше жолдарды қайталап жазу қажет болғанда for циклін қолдану тиімді болады. Мысалы, квадрат салатын бағдарлама құрайық.

```
    •    t.fd(100)
    •    t.rt(90)
    •    t.fd(100)
    •    t.rt(90)
    •    t.fd(100)
    •    t.rt(90)
    •    t.fd(100)
    •    t.rt(90)
for циклінің көмегімен оны қысқарта аламыз. Төмендегі кодты іске қосыңыз (2.19 –сурет).
```

Import turtle # Тасбақажасау

```
t = turtle.Turtle()
for i in range(4):
t.fd(100)
t.rt(90)
turtle.mainloop()
```

2.19 СУРЕТ -FOR ЦИКЛІН ҚОСУ

Мына кодыңызда шегініс (indentation) дұрыс қойылмаған, сондықтан Python қате береді. Дұрыс нұсқасы төмендегідей болуы керек:

```
import turtle
```

Тасбақа жасау

```
t = turtle.Turtle()
for i in range(4):
    t.fd(100)
    t.rt(90)
turtle.mainloop()
```

Түсініктеме: Берілген кодта for циклы i айнымалысының мәні 0-ден 3-ке дейін өзгергенше төрт рет қайталанады. Әр қайталануда тасбақа 100 бірлік алға жүріп, содан кейін 90 градусқа оңға бұрылады.

- Бірінші айналымда, i мәні 0, тасбақа 100 бірлік алға жүріп, оңға 90 градус бұрылады.
- Екінші айналымда, i 1, тасбақа қайтадан 100 бірлік алға жүріп, тағы 90 градусқа оңға бұрылады.
- Үшінші айналымда, i 2, тасбақа тағы 100 бірлік алға жылжып, оңға бұрылады.
- Төртінші айналымда, i 3, тасбақа соңғы 100 бірлік алға жүріп, тағы 90 градусқа бұрылады. Барлық төрт айналым аяқталған соң цикл тоқтайды және тасбақа бастапқы орнына оралып, төртбұрыш пішінін сызады. while циклы - бұл белгілі бір шарт шын болып тұрғанша кодтың белгілі бір бөлігін қайталап орындау үшін пайдаланылады. Егер шарт жалғанға айналса, цикл тоқтайды. Төменде осының мысалы келтірілген (2.20-сурет).

import turtle # Тасбақа жасау

```
t = turtle.Turtle()
n=10
while n <= 60:
t.circle(n)
n = n+10
turtle.mainloop()
```

2.20 СУРЕТ-WHILE ЦИКЛІН ҚОСУ

Мына кодта тасбақа n мәні 10-нан бастап 60-қа дейінгі аралықта, әрбір қадам сайын 10-ға көбейіп, сол радиустың шеңберлерін сызады. while циклы n ≤ 60 шарты орындалғанша жұмыс істейді, әр айналымда тасбақа сол радиустың шеңберін салады және n мәнін 10-ға арттырады. Осылайша, түрлі радиустардағы шеңберлер қатарынан пайда болады. Нәтижеден көріп тұрғанымыздай, while циклы арқылы бірнеше шеңбер салынады. Әрбір жаңа цикл орындалғанда, шеңбердің радиусы алдыңғысына қарағанда ұлғаяды. Мұнда n айнымалысы есептегіш ретінде қолданылады, және әр қайталау сайын оның мәні өседі. Циклдің жұмыс істеу тәртібін қарастырайық: Бірінші айналымда n бастапқы мәні 10-ға тең, яғни тасбақа радиусы 10 болатын шеңбер салады. Екінші айналымда n 10-ға ұлғаяды да, мәні 20 болады; тасбақа радиусы 20 бірлік шеңбер салады. Үшінші айналымда n тағы 10-ға өсіп, 30-ға тең болады; тасбақа радиусы 30 бірлік шеңберді сызады. Төртінші айналымда n 40-қа жетеді және тасбақа радиусы 40 бірлік шеңберді салады. Осылайша, әр қайталауда радиус біртіндеп ұлғая береді. Шартты оператор белгілі бір шарттың орындалуын тексереді. Егер шарт дұрыс болса, онда осы шартқа сәйкес келетін код бөлігі іске қосылады. Төменде шартты операторлардың мысалы берілген (2.21-сурет).

import turtle # Creating turtle

```
t = turtle.Turtle()
n = 40
if n<=50:
t.circle(n)
else:
t.forward(n)
```

```
t.backward(n-10)
turtle.mainloop()
```

2.21 СУРЕТ -ШАРТТЫ ОПЕРАТОРЫН ҚОСУ

Мынау кодың дұрыс жұмыс істеуі үшін шегіністерді реттеу керек. Төменде дұрыс форматталған нұсқасын ұсынамын:

```
import turtle
```

Тасбақа жасау

```
t = turtle.Turtle()
n = 40
if n <= 50:
    t.circle(n)
else:
    t.forward(n)
    t.backward(n - 10)
turtle.mainloop()
```

Бұл мысалда n мәні 50-ден кіші немесе тең болғанда тасбақа радиусы n болатын шеңбер салады. Ал егер n 50-ден үлкен болса, онда тасбақа алдыға n бірлік жылжып, содан кейін артқа n-10 бірлік кері қозғалады. Түсініктеме: Жоғарыдағы бағдарламада нәтиже қолданушы енгізген мәнге байланысты анықталады. Егер енгізілген сан 50-ден кіші болса, тасбақа шеңбер сызады, ал егер одан үлкен болса - басқа әрекет орындайды. Мұнда мысал ретінде 40 мәні беріліп, шеңбер салу іске асады. Енді тасбақа кітапханасының көмегімен әдемі әрі күрделі сызбалар жасауға көшейік.

2.4 Python Turtle кітапханасымен дизайнын жасау

Python тасбақа кітапханасының негізгі және озық мүмкіндіктерін пайдаланып, түрлі қызықты жобаларды жасауға болады. Бұл кітапхана ойындар, ерекше фигуралар және тағы басқа көптеген визуалды элементтерді жасауға мүмкіндік береді. Төменде тасбақа көмегімен жасалған бірнеше дизайн үлгісі қарастырылған. Шеңберлерден тұратын спиральды сурет

```
import turtle
```

Тасбақа дайындау

```
t = turtle.Turtle()
turtle.bgcolor("black")
turtle.pensize(2)
turtle.speed(0)
```

while True:

```
    for i in range(6):
    for color in ["red", "blue", "magenta", "green", "yellow", "white"]:
            turtle.color(color)
            turtle.circle(100)
            turtle.left(10)
turtle.hideturtle()
turtle.mainloop()
```

Осы бағдарлама нәтижесінде (2.22-сурет) түрлі-түсті шеңберлерден құралған динамикалық спиральды сурет пайда болады.

2.22 СУРЕТ - ШЕҢБЕРЛІ СПИРАЛЬДЫ ГРАФИК КӨРІНІСІ

Вибрациялы шеңбер сызу

```
import turtle
```

Тасбақа мен экранды дайындау

```
t = turtle.Turtle()
screen = turtle.Screen()
screen.bgcolor("black")

t.pencolor("red")
t.speed(0)

a = 0
b = 0

t.penup()
t.goto(0, 200)
t.pendown()
```

while True:

```
    t.forward(a)
    t.right(b)
```

a += 3 b += 1

```
    if b == 210:
```

break

```
t.hideturtle()
turtle.done()
```

Бұл бағдарламада тасбақа экранда қызыл түсті сызықпен қозғалып, қозғалыс бұрышын және қашықтықты біртіндеп арттыра отырып, вибрациялы эффектпен шеңбер тәрізді пішін салады. Цикл 210 рет орындалып, содан кейін тоқтайды.

2.23 СУРЕТ - ВИБРАЦИЯЛЫ ШЕҢБЕР СЫЗУ

Жүрекше сызу

```
import turtle
```

Тасбақа мен экранды дайындау

```
t = turtle.Turtle()
screen = turtle.Screen()
screen.bgcolor("black")
t.pensize(2)
```

Жұмсақ қисық сызық салу функциясы

```
def curve():
    for _ in range(200):
        t.right(1)
        t.forward(1)
t.speed(3)
```

```
t.color("red", "pink")
t.begin_fill()
t.left(140)
t.forward(111.65)
curve()
t.left(120)
curve()
t.forward(111.65)
t.end_fill()
t.hideturtle()
turtle.mainloop()
```

Бұл бағдарлама тасбақа көмегімен қызыл контурлы, қызғылт толтырмасы бар жүрек пішінін салады. curve() функциясы жүректің қисық бөліктерін жұмсақ әрі тегіс ету үшін қолданылады.

