

Python



Что такое Python? (Пайтон или Питон)



- Python — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ.
- Разработчик: Гвидо ван Россум
- ОС: кроссплатформенность
- Сайт: python.org (англ.)
- Выпуск: 3.10.3 (16 марта 2022);
- Расширение файлов: .py
- Класс языка: объектно-ориентированный язык программирования
- Появился в: 20 февраля 1991

Числа int/float/complex



- Числа в Python ничем не отличаются от обычных чисел:
 - Целые числа (int) / 4 , 10 , -100
 - Вещественные числа (float) / 3.324 , -0.5, 4.99943
 - Комплексные числа (complex) / 5+7i, -4+4i, -5
- Они поддерживают набор самых обычных математических операций -->

$x + y$	Сложение
$x - y$	Вычитание
$x * y$	Умножение
x / y	Деление
$x // y$	Получение целой части от деления
$x \% y$	Остаток от деления
$-x$	Смена знака числа
$\text{abs}(x)$	Модуль числа
$\text{divmod}(x, y)$	Пара ($x // y$, $x \% y$)
$x ** y$	Возведение в степень
$\text{pow}(x, y[, z])$	x^y по модулю (если модуль задан)

Списки list



- Списки в Python - упорядоченные изменяемые коллекции объектов произвольных типов

```
a1 = [1, 2, 3, 4, 5]
```

```
a2 = [10, 11, 12]
```

```
>>> a1.extend(a2)
```

```
[1, 2, 3, 4, 5, 10, 11, 12]
```

```
>>> a1.append(100)
```

```
[1, 2, 3, 4, 5, 100]
```

Метод	Что делает
<code>list.append(x)</code>	Добавляет элемент в конец списка
<code>list.extend(L)</code>	Расширяет список list, добавляя в конец все элементы списка L
<code>list.insert(i, x)</code>	Вставляет на i-ый элемент значение x
<code>list.remove(x)</code>	Удаляет первый элемент в списке, имеющий значение x. ValueError, если такого элемента не существует
<code>list.pop([i])</code>	Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент

Строки str



- Строки в Python - упорядоченные последовательности символов, используемые для хранения и представления текстовой информации, поэтому с помощью строк можно работать со всем, что может быть представлено в текстовой форме.

```
a1 = "Строка номер 1"  
a2 = "Скоро экзамены"
```

```
>>> a1 + a2  
"Строка номер 1 Скоро экзамены "
```

```
>>> a1[3]  
"о"
```

$S1 + S2$	Конкатенация (сложение строк)
$S1 * 3$	Повторение строки
$S[i]$	Обращение по индексу
$S[i:j:step]$	Извлечение среза
$len(S)$	Длина строки

Словари dict



- Словари в Python - (не)упорядоченные коллекции произвольных объектов с доступом по ключу.

```
d = {'test1': 1, 'test2': 2}
```

```
>>> d['test3'] = 5  
{'test1': 1, 'test2': 2, 'test3': 5}
```

```
>>> d['test1']  
1
```

```
>>> list(d.keys())  
['test1', 'test2', 'test3']
```

<code>dict.copy()</code>	возвращает копию словаря
<code>dict.get(key[, default])</code>	возвращает значение ключа
<code>dict.items()</code>	возвращает пары (ключ, значение).
<code>dict.keys()</code>	возвращает ключи в словаре.
<code>dict.values()</code>	возвращает значения в словаре.

Операторы сравнения



- Что такое оператор и операнды? Это можно объяснить простым примером: $10 > 5$. В этом выражении 10 и 5 — левый и правый операнды. Знак « $>$ » — оператор.

<code>==</code>	проверяет одинаково ли значение операндов, если одинаковы - то условие является истиной	<code>a == b</code>	False
<code>!=</code>	проверяет одинаково ли значение операндов, если НЕ одинаковы - то условие является истиной	<code>a != b</code>	True
<code>></code>	проверяет значение левого операнда, если он больше, чем правый - то условие является истиной	<code>a > b</code>	True
<code><</code>	проверяет значение левого операнда, если он меньше, чем правый - то условие является истиной	<code>a < b</code>	False
<code>>=</code>	проверяет значение левого операнда, если он больше или равен правому - то условие является истиной	<code>a >= b</code>	True
<code><=</code>	проверяет значение левого операнда, если он меньше либо равен правому - то условие является истиной	<code>a <= b</code>	False

Цикл for



- В цикле for указывается переменная и множество значений, по которому будет пробегать переменная

```
i = 1
```

```
for color in ['red', 'orange', 'yellow', 'green']:  
    print('#', i, 'цвет', color)  
    i = i + 1
```

```
for value in range(4): # равносильно инструкции for i in 0, 1, 2, 3:  
    print(i)
```


