Спеціалізовані мови програмування PYTHON

Викладач

Малінов Владислав Андрійович

Зміст курсу

1. ВВЕДЕННЯ В МОВУ ПРОГРАМУВАННЯ PYTHON
2. ОСНОВНІ СТАНДАРТНІ МОДУЛІ PYTHON
3. ЕЛЕМЕНТИ ФУНКЦІОНАЛЬНОГО ПРОГРАМУВАННЯ
4. ОБЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ
5. ЧИСЕЛЬНІ АЛГОРИТМИ. МАТРИЧНІ ОБЧИСЛЕННЯ
6. ОБРОБКА ТЕКСТІВ
7. РОБОТА З ДАНИМИ В РІЗНИХ ФОРМАТАХ
8. РОЗРОБКА WEB-ДОДАТКІВ
9. МЕРЕЖНІ ДОДАТКИ НА PYTHON

ВВЕДЕННЯ В МОВУ ПРОГРАМУВАННЯ PYTHON

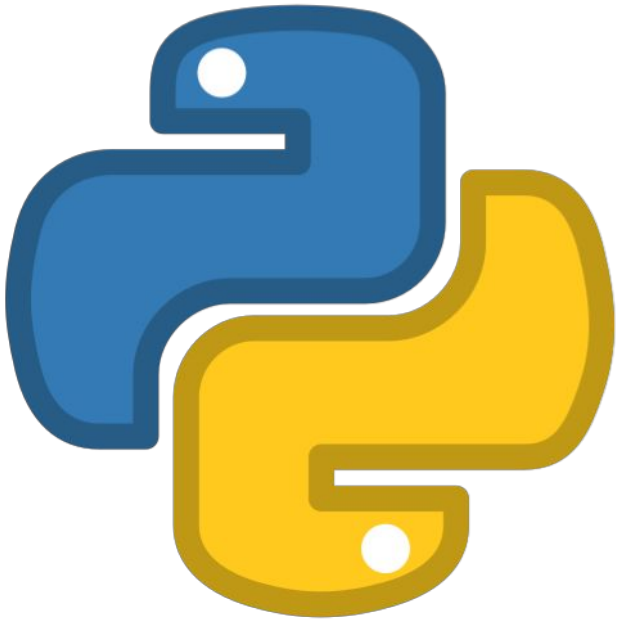
1.1. Що таке Python?
1.2. Як описати мову?
1.3. Історія мови Python
1.4. Основні алгоритмічні конструкції
1.5. Вбудовані типи даних
1.6. Вирази
1.7. Імена

Практична робота

Висновок

Питання для самоконтролю

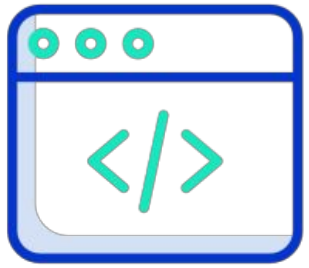
1.1. Що таке Python?



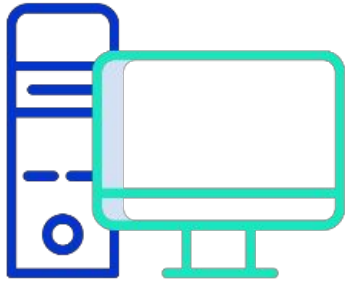
Python - це безкоштовна, open-source, кроссплатформенна мова програмування.

Python є мовою, що інтерпретується, це означає, що їй не потрібно перетворювати код в машинний для того, щоб виконати програму. Натомість код читається рядковим інтерпретатором.

Для чого використовується Python

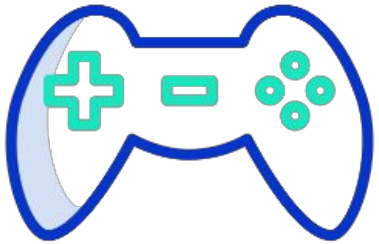


Web розробка

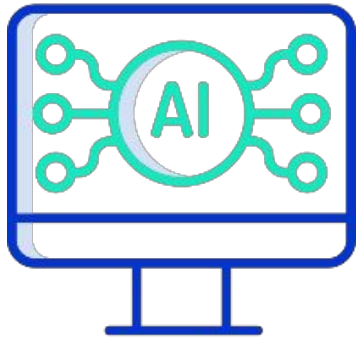


Desktop програми

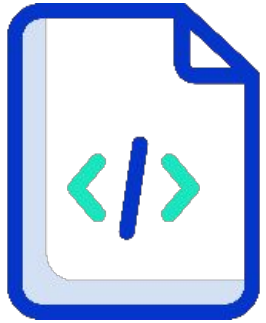
Аналіз даних та візуалізація



Розробка ігор



Data Science



Написання скриптів

1.2. Як описати мову?