2.24 СУРЕТ - ЖҮРЕКШЕ СЫЗУ

Жоғарыдағы бағдарламада экранға қисық сызық салу үшін арнайы curve функциясы жасалған. Ол жүректің тегіс, әдемі пішінін қалыптастыруға көмектеседі. Жүрек пішіні толтырылған және автоматты түрде түсімен боялады. Бұл кодты өз компьютеріңізде көшіріп, іске қосып, оған қосымша дизайн элементтерін қосуға болады. Төменде бірнеше қосымша мысал келтірілген:

python КопироватьРедактировать

```
import turtle
polygon_ = turtle.Turtle()
for i in range(6):
    polygon_.forward(100)
    polygon_.right(300)
turtle.done()
```

Бұл мысалда тасбақа алтыбұрыш салу үшін циклды қолданады, әрбір қадам сайын алға 100 пиксель қозғалып, 300 градусқа оңға бұрылады. Одан әрі түрлі көпбұрыштар жасауға болады.

```
import turtle
star = turtle.Turtle()
num_of_sides = 5
length_of_side = 50
```

each_angle = 720.0 / num_of_sides # 720 градус, себебі жұлдыз саламыз

```
for i in range(num_of_sides):
    star.forward(length_of_side)
    star.right(each_angle)
turtle.done()
```

Түсініктеме:

```
num_of_sides – жұлдыздың бұрыш саны (5).
```

length_of_side - әрбір қырының ұзындығы. each_angle - әр бұрыштың бұрылу градусы, жұлдыз үшін 720 градус бөлінген 5-ке.

Циклда тасбақа 5 рет жүріп, әр қадамда берілген қашықтыққа жүріп, қажетті бұрышқа бұрылады. Егер жұлдыздың нақты дұрыс пішінін салғыңыз келсе, бұрышты 144 градусқа қою керек (жұлдызды салуда 180 - 180/5 = 144). Мысалы:

```
import turtle
star = turtle.Turtle()
```

```
num_of_sides = 5
length_of_side = 50
each_angle = 144  # жұлдыз салу бұрышы
for i in range(num_of_sides):
    star.forward(length_of_side)
    star.right(each_angle)
turtle.done()
from turtle import *
colors = ['orange', 'red', 'pink', 'yellow', 'blue', 'green']
for x in range(360):
    pencolor(colors[x % 6])  # Түсті 6 түстен қайталаймыз
    width(x / 50 + 1)   # Қаламның қалыңдығын өзгертеміз (үлкейту үшін 50-ге бөлдік)
    forward(x)
left(20)
done()
```

Негізгі түзетулер:

```
from turtle import * – дұрыс синтаксис.
```

Тырнақшаларды ' немесе " деп ауыстырдым, өйткені оригиналдағы ‘’ қате таңба. width(x/5+1)-ді width(x / 50 + 1) деп өзгерттім, себебі x/5 өте үлкен және қалам тым қалың болады. Қажет болса, бұл мәнді өзгертуге болады. Әрбір for циклының ішіндегі командаларда шегініс керек. pencolor, forward, left, width - функциялар ретінде шақырылған.

```
Соңында done() функциясын шақырамыз, бұл терезені ашық ұстайды.
```

Осы кодты іске қоссаңыз, көп түрлі-түсті спираль пайда болады.

```
from turtle import *
penup()  # Қаламды көтеру, сызбай жылжу үшін
for a in range(40, -1, -1):
    stamp()       # Қазіргі тасбақа пішінін экранға басып шығару
    left(a)       # Солға бұрылу градусымен a градусқа
    forward(20)   # 20 бірлік алға жылжу
done()  # Терезені ашық ұстау
```

Негізгі түзетулер:

```
from turtle import * – толық импорт керек.
penup() пен басқа командалардың алдында міндетті түрде шегініс (индентация) болуы керек.
stamp() – тасбақа пішінін қалдырған із.
Соңында done() функциясын шақырып, графиктік терезенің жабылмауын қамтамасыз етеміз.
```

Осы код тасбақа белгісін әр бұрылыс сайын басып, алға жылжиды. Түсінікті және қарапайым командаларды пайдалана отырып, кез келген адам тасбақаның қозғалысын басқаратын, сурет салуға арналған терезе немесе кенеп жасай алады. Python бағдарламалау тіліндегі бұл командалар арқылы әртүрлі дизайндар мен үлгілерді оңай құруға болады. Осындай тәжірибе нәтижесінде әдемі және қызықты сызбалар пайда болады. Python Turtle кітапханасы, әсіресе балалар мен бастаушылар үшін, программалауды және Python тілін үйренудің тамаша, жеңіл әрі қызықты құралы болып табылады.

2.5 Turtle жұлдызы

Turtle кітапханасы арқылы күрделі пішіндерді қарапайым қимылдарды бірнеше рет қайталау арқылы оңай сызуға болады. Мұндай қайталанатын әрекеттердің көмегімен жұлдыз тәрізді күрделі фигураларды құруға болады (мысалы, 2.25 және 2.26-суреттердегі сияқты).

2.25 СУРЕТ - КҮНДІ СЫЗУ БАҒДАРЛАМАСЫ

2.26 СУРЕТ - КҮННІҢ СУРЕТІ

Графикалық мүмкіндіктерді пайдалана отырып, алғашқы Turtle бағдарламасын жасайық. Жаңа IDLE терезесінде төмендегі кодты жазып, оны KvadratSpiral1.py атауымен сақтауға болады (2.27-суретте көрсетілгендей).

2.27 СУРЕТ -KVADRATSPIRAL1.PY БАҒДАРЛАМА КӨРІНІСІ

Бағдарламаны орындағаннан кейін, экранда төмендегідей бейне пайда болады (2.28-суретте көрсетілгендей).

2.28 СУРЕТ - КВАДРАТ СПИРАЛЬДЫҢ СЫЗЫЛҒАН КӨРІНІСІ.

БАҒДАРЛАМА ҚАЛАЙ ЖҰМЫС ІСТЕЙДІ?

Бағдарламаның жұмыс істеу логикасын түсіну үшін оны жол-жолмен талдап көрейік. KvadratSpiral1.py файлының бірінші жолында «#» таңбасымен басталатын түсініктеме жазылған. Python тілінде мұндай таңбамен басталған жолдар компьютер тарапынан орындалмайды - олар тек адамға арналған ескертпе немесе түсініктеме ретінде қолданылады. Бұл мысалда түсініктеме бағдарлама атауын және оның не істейтінін қысқаша сипаттайды.

```
Келесі жолда turtle модулі импортталады. Бұл кітапхана Python тілінде графикалық сурет салуға арналған дайын құралдарды ұсынады. Python–ның ең жақсы мүмкіндіктерінің бірі – дайын жазылған модульдерді оңай қосып, оларды қайтадан қолдануға болатыны. Яғни, егер сіз пайдалы бағдарлама жасаған болсаңыз, оны басқа адамдар пайдалана алады немесе сіз оны өзге жобаларыңызда қайта қолдана аласыз. Turtle модулі алғаш рет Logo бағдарламалау тілінде 1960 жылдары енгізілген идеяларға негізделген. Ал Python–да оны пайдалану үшін жай ғана import turtle деп жазу жеткілікті – бұл жол бағдарламаның осы модульдегі барлық мүмкіндіктерге қол жеткізе алатынын білдіреді.
```

Үшінші жолда t = turtle.Pen() деген команданы көреміз. Мұнда біз t деген қысқа атауды тасбақа объектісіне беріп отырмыз. Бұл бізге ұзын turtle.Pen() орнына тек t арқылы тасбақаға бұйрық беруге мүмкіндік береді. Мысалы, t.forward(100) командасы тасбақаны алға 100 бірлікке жылжытады. Ал төртінші жол - циклдің басталуы. Бұл цикл белгілі бір әрекеттерді бірнеше рет қайталауға мүмкіндік береді. Мұнда range(100) дегеніміз - 0-ден бастап 99-ға дейінгі сандар. Python тілінде циклдер көбіне 0-ден басталады. Әр итерация сайын x айнымалысы 1-ге артып отырады: бастапқыда 0, содан кейін 1, 2, 3... болып жалғасады. Осылайша цикл 100 рет орындалады.