Цикл while



- Цикл `while` (“пока”) позволяет выполнить одну и ту же последовательность действий, пока проверяемое условие истинно. Условие записывается до тела цикла и проверяется до выполнения тела цикла. *Как правило, цикл `while` используется, когда невозможно определить точное значение количества проходов исполнения цикла.*

```
i = 1
while i <= 10:
    print(i ** 2)
    i += 1
```

Задача 1



- 1.1 Дан список чисел, необходимо вывести на экран только те, которые больше 5, но меньше 10 включительно
- Вход: [-1, -20, 18, 0, 5, 10] Выход: ???
- 1.2 Дан словарь содержащий 1 ключ и его значение. Расширить его путем добавления 3х ключей со своими значениями. Вывести список всех ключей в словаре, и список всех значений.
- Вход: {'ключ1' : 100}

Задача



- Условие:
- Пользователь вводит число N (размерность квадратной матрицы). Программа (или функция) должна заполнить эту матрицу «методом змейки*» числами от 0 до $N*N-1$

- *- метод змейки - это заполнение матрицы в виде:

0 (0,0)	7 (0,1)	6 (0,2)
1 (1,0)	8 (1,1)	5 (1,2)
2 (2,0)	3 (2,1)	4 (2,2)

- Пример:
- Вход: 3
- Выход: $\{(0, 0): 0, (1, 0): 1, (2, 0): 2, (2, 1): 3, (2, 2): 4, (1, 2): 5, (0, 2): 6, (0, 1): 7, (1, 1): 8\}$

Решение задачи 2



Зададим необходимые переменные. Которые нам пригодятся для выполнения программы

```
# словарь (матрица) в которую мы будем добавлять индексы и значения
matrix = {}
```

```
# количество элементов, которое необходимо заполнить
count_of_items = n * n
```

```
# максимальное и минимальное значение индексов
min_i, max_i = 0, n
min_j, max_j = 0, n
```

```
# первое значение
count = 0
```

```
# на основе этих переменных будем определять, по какому столбцу или строке мы заполняем матрицу
current_j, current_i = 0, 0
```

```
# Запустим цикл for по первой строке от начального до конечного индекса i
```

```
for index in range(min_i, max_i):
    #добавим элемент в матрицу с индексами (i,j)
    matrix[index, current_j] = count
    # увеличим значение на 1, т.к. в следующий раз нам нужно будет следующее число
    count += 1
```

```
# после того, как мы заполнили левый столбик, в качестве строки по которой мы будем идти будет последний ( т.е. Index)
```

```
current_i = index
```

```
# увеличим минимальный столбец на 1, чтобы больше не заполнять первый столбец
min_j += 1
```

0 (0,0)	7 (0,1)	6 (0,2)
1 (1,0)	8 (1,1)	5 (1,2)
2 (2,0)	3 (2,1)	4 (2,2)

Решение задачи 2



```
for index in range(min_j, max_j):  
    matrix[current_i, index] = count  
    count += 1  
current_j = index  
max_i -= 1
```

```
for index in range(max_i - 1, min_i - 1, -1):  
    matrix[index, current_j] = count  
    count += 1  
current_i = index  
max_j -= 1
```

```
for index in range(max_j - 1, min_j - 1, -1):  
    matrix[current_i, index] = count  
    count += 1  
current_j = index  
min_i += 1
```

1 (0,0)	2 (0,1)	3 (0,2)
8 (1,0)	9 (1,1)	4 (1,2)
7 (2,0)	6 (2,1)	5 (2,2)

1 (0,0)	2 (0,1)	3 (0,2)
8 (1,0)	9 (1,1)	4 (1,2)
7 (2,0)	6 (2,1)	5 (2,2)

1 (0,0)	2 (0,1)	3 (0,2)
8 (1,0)	9 (1,1)	4 (1,2)
7 (2,0)	6 (2,1)	5 (2,2)

Решение задачи 2



```
matrix = {}

count_of_items = n * n
min_i, max_i = 0, n
min_j, max_j = 0, n
count = 0
current_j, current_i = 0, 0
```

```
while count < count_of_items:
    for index in range(min_i, max_i):
        matrix[index, current_j] = count
        count += 1
    current_i = index
    min_j += 1
    for index in range(min_j, max_j):
        matrix[current_i, index] = count
        count += 1
    current_j = index
    max_i -= 1
    for index in range(max_i - 1, min_i - 1, -1):
        matrix[index, current_j] = count
        count += 1
    current_i = index
    max_j -= 1
    for index in range(max_j - 1, min_j - 1, -1):
        matrix[current_i, index] = count
        count += 1
    current_j = index
    min_i += 1
```

```
{(0, 0): 0, (1, 0): 1, (2, 0): 2, (2, 1): 3, (2, 2): 4, (1, 2): 5, (0, 2): 6, (0, 1): 7, (1, 1): 8}
```