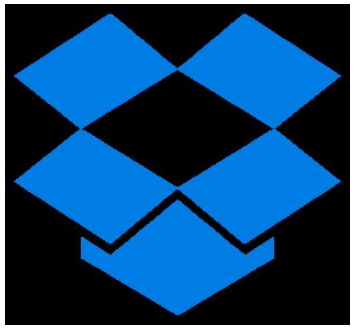
Найбільш повний опис мови Python можна знайти за посиланням https://www.python.org/

1.3. Історія мови Python

Python – порівняно «молода» мова. Створюючи її у 1990-1991 роках, її автор Гвідо ван Россум (Guido van Rossum) врахував усі переваги та недоліки попередніх мов програмування. Мова почала вільно поширюватися через Інтернет і сподобалася іншим програмістам. З 1991 року Python є цілком об'єктно-орієнтованим. Python також запозичив багато рис таких мов, як C, C++ й окремі риси функціонального програмування. 3 грудня 2008 року, після тривалого тестування, вийшла перша версія Python 3

Області застосування

- скрипти, утиліти
- наукова сфера
- дослідження даних
- веб-застосунки
- сервіси
- розробка ігор



Інтерпретатор та інтегроване середовище розробки

Інтерпретатор Python

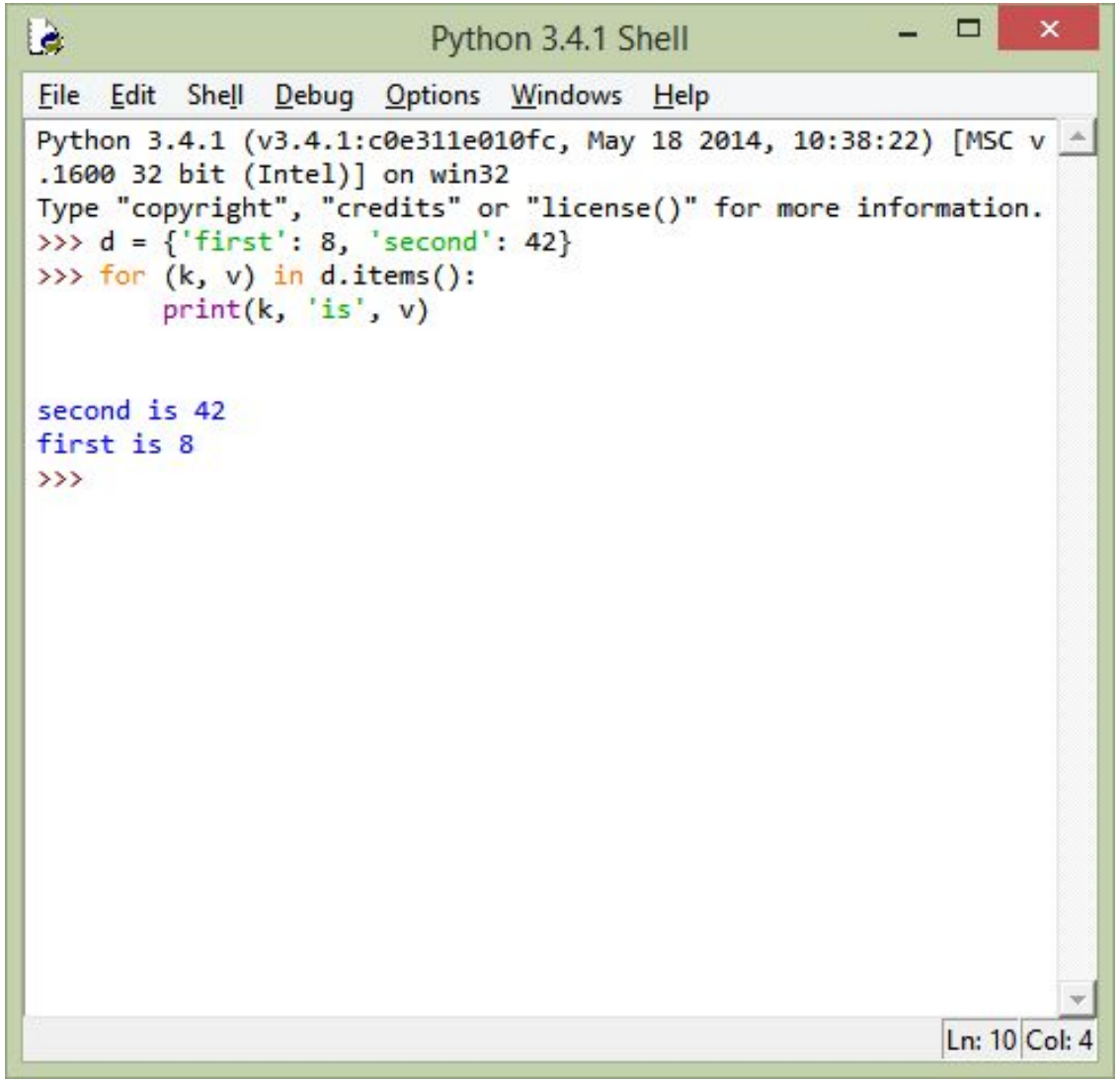
Інтерпретатор - програма (різновид транслятора), що виконує інтерпретацію. Офіційний сайт Python: https://python.org/