Айнымалы және цикл жұмысын түсіну Бұл жерде x айнымалысы ретінде әрекет етеді. Бұған дейінгі бөлімдерде айнымалылар туралы айтылған. Келесі екі жол циклдің денесіне жатады, өйткені олар шегіністен жазылған. Яғни, бұл жолдар цикл ішінде орындалады және циклдің әр айналымында қайта іске қосылады. Нақтырақ айтқанда, x 0-ден 99-ға дейін мән қабылдай отырып, осы кодтар 100 рет қайталанады. Python бағдарламасы циклдің ішіндегі нұсқауларды қалай орындайтынын қарастырайық. t.forward(x) командасы тасбақаны x бірлікке алға жылжытады. Циклдің алғашқы айналымында x = 0, сондықтан тасбақа қозғалмайды. Одан кейінгі t.left(90) командасы оны 90 градусқа солға бұрады. Осыдан соң цикл басына қайта оралып, x айнымалысының мәні 1-ге өседі. Себебі x әлі 0 мен 99 аралығында болғандықтан, цикл жалғасады. Енді тасбақа 1 бірлік алға жылжып, 90 градусқа бұрылады. Бұл әрекеттер әр айналым сайын қайталанып отырады. Соңғы айналымда, x = 99 болған кезде, тасбақа ең ұзын сызықты сызады, бұл спиральдың шеткі бөлігіне сәйкес келеді. Нәтижесінде, x айнымалысының мәні біртіндеп 0-ден 99-ға дейін өзгеріп отырған кезде, әр қадам сайын сызық ұзарып, бұрылу бағыты тұрақты болып қалады. Осылайша, шаршы пішіндегі спираль пайда болады.

2.2 КЕСТЕ - FOR ЦИКЛІНІҢКЕЗЕҢ-КЕЗЕҢМЕН ВИЗУАЛИЗАЦИЯЛАУЫ

for x in range (100):	t.forward (x)	t.left(90)

```
0–ден 4–ке дейінгі цикл: (x=4) алғашқы төрт сызықты сызады
```

5-тен 8-ге дейінгі цикл: тағы төрт сызықты сызады, экранда квадрат пайда болады

```
9–дан 12–ге дейінгі цикл: квадраттық спираль 12 сызыққа дейін көбейеді (үш квадрат)
```

Компьютер экранында көрсетілетін нүктелер - пиксельдер - өте ұсақ болғандықтан, оларды көзбен анық көру қиын. Бірақ x айнымалысының мәні 100-ге дейін біртіндеп ұлғайған сайын, тасбақа экранда көбірек және ұзын сызықтар жүргізе бастайды. Яғни, x неғұрлым үлкен болса, t.forward(x) арқылы тасбақа соғұрлым ұзын қадам жасайды. Экрандағы тасбақа көрсеткісі алға жылжып, 90 градусқа бұрылып отырады. Әр бұрылған сайын бұрынғысынан ұзағырақ сызық сызады. Бұл әрекет бірнеше рет қайталанғанда, әр жолдың ұзындығы алдыңғысынан артып отырады.

БАҒДАРЛАМА ОРЫНДАЛҒАН СОҢ, ЭКРАНДА ТӨРТБҰРЫШ ПІШІНДІ ҚҰРЫЛЫМ ПАЙДА БОЛАДЫ. МҰНДАЙ КӨРІНІС 90 ГРАДУСҚА ТӨРТ РЕТ БҰРЫЛУ АРҚЫЛЫ БАСТАПҚЫ ОРНЫНА ҚАЙТА ОРАЛУҒА МҮМКІНДІК БЕРЕДІ. ДЕГЕНМЕН, БҰЛ ЖАҒДАЙДА БІЗ НАҚТЫ ШАРШЫ ЕМЕС, СПИРАЛЬҒА ҰҚСАС СУРЕТ АЛАМЫЗ. СЕБЕБІ ӘР АЙНАЛЫМДА ТАСБАҚА АЛҒА БҰРЫНҒЫДАН ҰЗАҚ ҚАДАМ ЖАСАП ОТЫРАДЫ. МЫСАЛЫ, АЛҒАШЫНДА БІР ҚАДАМ АЛҒА ЖЫЛЖИДЫ (X = 1), КЕЛЕСІДЕ ЕКІ ҚАДАМ, СОДАН КЕЙІН ҮШ, ТӨРТ, ОСЫЛАЙША 99-ҒА ДЕЙІН.

Жекелеген пиксельдерді көрмесек те, тасбақа жүргізген сызықтардың біртіндеп ұзарғаны айқын байқалады. Барлық бұрылулар 90 градуста жасалатын болғандықтан, жалпы сурет төртбұрыш пішінді болып шығады. Енді осы бағдарламаның бір бөлігін өзгертсек, нәтижесі қалай өзгереді екенін байқап көрейік. Мысалы, соңғы жолдағы бұрылу бұрышын t.left(91) деп өзгертсек, спиральдың бағыты мен формасы өзгереді. Бұл өзгерісті жаңа файл ретінде KvadratSpiral2.py атауымен сақтап көруге болады.

2.29 СУРЕТ- KVADRATSPIRAL2.PY БАҒДАРЛАМАСЫ

90 градусқа бұрылу нақты квадрат пішінін алуға мүмкіндік береді. Ал егер бұрыш сәл ғана - мысалы, 91 градусқа - арттырылса, онда бұл пішінді аздап өзгеріске ұшыратады. Әрбір келесі бұрылыста осы кішкентай ауытқу жинақталып, бастапқы шаршы формадан біртіндеп алыстайды. Бағдарлама жұмысын жалғастыра берген сайын бұл ауытқулар анық байқала бастайды, ал соңында пайда болатын сурет кәдімгі шаршы емес, бұралған спираль тәрізді көрініске айналады. Мұндай форма бұрандалы баспалдаққа немесе шексіз айналып жатқан өрнекке ұқсайды. Бұл құбылысты 2.30-суреттен анық байқауға болады.

2.30 СУРЕТ - БҰРАНДАЛЫ БАСПАЛДАҚ ТӘРІЗДІ КВАДРАТ КӨРІНІСІ

Соңғы жолдағы бұрыш мәнін 91, 46, 61 немесе 121 градус сияқты әртүрлі мәндерге өзгертіп көруге болады. Бірақ өзгерістерді сақтауды ұмытпаңыз. Бағдарлама іске қосылғаннан кейін, өзгертілген параметрдің нәтижесінде пайда болатын жаңа өрнектерді бақылауға болады. Геометрия негіздерін жақсы білетін үлкендер таңдалған бұрыштың қандай пішін беретінін алдын ала болжай алуы мүмкін. Ал жас қолданушылар үшін бұл - тәжірибе арқылы үйренудің тамаша тәсілі: олар бұрышты өзгертіп, түрлі суреттердің қалай пайда болатынын көріп, қызығушылықпен зерттей алады. Мұндай тапсырма болашақта геометрия сабағында таныс ой тудыруы әбден мүмкін.

Turtle арқылы дөңгелек фигураларды салу. Turtle графикалық модулі тек түзу сызықтармен шектелмейді - ол арқылы әртүрлі күрделі пішіндер де салуға болады. Енді Python-ның Turtle кітапханасының тағы бір мүмкіндігін қарастырайық. Алдыңғы кодта t.forward(x) жолын қолдану арқылы тасбақа тік сызық салып отырғанын байқадық. Бұл команда тасбақаны x пиксельге ілгерілетіп, біртіндеп бұрылу арқылы кескін жасайды. Енді осы жолды өзгеше пішін - шеңбер салу үшін өзгертіп көрейік. t.forward(x) жолын t.circle(x) деп ауыстырсақ, тасбақа экранда радиусы x болатын шеңбер салады. Бұл команда түзу сызық салуға арналған нұсқа сияқты оңай орындалады. Осы өзгертілген кодты CircleSpiral1.py атауымен сақтап, нәтижесін бақылап көріңіз (2.31-сурет).

2.31 СУРЕТ - CIRLESPIRAL1.PY БАҒДАРЛАМАСЫ

t.forward(x) командасын тек t.circle(x)-ке ауыстыру арқылы әлдеқайда күрделі және қызықты пішін алуға болады (бұл 37-суретте көрінеді). t.circle(x) функциясы тасбақаға дәл тұрған жерінен радиусы x болатын дөңгелек салуды тапсырады. Шыққан фигура шаршы спиральмен ұқсас тұстарға ие екенін байқауға болады: салынған сурет төрт дөңгелек спиральдан тұрады. Бұл - бұған дейінгі квадрат спиральдағы төрт қабырғаға сәйкес келеді. Мұның себебі - тасбақа әр бұрылған сайын t.left(91) командасы арқылы 90 градустан сәл көбірек бұрылып отырады. Ал бұл төрт бұрылыс - 4×90=360 градустық толық айналымға ұқсас. Сонымен қатар, бір маңызды айырмашылықты да атап өткен жөн: бұл шеңберлі спираль квадраттық спиральмен салыстырғанда екі есе үлкен болып көрінеді. Бұның себебі - t.circle(x) командасы шеңбердің радиусын x мәні ретінде қабылдайды, яғни бұл шеңбердің ені немесе диаметрі x-тен екі есе көп болады.

Мысалы, егер x = 1 болса, диаметр 2 пиксель болады, ал x = 99 болса - диаметрі 198 пиксельге жетеді. Бұл - ең үлкен квадраттың қабырғасынан шамамен екі есе ұзын деген сөз. Сондықтан экрандағы бұл дөңгелек спираль бұрынғы шаршы спиральмен салыстырғанда екі есе кең аумақты қамтиды (қосымша ақпарат 2.32-суретте көрсетілген).

2.32 СУРЕТ - ДӨҢГЕЛЕК СПИРАЛЬ БЕЙНЕСІ

Түстер қосу Спиральдар әдемі әрі қызықты, бірақ оларды түрлі түстермен безендіру олардың көрінісін одан сайын жақсартады. Енді квадрат спираль кодына оралып, t = turtle.Pen() жолынан кейін тасбақаның қалам түсін қызыл етіп орнататын команданы қосайық (2.33-суретте көрсетілгендей).

2.33 СУРЕТ - КВАДРАТ СПИРАЛЬҒА ТҮСТІ ҚОСУ

Бағдарламаны іске қосқаннан кейін экранда жарқын қызыл түспен жасалған квадрат спираль көрінісі пайда болады (2.34-сурет).

2.34 СУРЕТ - ҚЫЗЫЛ ТҮСТІ КВАДРАТ СПИРАЛЬ

"Red" (қызыл) түсін blue (көк) немесе green (жасыл) сияқты басқа түстермен ауыстырып, бағдарламаны қайта іске қосып көріңіз. Turtle кітапханасы көптеген түрлі түстерді қолдағандықтан, әртүрлі түсті нұсқаларын оңай жасауға болады. Толық бір түсті спираль жақсы көрінеді, бірақ егер спиральдың әр қырын жеке түспен бояғыңыз келсе ше? Мұны іске асыру үшін бағдарламаға қосымша өзгерістер енгізу қажет болады. Төрт түсті спираль жасау алгоритмі келесідей болуы мүмкін:

- Turtle модулін импорттап, тасбақа объектісін дайындаңыз.
- Қолданғыңыз келетін түстер тізімін анықтаңыз.
- Спиральдың 100 қадамын салу үшін цикл құрыңыз.
- Әр қадамда спиральдың әр жағы үшін тізімнен сәйкес түсті таңдаңыз.
- Тасбақаны алға жылжытатын команданы орындаңыз.
- Әр қадамнан кейін тасбақаны солға бұрып, келесі сызықты салуға дайындалыңыз. Бастау үшін, бірнеше түстерді қамтитын colors деп аталатын тізімді жасаймыз және оған төрт түрлі түсті орналастырамыз:

Төрт түрлі түстер тізімі әрбір спираль қырына жеке-түсті бояу береді. Мұнда түстердің аттары тырнақша ішінде жазылған, себебі олар мәтін (жол) түрінде беріледі. Бұл түстердің аттары pencolor функциясында қолданылған кезде, Python оларды түс атаулары ретінде дұрыс таниды. Түстердің сақталуы үшін біз colors деген айнымалыға тізім ретінде орналастырамыз. Кейін бағдарламада қажетті түс ретінде осы тізімнен таңдап алып, қаламның түсін орнатамыз. Айнымалылар өзгермелі мәндерді сақтайтындықтан, біз қалаған түсті осы айнымалыдан ала аламыз. Сурет салу үшін жазылған цикл ішінде, әр қадамда қаламның түсін ауыстыру үшін t.pencolor() функциясын шақыру қажет. Бұл функцияға тізімнен қандай түсті алу керектігін көрсетеміз. Мысалы, әр спираль қыры үшін сәйкес түсті алу үшін, цикл индексін colors тізімінің ұзындығына қарай модулдық операциямен бөліп, дұрыс элементті таңдаймыз (2.35-суретте көрсетілгендей). Осылайша, циклдың әр айналымында pencolor() функциясы шақырылып, сәйкес түс орнатылады, бұл спиральдың түрлі-түсті болуының себебі.

2.35 СУРЕТ - КВАДРАТКА ТӨРТ ТҮС БЕРУ

Төрт түрлі түстер тізімін пайдалану өте тиімді, сондықтан оны іске қосылған мысалда қолданған дұрыс (2.36-сурет). Спираль сызығы ұзара түскен сайын, түстердің ауысуы көрнекілікті арттырады және дизайнды әсем етеді. Мүмкіндігінше көп қадамға созылған сайын, нәтиже де соғұрлым әсерлі болады.

2.36 СУРЕТ - ТӨРТ ТҮСТІ КВАДРАТТЫҢ БЕЙНЕЛЕНУІ

Бағдарламадағы басты өзгеріс - pencolor функциясының параметрі ретінде қолданылатын (colors[x//4]) өрнегі. Мұндағы x - циклда қолданылатын айнымалы, ол 0-ден 99-ға дейін өзгереді. colors айнымалысы - Python бағдарламасына қосылған түстер тізімі екенін білдіреді. [x//4] өрнегі Python-ға түстер тізімінің алғашқы төрт элементін - 0-ден 3-ке дейінгі индекстерді - қолданатынымызды және x мәні ұлғайған сайын осы төрт түстің біреуін таңдайтынымызды көрсетеді. Осылайша, әр төрт қадам сайын тасбақаның қалам түсі өзгеріп отырады.

Мұндағы % операторы - модульді бөлу, яғни бөлгендегі қалдықты есептейді. Мысалы, 5 % 4 = 1, өйткені 5-ті 4-ке бөлгенде қалдық 1 шығады; 6 % 4 = 2, себебі 6-ны 4-ке бөлгенде қалдық 2 болады. Бұл оператор тізімдегі элементтерді қайталап қолдану қажет болғанда өте пайдалы. 100 қадамда өрнек [x % 4] төрт түсті (қызыл, сары, көк және жасыл - индекстері 0, 1, 2, 3) 25 рет қайталайды. Егер уақытыңыз болса және үлкейту әйнегі болса, 25 қызыл, 25 сары, 25 көк және 25 жасыл сызықтарды санауға болады (2.36-сурет). Циклдың бірінші кезеңінде Python тізімнен қызыл түсті таңдайды, келесі қадамда сарыны, тағы кейін көк пен жасылды пайдаланады. Бесінші қадамда тағы қызылға оралады, сосын сарыға және т.с.с. Осылайша, әр төртінші өтуден кейін цикл қайтадан қызылдан басталады. Фон түсін өзгертіп,

2.36-суреттегі өрнектен көркем әрі айқын дизайн жасауға болады. Мысалы, сары түс ақ фонға қарағанда нашар байқалады - экрандағы сары нүктелер ақ фонға еріп кетеді. Осы мәселені шешу үшін фон түсін қараға ауыстыруға болады. Бұл үшін импорттау жолынан кейін бағдарламаның кез келген жеріне келесі команданы қосыңыз:

Бір ғана команда арқылы экранда әсем сурет пайда болады: енді барлық түстер қара фонда айқын көрінеді. Тасбақаның параметрлері, яғни t айнымалысы өзгертілмегенін байқаңыз. Оның орнына біз тасбақа терезесінің - нақтырақ айтсақ, экранның - фон түсін өзгерттік. Turtle кітапханасындағы bgcolor() функциясы экранның фон түсін кез келген Python-ның қолжетімді түсіне өзгертуге мүмкіндік береді. Мысалы, bgcolor("black") командасы экранның фон түсін қара етіп қояды, сол арқылы қызыл, сары, көк және жасыл сияқты түстер жарқын әрі анық көрінеді. Бағдарламаны аяқтамас бұрын, циклдегі range() функциясының шегін 200 немесе одан үлкен мәнге өзгерте аласыз. Бұл спиральдың жолдарын ұзартып, үлкен әрі көркем өрнек жасайды. Нәтижесінде экранда 200 сызықтан тұратын және қара фонға орналастырылған жаңа сурет пайда болады (2.37-сурет).