Завантажити інтерпретатор: https://www.python.org/downloads/

Два режими роботи:

• виконання програм;

• Інтерактивний режим.

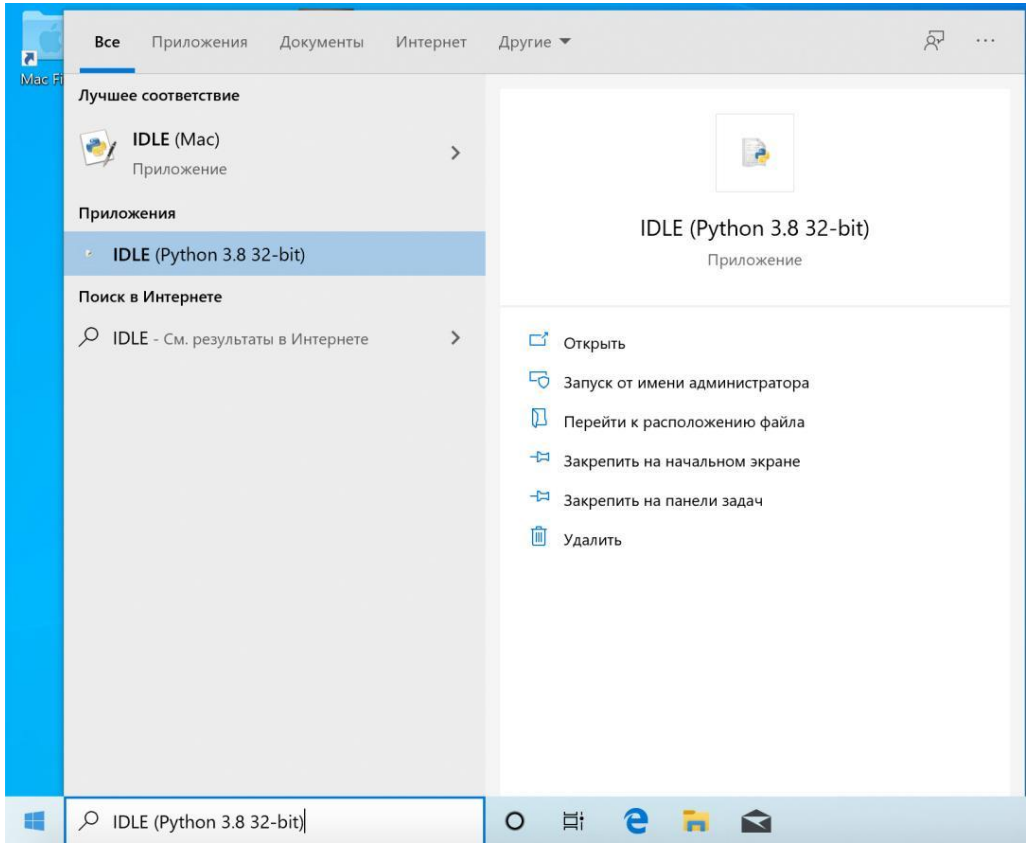


Як писати програми на Python

Є два варіанти написати програму на Python:
Це скористатись IDLE (інтегроване середовище розробки для Python), яке у вас автоматично буде встановлене на комп'ютер разом з Python.
Це встановити середовище розробки (IDE) від сторонніх виробників.