2.37 СУРЕТ -BGCOLOR ЖАҢА ҚАТАРЫН ҚОСУ БАҒДАРЛАМАСЫ

Бұл бағдарламаның нәтижесі 2.38-суретте көрсетілген.

2.38 СУРЕТ - BGCOLOR ЖАҢА ҚАТАРЫН ҚОСУ КӨРІНІСІ

Спираль фигурасының түсі, өлшемі және бұрылу бұрышын басқару үшін айнымалыларды қолдандық. Енді фигураның қабырғыларының санын анықтайтын жаңа айнымалы - sides енгізейік. Бұл қосымша айнымалы спиральдың қалай өзгеретініне әсер етеді. Түсіну үшін, бұрынғы ColorSpiral.py бағдарламасын іске қосып көріңіз (2.39-сурет).

2.39 СУРЕТ - БАҒДАРЛАМАҒА SIDES АЙНЫМАЛЫСЫН ҚОСУ

Sides айнымалысын 6-дан 2-ге дейін төмендетуге болады (бір ғана қабырғасы бар фигураны салу қызықсыз болады, сондай-ақ кодтың алтыншы жолында жаңа түстерді қоспасаңыз, үлкен мәндерді пайдалану шектеледі). Бағдарламаны өз қалауыңызша өзгертіп, сақтап, іске қосыңыз. 2.40- 2.41- 2.42- 2.43- суреттерде sides=6, sides=5 және sides=2 болғандағы нәтижелер көрсетілген. Сонымен қатар, бояу циклiндегі түстердің ретін өзгертуге, санын көбейтуге немесе азайтуға болады. Егер бағдарламада қате пайда болса, бастапқы ColorSpiral.py файлына қайта оралып, оны қалпына келтіру керек болады.

2.40 СУРЕТ - 6 ҚЫРЛЫ СПИРАЛЬ КӨРІНІСІ

ColorSpiral.py бағдарламасында біз жаңа t.width() командасын қолдандық. Бұл команда тасбақа тұтқасының сызық қалыңдығын реттеуге мүмкіндік береді. Бағдарламада сызықтар ұзара келе, олардың қалыңдығы да біртіндеп артатынын байқаймыз.

2.41 СУРЕТ -5 ҚЫРЛЫ СПИРАЛЬ КӨРІНІСІ

2.42 СУРЕТ -4 ҚЫРЛЫ СПИРАЛЬ КӨРІНІСІ

2.43 СУРЕТ - 3 ҚЫРЛЫ СПИРАЛЬ КӨРІНІСІ

```
Мәтін арқылы спираль түрін өзгертуге болады  (сурет 68).
```

2.44 СУРЕТ - СПИРАЛЬҒА АТЫН ЕНГІЗУ БАҒДАРЛАМАСЫ

Нәтижесі 2.45- суреттегідей шығады.

2.45 СУРЕТ -ТҮРЛІ ТҮСТІ МӘТІНДІК СПИРАЛЬ

Егер бағдарламаны іске қосқанда қолданушыға қырлар санын өздігімен енгізу мүмкіндігін бергіміз келсе, онда sides айнымалысына пайдаланушының енгізген мәнін сақтап қоюға болады. Мұны істеу үшін input() функциясын қолданамыз. Қолданушы енгізген мәнді дұрыс сан ретінде өңдеу үшін eval() функциясын пайдалануға болады (2.46-2.56 суреттерде көрсетілгендей). Бұл тәсіл арқылы бағдарламамыз қолданушының енгізген мәнін сан ретінде қабылдап, sides айнымалысына меншіктейді.

2.46 СУРЕТ - ПАЙДАЛАНУШЫДАН ҚЫРЛАР САНЫН ЕНГІЗУДІ СҰРАУ БАҒДАРЛАМАСЫ

2.47 СУРЕТ -ПАЙДАЛАНУШЫДАН ҚЫРЛАР САНЫН ЕНГІЗУДІ СҰРАУ БАҒДАРЛАМАСЫНЫҢ ОРЫНДАЛУЫ. 5 ҚЫРЛЫ

2.48 СУРЕТ - ПАЙДАЛАНУШЫДАН ҚЫРЛАР САНЫН ЕНГІЗУДІ СҰРАУ БАҒДАРЛАМАСЫ НӘТИЖЕСІ. 5 ҚЫРЛЫ

2.49 СУРЕТ - ПАЙДАЛАНУШЫДАН ҚЫРЛАР САНЫН ЕНГІЗУДІ СҰРАУ БАҒДАРЛАМАСЫНЫҢ ОРЫНДАЛУЫ. 3 ҚЫРЛЫ

2.50 СУРЕТ - ПАЙДАЛАНУШЫДАН ҚЫРЛАР САНЫН ЕНГІЗУДІ СҰРАУ БАҒДАРЛАМАСЫ НӘТИЖЕСІ. 3 ҚЫРЛЫ

2.51 СУРЕТ - ПАЙДАЛАНУШЫДАН ҚЫРЛАР САНЫН ЕНГІЗУДІ СҰРАУ БАҒДАРЛАМАСЫНЫҢ ОРЫНДАЛУЫ. 6 ҚЫРЛЫ

2.52 СУРЕТ - ПАЙДАЛАНУШЫДАН ҚЫРЛАР САНЫН ЕНГІЗУДІ СҰРАУ БАҒДАРЛАМАСЫ НӘТИЖЕСІ. 6 ҚЫРЛЫ

2.53 СУРЕТ - ПАЙДАЛАНУШЫДАН ҚЫРЛАР САНЫН ЕНГІЗУДІ СҰРАУ БАҒДАРЛАМАСЫНЫҢ ОРЫНДАЛУЫ. 2 ҚЫРЛЫ

2.54 СУРЕТ - ПАЙДАЛАНУШЫДАН ҚЫРЛАР САНЫН ЕНГІЗУДІ СҰРАУ БАҒДАРЛАМАСЫ НӘТИЖЕСІ. 2 ҚЫРЛЫ

2.55 СУРЕТ - ПАЙДАЛАНУШЫДАН ҚЫРЛАР САНЫН ЕНГІЗУДІ СҰРАУ БАҒДАРЛАМАСЫНЫҢ ОРЫНДАЛУЫ. 4 ҚЫРЛЫ

2.56 СУРЕТ - ПАЙДАЛАНУШЫДАН ҚЫРЛАР САНЫН ЕНГІЗУДІ СҰРАУ БАҒДАРЛАМАСЫ НӘТИЖЕСІ. 4 ҚЫРЛЫ

Біз Python тіліндегі Turtle кітапханасын қолданып, түрлі-түсті және әсерлі фигураларды салуды үйрендік. Turtle кітапханасын пайдалану үшін бағдарламаға "import" командасы арқылы қосу керек екенін білдік. Сонымен қатар, дайын кодты қайта қолдану - бағдарламалаудағы ең тиімді тәсілдердің бірі екенін түсіндік. Бұл әдіс арқылы өзіміз уақыт үнемдеп, басқалардың жазған пайдалы кодтарын өз жобамызға импорттап, күрделі әрі әдемі шешімдер жасай аламыз. Бағдарламамызда x және sides сияқты айнымалылар қолданылды. Айнымалылар - бұл орындалу барысында сан немесе мән сақтап тұратын контейнерлер, оларды бірнеше рет пайдаланып, қажет болғанда өзгерте аламыз.