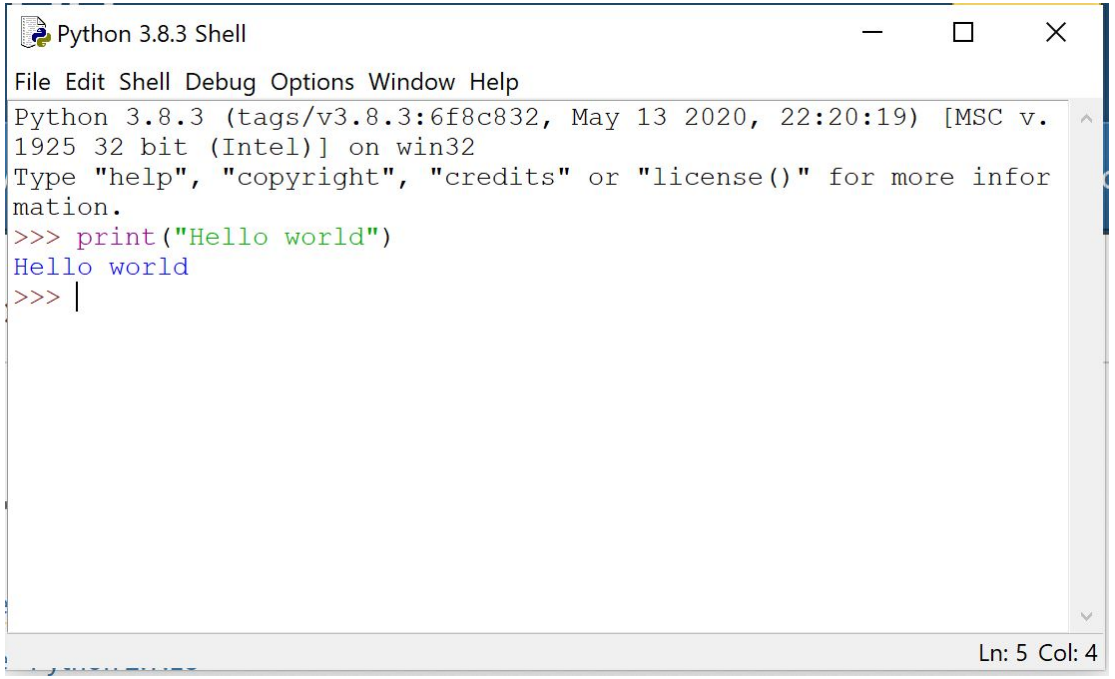
Як писати програми на Python

Для того, щоб запустити IDLE, скористайтесь пошуком:



Як писати програми на Python

Щоб вивести повідомлення, використовуємо метод print, передаючи йому текст, який ви хочете вивести.
Далі вводимо наш код у відкритому вікні програми та натискаємо Enter:



1.4. Основні алгоритмічні конструкції

```
>>> a = 1
>>> b = 2
>>> a = a + b >>> b = a - b >>> a = a - b >>> print(a,
b) 21
```

```
if a > b: c =a
else:
c= b
```

1.4. Основні алгоритмічні конструкції

```
if a < 0: s = -1
elif a == 0: s =0
else:
s =1
```

```
if a < 0: s = -1
else:
if a == 0:
s =0 else:
s =1
```

1.4. Основні алгоритмічні конструкції

```
s = "abcdefghijklmnop" while s != "":
print(s)
s = s[1:-1]
```

```
f = open("file.txt", "r") while true:
l = f.readline()
```

```
if not l: break
if len(l) > 5: print(l)
f.close()
```

1.4. Основні алгоритмічні конструкції

```
for i in range(1, 10):
 for j in range(1, 10):
print("%2i" % (i*j))
print()
```

1.4. Основні алгоритмічні конструкції

```
def price(hrn, kop=0):
return "%i hrn. %i kop." % (hrn, kop)

print(price(8, 50))
print(price(7))
print(price(hrn=23, kop=70))
```

1.4. Основні алгоритмічні конструкції

```
try:
res = int(open('a.txt').read()) / int(open('c.txt').read())
print(res)
except IOError:

print("Input/outpet error")
except ZeroDivisionError:
print("Zero division Error")
except KeyboardInterrupt:
print("Keyboard Interrupt")
except:
print("Error")
```

1.4. Основні алгоритмічні конструкції

```
class MyError(Exception):
pass
try:
raise MyError("my error 1")
except MyError:
print("Error:")
raise
```

1.4. Основні алгоритмічні конструкції

```
a=1
b=9
c = a +b
assert c == a + b

try:
... finally:
print("The programm has been done")
```

1.5. Вбудовані типи даних

Всі дані в Python представлені об'єктами. Імена є лише посиланнями на ці об'єкти і не несуть навантаження по декларації типу. Значення вбудованих типів
мають спеціальну підтримку в синтаксисі мови: можна записати літерал рядка, числа, списку, кортежу, словника. Синтаксичну ж підтримку операцій над вбудованими типами можна легко зробити доступною і для об'єктів визначених користувачами класів.
Слід також зазначити, що об'єкти можуть бути змінними (mutable) та незмінними (immutable). Наприклад, рядки в Python є незмінними, тому операції над рядками створюють нові рядки.

1.5. Вбудовані типи даних

```
>>> type(1)
<class 'int'>
```

1.5. Вбудовані типи даних

```
>>> type(1)
<class 'int'>

>>> isinstance(1, int)
True
```

1.5. Вбудовані типи даних

```
>>> type(1)
<class 'int'>

>>> isinstance(1, int)
True

>>> 1 + 1
2
```

1.5. Вбудовані типи даних

```
>>> type(1)
<class 'int'>

>>> isinstance(1, int)
True

>>> 1 + 1
2

>>> 1 + 1.0
2.0
>>> type(2.0)
<class 'float'>
```

1.5. Вбудовані типи даних

Деякі оператори (такі, як додавання) за потреби перетворюють цілі числа в дробові. Ви можете зробити це саме самостійно.

1.5. Вбудовані типи даних

Деякі оператори (такі, як додавання) за потреби перетворюють цілі числа в дробові. Ви можете зробити це саме самостійно.

>>> float(2) 2.0

Можна явно перетворювати int у float, використовуючи функцію float()

1.5. Вбудовані типи даних

Деякі оператори (такі, як додавання) за потреби перетворюють цілі числа в дробові. Ви можете зробити це саме самостійно.

>>> float(2) 2.0

Можна явно перетворювати int у float, використовуючи функцію float()

>>> int(2.0) 2

Дробове число можна перетворити на ціле за допомогою функції int

1.5. Вбудовані типи даних

Деякі оператори (такі, як додавання) за потреби перетворюють цілі числа в дробові. Ви можете зробити це саме самостійно.

>>> float(2) 2.0

Можна явно перетворювати int у float, використовуючи функцію float()

>>> int(2.0) 2

Дробове число можна перетворити на ціле за допомогою функції int

>>> int(2.5) 2

Функція int просто відкидає дробову частину, а не округлює.

1.5. Вбудовані типи даних

>>> int(-2.5)
-2

Функція int для від’ємних повертає найменше ціле число, більше або рівне даному (тобто теж просто відкидає дробову частину). Тому не плутайте її з функцією math.floor.

>>> 1.12345678901234567890

1.1234567890123457

Десяткові дроби мають точність до 15 знаків після коми

>>> type(1000000000000000) <class 'int’>

Цілі можуть бути як завгодно великими.

1.5. Вбудовані типи даних

>>> 11 / 2
5.5

Оператор / виконує ділення чисел з плаваючою крапкою. Він повертає float, навіть якщо чисельник та знаменник цілі.

1.5. Вбудовані типи даних

>>> 11 / 2
5.5

Оператор / виконує ділення чисел з плаваючою крапкою. Він повертає float,
навіть якщо чисельник та знаменник цілі.

>>> 11 // 2
5

Оператор // виконує цілочисельне ділення. Коли результат додатній, ви можете вважати його відкиданням (не округленням) дробової частини звичайного ділення, але будьте обережними з цим.

1.5. Вбудовані типи даних

>>> −11 // 2
−6

При цілочисельному діленні від’ємних чисел оператор // округлює "вгору" до найближчого цілого. Хоча, говорячи формально, він округлює вниз, тому що -6 менше за -5.