2.6 Pygame графикалық интерфейсі

Графикалық пайдаланушы интерфейсі (GUI, ағылшын тілінен «graphical user interface») - компьютер экранында көрінетін батырмалар, иконкалар, мәзірлер мен терезелер жиынтығы. Қысқаша айтқанда, GUI - бұл пайдаланушы мен компьютер арасындағы визуалды өзара әрекеттесу ортасы. Мысалы, файлды ашу үшін оның белгішесін екі рет шерту немесе ойындарда пернелерді басу, тышқанды жылжыту, шерту сияқты әрекеттер GUI арқылы жүзеге асады. Бағдарламашы осы графикалық интерфейсті құрып, пайдаланушы әрекеттеріне жауап беруді қамтамасыз етеді. Қазіргі заманғы көпшілік графикалық интерфейстер WIMP моделіне негізделеді, яғни Window (терезе), Icon (иконка), Menu (мәзір), Pointer (көрсеткіш). Бұл модель негізінде әрбір терезеде бірнеше графикалық элементтер - виджеттер орналасады. Мәзірлер терезенің әртүрлі аймақтарында болуы мүмкін, бірақ олардың мақсаты - алдын ала анықталған әрекеттер тізімінен қажетті функцияны таңдау. Пайдаланушы графикалық интерфейске көрсеткішпен (әдетте, тінтуірдің меңзері немесе джойстик) әрекет етеді, бірақ басқа да нұсқағыш құрылғылар қолданылады. Графикалық интерфейстің басты мақсаты - пайдаланушы мен компьютер арасындағы қарым-қатынасты жеңілдету. Интерфейс жобалауда бағдарламашы немесе дизайнер ең қолайлы виджеттерді таңдап, қажетсіз элементтерді азайтуы тиіс. Сонымен қатар, интерфейстің сыртқы пішіні мен оның функциялары пайдаланушыға бірден түсінікті болуы қажет. Жақсы жобаланбаған GUI, тіпті ішкі алгоритмдері күрделі болса да, пайдаланушыға түсініксіз болуы мүмкін. Интерфейс пайдаланушының жиі жасайтын әрекеттерін жеңіл әрі ыңғайлы түрде қамтамасыз етуі керек. Көптеген бағдарламаларда мұндай әрекеттерді бөліп көрсету үшін «шеберлер» (wizards) деп аталатын арнайы қадамдық экрандар қолданылады. Бірақ егер қолданушы өзіне қажетті шешімдерді өз қолымен жасауға мүмкіндік беретін конструктор болса, бұл да тиімді тәсіл болып табылады. Типтік әрекеттерді анықтау әрдайым оңай емес, сондықтан кейде «шеберлер» мен конструкторлар үйлесімін қолдану ыңғайлы. Дегенмен, барлық жағдайда графикалық интерфейс ең тиімді шешім емес. Кейбір салаларда пайдаланушыға шешімді формальды тіл немесе сценарий түрінде сипаттау әлдеқайда ыңғайлы болады. Кітапхана дегеніміз - функциялар мен кластардың жиынтығы болып табылатын кодтар топтамасы. Python тілінде мұндай код кітапханаларын сипаттау үшін «модуль» (module) термині қолданылады. Мысалы, pygame модулінің құрамында pygame.draw, pygame.image, pygame.mouse сияқты ішкі модульдер бар. PyGame - бұл Python программалау тілінде 2D ойындар мен мультимедиялық қосымшаларды жасауға арналған арнайы модуль. Turtle кітапханасына ұқсас, Pygame өте визуалды және ойындар, анимациялар мен басқа графикалық интерфейстерді құру үшін өте қолайлы. Бұл кітапхана кроссплатформалық болып табылады, яғни Windows, macOS және Linux сияқты әртүрлі операциялық жүйелерде жұмыс істейді. Сондықтан Pygame көмегімен жазылған ойындар мен бағдарламаларды әртүрлі компьютерлерде орындауға болады. Pygame кітапханасын орнату үшін ресми Pygame сайтына кіру қажет (2.57-сурет).

2.57 СУРЕТ - PYGAME КІТАПХАНАСЫ САЙТТА ОРНАЛАСТЫРЫЛҒАН ОҚУЛЫҚТАР МЕН ОЙЫН МЫСАЛДАРЫ КӨРІНІСІ

Конец формы

2.3 КЕСТЕ - PYGAME-НІҢ НЕГІЗГІ МОДУЛЬДЕРІ

Модуль атауы	Мақсаты
pygame.cdrom	CD-ROM жетектерін басқару және оған кіру

pygame.cursors	Курсор бейнелерін жүктеу
pygame.display	Экранға кіру
pygame.draw	Фигураларды, сызықтарды және нүктелерді сызу
pygame.event	Сыртқы оқиғаларды басқару
pygame.font	Жүйелік қаріптерді қолдану
pygame.image	Кескінді жүктеу және сақтау
pygame.key	Пернетақтада басқан пернені оқу
pygame.mixer	Дыбыстарды жүктеу және ойнату
pygame.mouse	Тінтуірмен басқару
pygame.movie	Кинофильм файлдарын ойнату
pygame.music	Музыка және аудиолармен жұмыс
pygame.rect	Төртбұрышты аймақтарды басқару
pygame.sndarray	Дыбыс деректерін басқару
pygame.surface	Суреттер мен экранды басқару
pygame.transform	Суреттерді өзгерту және жылжыту
pygame.time	Уақыт пен кадр жиілігін басқару

Pygame көмегімен нүкте салу. Pygame орнатылғаннан кейін, 2.58-суреттегі мысалға сәйкес, экранға нүкте салу үшін қарапайым бағдарламаны іске қосуға болады.

2.58 СУРЕТ - БАҒДАРЛАМА SHOWDOT.PY ӘРЕКЕТІ

Келесі кодты жаңа IDLE терезесіне көшіріп енгізіңіз: # ShowDot.py

```
import pygame
pygame.init()
```

screen = pygame.display.set_mode([800, 600]) # Терезенің өлшемін белгілейміз: ені 800, биіктігі 600 пиксель

```
running = True

GREEN = (0, 255, 0)  # RGB форматында жасыл түс (қызыл, жасыл, көк)
```

radius = 50 # Шеңбердің радиусы 50 пиксель

while running:

```
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    pygame.draw.circle(screen, GREEN, (100, 100), radius)  # Экранға жасыл шеңбер салу
    pygame.display.update()  # Экрандағы өзгерістерді көрсету

pygame.quit()  # Бағдарламаны аяқтап, Pygame-ді дұрыс жауып қою
```

Осы бағдарламаның әрбір жолын түсініп көрейік:

```
    •    Алдымен pygame модулі импортталады, бұл арқылы оның функцияларын қолдана аламыз. Pygame-ді пайдалану үшін бағдарламаның басында әрдайым import pygame деп жазу керек.
```

- pygame.display.set_mode([800, 600]) функциясы 800 пиксель ені мен 600 пиксель биіктігі бар экран терезесін жасайды. Бұл экранның беті screen айнымалысына сақталады. Pygame-де терезе мен графика беттері деп аталады, оларда барлық кескіндер көрсетіледі.
- Ойын циклында қолданылған keep_going (немесе running) айнымалысы циклдің жұмыс істеуін басқарады. Осы мысалда ол пайдаланушы терезені жапқанша экранда графиканы үздіксіз салу үшін пайдаланылады.
- Екі айнымалы - GREEN және radius - шеңберді сызу кезінде қолданылады. GREEN айнымалысы ашық жасыл түске сәйкес RGB форматындағы (0, 255, 0) мәнін қабылдайды.
- RGB (қызыл, жасыл, көк) - түстерді көрсету тәсілі, онда әр түс 0-ден 255-ке дейінгі мәнмен анықталады. Мысалы, (0, 255, 0) - таза жасыл түс.
- Айнымалы атаулары кейде тұрақты мән ретінде қарастырылады, сондықтан оларды бас әріптермен жазуға болады. Мұнда GREEN тұрақты түс ретінде белгіленген, өйткені оның мәні бағдарлама барысында өзгермейді. radius айнымалысына 50 мәні берілген, бұл радиусы 50 пиксель шеңберді салуға мүмкіндік береді (диаметрі - 100 пиксель).
- Бағдарламадағы басты цикл - ойын циклі. Ол пайдаланушы ойыннан шыққанға дейін үздіксіз жұмыс істейді.
- Цикл ішінде for құрылымы арқылы барлық оқиғалар тексеріледі. Бұл мысалда тек бір оқиғаға назар аударылады - пайдаланушының терезенің жабу батырмасын басуы (pygame.QUIT), ол keep_going айнымалысын False етіп өзгертіп, циклды тоқтатады.
- pygame.draw.circle() функциясы экран бетінде (screen) жасыл (GREEN) түспен, радиусы 50 болатын шеңберді (100, 100) координатасында салады. Бұл координаталар терезенің жоғарғы сол жақ бұрышынан бастап есептеледі: оңға 100 пиксель және төмен 100 пиксель.
- Pygame-де шеңбер, тіктөртбұрыш, сегмент сияқты фигураларды салу үшін арнайы функциялар бар.
- Экрандағы өзгерістерді көрсету үшін pygame.display.update() функциясы шақырылады, ол жаңа сызбаларды экранға шығарады.
- Бағдарлама аяқталған соң, яғни пайдаланушы ойын циклынан шыққанда, pygame.quit() шақырылып, Pygame модулі тазартылады - бұл барлық ашылған ресурстарды босатып, экран терезесін жабуға мүмкіндік береді. Осылайша, Pygame кітапханасын пайдалану арқылы қарапайым сурет салудың қаншалықты оңай және қуатты екенін көреміз. Бұл бағдарлама негіз болып, әрі қарай күрделі графика, анимация және ойындарды жасауға мүмкіндік береді.