>>> 11.0 // 2
5.0

Оператор // не завжди повертає цілі. Якщо чисельник чи знаменник дробові, // повертає float, яке, щоправда, все одно округлене до найближчого цілого.

1.5. Вбудовані типи даних

>>> −11 // 2
−6
При цілочисельному діленні від’ємних чисел оператор // округлює "вгору" до найближчого цілого. Хоча, говорячи формально, він округлює вниз, тому що -6 менше за -5.

>>> 11.0 // 2
5.0
Оператор // не завжди повертає цілі. Якщо чисельник чи знаменник дробові, // повертає float, яке, щоправда, все одно округлене до найближчого цілого.

>>> 11 ** 2
121
Оператор ** означає піднесення до степеня. 11^2=121.

>>> 11 % 2
1
Оператор % повертає остачу від цілочисельного ділення.

1.5. Вбудовані типи даних

```
for i in (False, True):
for j in (False, True):
print( i, j, ":", i and j, i or j, not i)
```

1.5. Вбудовані типи даних

```
for i in (False, True):
for j in (False, True):
print( i, j, ":", i and j, i or j, not i)
```

Результат:
False False : False False True
False True : False True True
True False : False True False
True True : True True False

1.5. Вбудовані типи даних

Python рядки бувають двох типів: звичайні і Unicode-рядка. Фактично рядок - це послідовність символів (в разі звичайних рядків можна сказати "послідовність байтів"). Рядки-константи можна задати в програмі за допомогою строкових літералів. Для литералів нарівні використовуються як апострофи ('), так і звичайні подвійні лапки ("). Для багаторядкових литералів можна використовувати потроєння апострофи або потроєння лапки. Керуючі послідовності всередині строкових літералів задаються зворотною косою межею (\). Приклади написання строкових літералів:

1.5. Вбудовані типи даних

Python рядки бувають двох типів: звичайні і Unicode-рядка. Фактично рядок - це послідовність символів (в разі звичайних рядків можна сказати "послідовність байтів"). Рядки-константи можна задати в програмі за допомогою строкових літералів. Для литералів нарівні використовуються як апострофи ('), так і звичайні подвійні лапки ("). Для багаторядкових литералів можна використовувати потроєння апострофи або потроєння лапки. Керуючі послідовності всередині строкових літералів задаються зворотною косою межею (\). Приклади написання строкових літералів:

```
s1 = "строка1"
s2 = 'строка2\nз переводом строки на наступний рядок'
s3 = """строка3
переводом строки на наступний рядок """
u1 = u'\u043f\u0440\u0438\u0432\u0435\u0442'
u2 = u'Ще один приклад
```

1.5. Вбудовані типи даних

Операції над рядками включають конкатенацію "+", повтор "*", форматування "%". Також рядки мають велику кількість методів, деякі з яких наведено нижче. Повний набір методів (та їх необов'язкових аргументів) можна отримати в документації по Python.

1.5. Вбудовані типи даних

Операції над рядками включають конкатенацію "+", повтор "*", форматування "%". Також рядки мають велику кількість методів, деякі з яких наведено нижче. Повний набір методів (та їх необов'язкових аргументів) можна отримати в документації по Python.

```
>>> "A" + "B"
'AB’

>>> "%s %i" % ("abc", 12)
'abc 12'
>>> "A"*10
'AAAAAAAAAA'
```

1.5. Вбудовані типи даних

Для представлення константної послідовності (різнорідних) об'єктів використовується тип кортеж. Літерал кортежу зазвичай записується в круглих дужках, але можна, якщо не виникають неоднозначності, писати та без них. Приклади запису кортежів:

1.5. Вбудовані типи даних

Для представлення константної послідовності (різнорідних) об'єктів використовується тип кортеж. Літерал кортежу зазвичай записується в круглих дужках, але можна, якщо не виникають неоднозначності, писати та без них. Приклади запису кортежів:

p = (1.2, 3.4, 0.9)
for s in "one", "two", "three":
print (s)
one_item = (1,)
empty = ()
p1 = 1, 3, 9 # без дужок
p2 = 3, 8, 5, # кома в кінці кортежу ігнорується

Використовувати синтаксис кортежів можна і в оператораприсвоювання. У цьому випадку на основі обчислених справа значень формується кортеж та зв'язується один в один з іменами в лівій частині.