Бақылау сұрақтары

- Python-ның графикалық мүмкіндіктері қандай кітапханалар арқылы жүзеге асады?
- Python-да графика құру үшін қандай құралдар қолданылады?
- GUI дегеніміз не?
- Python-да графикалық интерфейсті жасау үшін қандай негізгі модульдер бар?
- Tkinter кітапханасы туралы қысқаша түсінік беріңіз.
- Pygame модулі қандай мақсатта қолданылады?
- Turtle кітапханасының негізгі ерекшеліктері қандай?
- Python графикасында пиксель дегеніміз не?
- Экран координат жүйесі қалай ұйымдастырылған?
- Графикада түс қалай көрсетіледі?
- Python-ның графикаға қатысты қандай негізгі функциялары бар?
- Графикадағы оқиғалар (event) деген не?
- Python-да графикалық элементтерді қалай басқаруға болады?
- Графикадағы анимация дегеніміз не?
- Графикалық интерфейстің негізгі элементтері қандай?
- Python-да экран түсін өзгерту үшін қандай әдістер қолданылады?
- RGB түс жүйесі қалай жұмыс істейді?
- Python-да түс параметрлерін қалай көрсетуге болады?

- Pygame-да экран түсін өзгерту үшін қандай функция қолданылады?
- Turtle кітапханасында экран түсін қалай өзгертеміз?
- Түсті атаумен қалай анықтауға болады?
- Python-да түс кодын қалай жазамыз?
- Экранның фон түсін динамикалық өзгерту үшін қандай алгоритм қолдануға болады?
- Түсті градиентпен қалай ауыстыруға болады?
- Экран түсін цикл ішінде өзгертуге бола ма?
- RGB түсінің әрбір компоненті қандай диапазонда болуы керек?
- Pygame экранында түсті қалай тазартуға болады?
- Python-да HEX форматындағы түсті қалай қолдануға болады?
- Экран түсін өзгерткенде қандай қателіктер жиі кездеседі?
- Экран түсін өзгертуге арналған кітапханаларды салыстырыңыз.
- Python-да цикл дегеніміз не?
- while циклінің негізгі құрылымы қандай?
- for циклы қалай жұмыс істейді?
- Шартты операторлардың негізгі түрлері қандай?
- if, elif, else операторларының айырмашылығы неде?
- Цикл ішінде шартты операторды қалай қолдануға болады?
- Циклдан шығу үшін қандай оператор қолданылады?
- break және continue операторларының айырмашылығы неде?
- Циклдар мен шарттардың графикада қолданылуы қалай жүзеге асады?
- Цикл ішінде графикалық объектілерді қалай жаңартуға болады?
- Циклдың шартын қате жазған жағдайда не болады?
- for циклының жұмыс істеу аймағын сипаттаңыз.
- while циклінің шексіз циклге айналуын қалай болдырмауға болады?
- if операторымен қатар логикалық операторларды қалай қолданамыз?
- Python-да шарттар мен циклдардың орындалу реті қандай?
- Шартты операторлардың негізгі мақсаты неде?
- Python-да if операторы қалай жазылады?
- Екі немесе одан көп шартты қалай біріктіруге болады?
- Шарттарда логикалық операторлардың қолданылуын түсіндіріңіз.
- else блогының міндеті неде?
- Python-да көп деңгейлі шарттарды қалай жазамыз?
- Тернарлы оператор деген не және қалай қолданылады?
- Шартты операторда қандай типтегі мәліметтерді тексеруге болады?
- Қандай жағдайда if операторы орындалмайды?
- if...elif...else конструкциясының жұмысы қалай жүзеге асады?
- Шартты операторлардың шарттарын жазуда жиі кездесетін қателіктер қандай?
- Пайдаланушы енгізген мәнге шартты оператор арқылы қалай жауап беруге болады?
- Python-да шарттар мен циклдарды біріктіру мысалын келтіріңіз.
- Шарттарды жазу кезінде скобкаларды қолдану қажет пе?
- if операторының ішінде тағы да if қолдану мүмкін бе?
- Turtle кітапханасының мақсаты қандай?
- Turtle графикасында жұлдыз салу үшін қандай командалар пайдаланылады?
- Turtle модулін қалай импорттаймыз?
- Turtle экраны қалай ашылады?

- Жұлдыз сызу үшін циклді қалай қолданамыз?
- Turtle-да түстерді қалай орнатуға болады?
- Позицияны қалай өзгертуге болады?
- Turtle-де жылдамдықты қалай баптаймыз?
- Жұлдыздың әр бұрышын қалай есептейміз?
- Turtle графикасында цикл мен шартты операторлың ролі қандай?
- Turtle кітапханасында экранды тазарту командасы қандай?
- Жұлдызды сызу кезінде қандай қателіктер болуы мүмкін?
- Turtle графикасында координат жүйесі қалай ұйымдастырылған?
- Жұлдызды салу кезінде Turtle-дің бағытын қалай өзгертуге болады?
- Turtle кітапханасының артықшылығы мен кемшіліктерін атаңыз.
- Pygame деген не?
- Pygame-ды қалай орнатамыз?
- Pygame модулін программада қалай импорттаймыз?
- Pygame-де экранды қалай құруға болады?
- Pygame-да түсті қалай орнатамыз?
- Pygame-да оқиғаларды қалай өңдейміз?
- Pygame-да ойын циклінің құрылымы қандай?
- Pygame-де графикалық элементтерді қалай салуға болады?
- Pygame-де pygame.draw.circle() функциясының аргументтері қандай?
- Pygame-де экранды жаңарту үшін қандай функция қолданылады?
- Pygame терезесін қалай жабуға болады?
- Pygame-де шрифттар мен мәтінді қалай көрсетуге болады?
- Pygame-да дыбыстарды қалай қосуға болады?
- Pygame-де спрайттар деген не?
- Pygame-да қозғалысты қалай жүзеге асырамыз?
- Pygame-да таймерлер қалай жұмыс істейді?
- Pygame-де анимация жасау принциптері қандай?
- Pygame-де тінтуір мен клавиатура оқиғаларын қалай өңдейміз?
- Pygame-да оқиға циклінің маңызды элементтері қандай?
- Pygame-да фондық суреттерді қалай жүктеп, көрсетуге болады?
- Pygame-да қателіктерді қалай табуға және түзетуге болады?
- Pygame-де ойынның FPS мәнін қалай бақылауға болады?
- Pygame-да объектілердің позициясын қалай басқаруға болады?
- Pygame-да графикалық жобаларды ұйымдастырудың тиімді тәсілдері қандай?
- Pygame-ды үйрену үшін қандай қосымша ресурстарды пайдалануға болады?