лівій
частині

a, b = b, a

1.5. Вбудовані типи даних

В Python немає масивів з довільним типом елемента. Замість них використовуються списки. Їх можна задати за допомогою літералів, що записуються в квадратних дужках, або за допомогою спискових включень.

lst1 = [1, 2, 3,]
lst2 = [x**2 for x in range(10) if x % 2 == 1]
lst3 = list("abcde")

Таблиця 1.1 – операції та методи для роботи з послідовностями

Операція	Пояснення
len(s)	Довжина послідовності s
x in s (x not in s)	Перевірка приналежності елемента послідовності. Повертає True або False
s + s1	Конкатенація послідовностей s та s1
s*n, n*s	Послідовність з n раз повтореної послідовності s. Якщо n <0, повертається порожня послідовність
s[i]	Повертає i -й елемент s або len (s) -i -й, якщо i <0
s[i:j:d]	Зріз з послідовності s від i до j з кроком d
min(s)	Найменший елемент s
max(s)	Найбільший елемент s
s[i] = x	i -й елемент списку s замінюється на x
s[i:j:d] = t	Зріз від i до j (з кроком d) замінюється на (список) t
del s[i:j:d]	Видалення елементів зрізу з послідовності

Таблиця 1.2 – операції та методи для роботи зі змінними послідовностями

Метод	Пояснення
append(x)	Додає елемент в кінець послідовності
count(x)	Рахує кількість елементів, рівних x
extend(s)	Додає в кінець даної послідовності послідовність s
index(x)	Повертає найменше i, при якому s [i] == x.
insert(i, x)	Вставляє елемент x в i -й елемент
pop(i)	Повертає i -й елемент, видаляючи його з послідовності
reverse(s)	Змінює порядок елементів s на зворотний
sort([cmpfunc])	Сортує елементи s за заданою функцією cmpfunc

1.5. Вбудовані типи даних

Словник (хеш, асоціативний масив) - це змінна структура даних для зберігання пар ключ-значення, де значення однозначно визначається ключем. Ключем може виступати незмінний тип даних (число, рядок, кортеж і т.п.). Порядок пар ключ-значення довільний. Нижче наведено літерал для словника і приклад роботи зі словником:

1.5. Вбудовані типи даних

Словник (хеш, асоціативний масив) - це змінна структура даних для зберігання пар ключ-значення, де значення однозначно визначається ключем. Ключем може виступати незмінний тип даних (число, рядок, кортеж і т.п.). Порядок пар ключ-значення довільний. Нижче наведено літерал для словника і приклад роботи зі словником:

```
d = {1: 'one', 2: 'two', 3: 'three', 4: 'four'}
d0 = {0: 'zero'}
print(d[1]) # береться значення по ключу
d[0] = 0 # присвоюється значення по ключу
del d[0] # видаляється пара ключ-значення з даними ключем print(d)
for key& val in d|items()^ # цикл по всьому словнику
print (key, val)
for key in d.keys(): # цикл по ключам словника
print (key, d[key])
for val in d.values(): # цикл по значенням словника
print (val)
d.update(d0) # доповнення словника іншим словником
print (len(d)) # виводить кількість елементів ключ–значення
```

1.5. Вбудовані типи даних

Об'єкти цього типу призначені для роботи із зовнішніми даними. В простому випадку - це файл на диску. Файлові об'єкти повинні підтримувати основні методи: read(), write(), readline(), readlines(), seek(), tell(), close() і т.д.
Наступний приклад показує копіювання файлу:

```
f1 = open("file1.txt", "r")
f2 = open("file2.txt", "w")
for line in f1.readlines():
f2.write(line)
f2.close()
f1.close()
```

1.5. Вбудовані типи даних

Варто зауважити, що крім власне файлів в Python використовуються і файлоподібні об'єкти. В дуже багатьох функціях просто неважливо, переданий їй об'єкт типу file або іншого типу, якщо він має всі ті ж методи. Наприклад, копіювання вмісту за посиланням (URL) в файл file2.txt можна досягти, якщо замінити перший рядок на:

import urllib
f1 = urllib.urlopen("https://python.org")

1.6. Вирази

lambda	лямбда-вираз
or	логічне АБО
and	логічне І
not x	логічне НІ
in, not in	перевірка приналежності
is, is not	перевірка ідентичності
<,<=,>,>=,!=,==	порівняння
|	побітовое АБО
^	побітовое виключаюче АБО
&	побітовое І
<<, >>	побітові зсуви
*, /, %	додавання і віднімання
+, -	множення, ділення, залишок
+x, -x	унарний плюс і зміна знака
~x	побітовое НЕ
**	піднесення до степеня
x.атрибут	посилання на атрибут
x [індекс]	взяття елемента за індексом
x [від: до]	виділення зрізу (від і до)
f (аргумент, ...)	визов функції
(...)	дужки або кортеж
[..]	список або спискове включення
{кл: зн ...}	словник пар ключ–значення
`вираз`	перетворення до рядка (repr)

1.7. Імена

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class',
'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for',
'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal',
'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

Практична робота

Відповідно до свого номеру по списку реалізувати на мові програмування Python наступні функції, які вказані в таблиці

Практична робота

№	$f(x)$	h; $[a; b]$
1	2	3
1	$y=\ln(x)$	h=0.1; a=1; b=1.5
2	$y=1+\ln^2(x)$	h=0.1; a=0.4; b=1.0
3	$y=1+e^x$	h=0.01; a=0.5; b=0.6
4	$y=e^{x^2}/2$	h=0.2; a=2; b=3
5	$y=\cos(x)\cdot e^{-x}$	h=0.2; a=1; b=2
6	$y=1/(1+e^{-x})$	h=0.2; a=3; b=4
7	$y=\sin(x)\cdot\sinh(x)$	h=1; a=1; b=5
8	$y=0.5+\sinh^2(x)$	h=0.2; a=2; b=3
9	$y=\sqrt{x}\cdot\cosh(x)$	h=0.2; a=3; b=4
10	$y=1/(1+\cosh^2(x))$	h=0.5; a=2; b=4
11	$y=\sqrt{x}\cdot\sinh(x)$	h=1; a=1; b=5
12	$y=e^{-x}\cdot\cosh(x)$	h=1; a=1; b=4
13	$y=\ln(x^2)$	h=0.1; a=1; b=1.4
14	$y=x+\ln(x)$	h=1; a=1; b=5
15	$y=1/(1+\sin(x))$	$h=\pi/10$; $a==\pi/6$; $b==\pi/3$
16	$y=\sin(x)+\sqrt{x}$	$h=\pi/10$; $a==\pi/6$; $b==\pi/4$
17	$y=x\cdot(1-\cos(x))$	h=0.1; a=0.4; b=0.8
18	$y=e^{x+3}\sin(x)$	h=0.5; a=0; b=2
19	$y=\cos(x)\cdot\cosh(x)$	h=1; a=1; b=5
20	$y=e^{x+1}\cdot\sinh(x)$	h=1; a=1; b=4
21	$y=10^{-2}(5+4x)-e^{x^3+4}$	h=0.1; a=-3.4; b=-1.4
22	$y=4x^3+2^{5/4}xe^{-x}$	h=1.01; a=2.4; b=10.4
23	$y=9(x^3+3.2)\cdot\mathrm{tg}(x)$	h=0.2; a=1; b=2.4
24	$y=1.2e^{x^2}+x$	h=-0.05; a=-0.75; b=-1.5
25	$y=x^{2^2}+\cos\left(2^{3/4}+x^{3/2}\right)$	$h==\pi/3$; a=14; b=19
26	$y=\left(x^{5/2}-0.8\right)\cdot\ln\left(x^2+12.7\right)$	h=0.3; a=0.25; b=5
27	$y=0.8\cdot10^{-5}(x^3+6.7)^{7/6}$	h=0.1; a=-0.5; b=0.4
28	$y=0.4+x^{2/3}\cos(x+e^x)$	$h=\pi/10$; a=5.6; b=15.4

Висновок

В даному розділі були розглянуті основи мови програмування Python 3, а саме: типи даних, основні структури даних, стандартні методи, наведені приклади створення власних функцій та роботи з файлами.

Питання для самоконтролю

1. Що із себе представляє мова програмування Python?
2. Які типи даних в мові Python ви знаєте? Наведіть приклади.
3. Які основні логічні структури присутні в Python?
4. Що таке вирази? Які вони бувають? Наведіть приклади.
5. Що таке винятки і як їх можна реалізувати в мові Python 3? 6. Яким чином можна реалізувати функцію в Python 3?

Дякую за увагу!