ГЛОССАРИЙ

Алгоритм - белгілі бір тапсырманы орындау үшін қажет болатын нақты қадамдар жиынтығы, мысалы, ас әзірлеу рецепті секілді. Анимация - жылдам ауысатын ұқсас суреттер тізбегі, ол қозғалыс иллюзиясын тудырады, мысалы, мультфильмдегі кадрлар.

```
Аргумент — функцияға берілетін енгізу мәні. Мысалы, range(10) функциясында 10 — аргумент.
```

Блок - бірнеше программалық командалардан тұратын топ. Логикалық мәндер - шындыққа сәйкес келетін (True) немесе сәйкес келмейтін (False) мәндер немесе олардың комбинациялары. Енгізу - компьютерге берілетін кез келген деректер немесе ақпарат, оны пернетақта, тышқан, микрофон, камера сияқты құрылғылар арқылы жіберуге болады.

```
Ішкі цикл (кірістірілген цикл) — бір циклдың ішінде орналасқан басқа цикл.
```

Өрнек - нәтижені есептейтін айнымалылар, функциялар, операторлар мен мәндердің белгілі бір ретпен жазылуы. Диапазон - белгілі бір басталу және аяқталу мәндері арасында орналасқан реттелген мәндер жиыны. Python-да range функциясы мысалы, 0-ден 9-ға дейінгі сандарды береді. Импорттау - басқа файлдан немесе модульден жазылған кодты ағымдағы бағдарламаға қосу. Индекс - тізімдегі элементтің орны немесе нөмірі. Инициализация - айнымалыға немесе объектіге алғашқы мәнді беру процесі. Итеративті нұсқа - бағдарламаға кіші өзгерістер енгізіп, оны жаңа нұсқа ретінде сақтау әдісі, мысалы, "Ойын1", "Ойын2" деген сияқты. Кадр - анимация немесе видеоға арналған жеке сурет. FPS (секундына кадрлар саны) - анимациядағы немесе ойындағы экрандағы кескіндердің секундына көрсетілу жылдамдығы. Класс - объектілердің қасиеттері мен әрекеттерін анықтайтын бағдарламалық үлгі. Кілт сөз - бағдарламалау тілінде арнайы мағынасы бар, өзгертілмейтін сөздер. Код - компьютер түсінетін және орындауға арналған программалық нұсқаулар жиыны.

```
Қосу (байланыстыру) — мәтіннің екі бөлігін бір жолға біріктіру операциясы.
Тұрақты — мәні өзгермейтін айнымалы сияқты атаулы мән, мысалы math.pi (3.1415...).
```

Массив - бір типтегі мәндер немесе объектілердің реттілігі, әр элементке индекспен қол жеткізіледі. Модуль - қайта қолдануға арналған функциялар, айнымалылар немесе сыныптар жиынтығы бар файл немесе файлдар пакеті. Соқтығысуды анықтау - экрандағы екі виртуалды объектінің (мысалы, доп пен ракетка) жанасуын тексеру. Қабық (shell) - командаларды қабылдап, орындаушы мәтіндік интерфейс. Мысалы, Python-ның IDLE қабығы. Нысан - кластан жасалған айнымалы, нақты объектіні білдіреді, мысалы, Sprite класының бір элементі.

```
Хабарландыру (декларация) — айнымалы немесе функцияның атын компьютерге таныстыру.
```

Параметр - функцияның кіріс мәнін анықтайтын айнымалы, функция анықтамасында көрсетіледі. Айнымалы - бағдарламада мәнін өзгерте алатын атаулы деректер.

```
Пиксель — экрандағы ең кіші түсті нүкте (picture element).
Қосымша (ауыстыру) — жолға немесе тізімге жаңа элементтерді соңынан қосу операциясы.
```

Тағайындау - айнымалыға мән беру операциясы, мысалы x = 5 - x айнымалысына 5 мәнін беру.

БАҒДАРЛАМА - КОМПЬЮТЕРГЕ ОРЫНДАЛАТЫН ӘРЕКЕТТЕР ТІЗБЕГІ РЕТІНДЕ ЖАЗЫЛҒАН НҰСҚАУЛАР ЖИЫНЫ.

Мөлдірлік - графикада кескіннің артындағы нысандарды көрсету мүмкіндігі. Жалған кездейсоқтық - кездейсоқ көрінетін, бірақ алгоритм бойынша анықталатын мәндер тізбегі, мысалы ойындағы кездейсоқтықты модельдеу үшін қолданылады. Синтаксис - бағдарламалау тілінің грамматикалық және құрылымдық ережелері. Кездейсоқ сандар - белгілі бір диапазонда алдын ала болжамсыз сандар тізбегі. Оқиға - компьютер қабылдай алатын әрекет, мысалы тышқанды басу, пернені басу немесе таймердің аяқталуы. Оқиғаларға жауап беретін функциялар «оқиға өңдеушілер» деп аталады.

```
Жол (string) — әріптер, сандар, символдар мен бос орындардың қатарынан тұратын мәтін.
```

Шартты өрнек - белгілі бір шарттың орындалуына байланысты әртүрлі әрекеттерді жүзеге асыратын өрнек. Функция - белгілі бір тапсырманы орындауға арналған командалар жиынтығы. RGB түсі - қызыл, жасыл және көк компоненттердің араласуымен алынатын түсті білдіреді. For циклі - белгілі бір блокты бірнеше рет орындауға арналған цикл конструкциясы.

ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТТЕР

- A. Sweigart Разработка компьютерных игр на языке Python / Sweigart A. - 2-е изд., испр. - М.: Национальный Открытый Университет «ИНТУИТ», 2016. -505 с.
- Ч. Северенс Введение в программирование на Python / Северенс Ч. - 2-е изд., испр. - М.: Национальный Открытый Университет «ИНТУИТ», 2016. - 231 с.
- Л.П. Коэльё, В. Ричерт Построение систем машинного обучения на языке Python: пер. с англ. / Коэльё Л.П., Ричерт В. - М.: ДМК Пресс, 2015. – 234 с.
- У. Маккинли Python и анализ данных: пер. с англ. / Маккинли У. - М.: ДМК Пресс, 2015. - 482 с.
- Марк Саммерфилд Python на практике: пер. с англ. / Марк Саммерфилд - М.: ДМК Пресс, 2014. - 338 с.
- Марк Лутц. Программирование на Python: пер. с англ. / Марк Лутц. - СПб.: Символ-Плюс, 2011. - Т.I, II. - 992 с.
- Ж.Ахметова, А.Баймұқанов Python тілінде бағдарламалау. - Алматы: Қазақ университеті, 2022. - 184 б.
- Информатика. Бағдарламалау негіздері (Python). - Электронды оқулық, Bilimland.kz платформасы, 2023.
- А. Әлібекова, С. Оразбекова Бағдарламалау негіздері. - Астана: НЗМ ДББҰ, 2021. - 136 б.
- Python тілі бойынша кіріспе курс. - Aitu.edu.kz - Цифрлық Қазақстан бағдарламасы аясында, 2022.
- Informatik.kz - Python тілінде Turtle - шартты операторлар және циклдер бойынша әдістемелік материалдар.
- Б. Любанович Простой Python. Современный стиль программирования. - СПб: Питер, 2022. - 384 с.
- М. Лутц Изучаем Python. - 5-е изд. - СПб: БХВ-Петербург, 2020. - 1216 с.
- А. Свейгарт Автоматизация рутинных задач с помощью Python. - М.: ДМК Пресс, 2021. - 592 с.
- А. Васильев Самоучитель Python 3. - М.: Наука и Техника, 2020. - 320 с.
- Бриггс Дж. Python для детей. Весёлое знакомство с программированием. - СПб: Питер, 2018. - 368 с.
- E. Matthes Python Crash Course. - 3rd ed. - No Starch Press, 2023. - 552 pp.
- A. Sweigart Automate the Boring Stuff with Python. - 2nd ed. - No Starch Press, 2019. - 592 pp.
- M. Lutz Learning Python. - 5th ed. - O’Reilly Media, 2013. - 1600 pp.
- A. B. Downey Think Python: How to Think Like a Computer Scientist. - 2nd ed. - O’Reilly Media, 2015. - 292 pp.
- J. Zelle Python Programming: An Introduction to Computer Science. - 3rd ed. - Franklin, Beedle & Associates, 2016. - 552 pp.

ИНТЕРНЕТ РЕСУРСТАР

- https://www.pygame.org
- https://Python-scripts.com›tkinter
- https://www.pygame.org/wiki/GettingStarted
- https://github.com›topics/tkinter-game
- https://informatik.kz
- https://www.youtube.com/@pythonqazaqsha
- https://aitu.edu.kz
- https://stepik.org/course/67
- https://habr.com/ru/hub/python/
- https://pythonworld.ru
- https://www.python.org.
- https://docs.python.org/3/tutorial/
- https://realpython.com
- https://www.w3schools.com/python/
- https://www.codecademy.com/learn/learn-python-3
- https://automatetheboringstuff.com