BRAIN SKILLS

Основы Python: введение, условные конструкции, циклы

Макар Басалаев

Почему программисты используют Python?

- Удобочитаемость, ясность кода
- Меньше кода (чем в C++, Java) => быстрее отладка, сопровождение, запуск (нет компиляции)
- Переносимость программ (Linux, Windows)
- Большое число библиотек

Кто в наше время использует Python?

- Компания Google широко использует Python в своей поисковой системе и оплачивает труд создателя Python
- BitTorrent написана на языке Python
- Такие компании, как Intel, Cisco и IBM, используют Python для тестирования аппаратного обеспечения
- Pixar использует Python в производстве анимационных фильмов
- JPMorgan Chase применяет Python для прогнозирования финансового рынка
- NASA, Fermilab и другие используют Python для научных вычислений
- NSA использует Python для шифрования и анализа разведданных

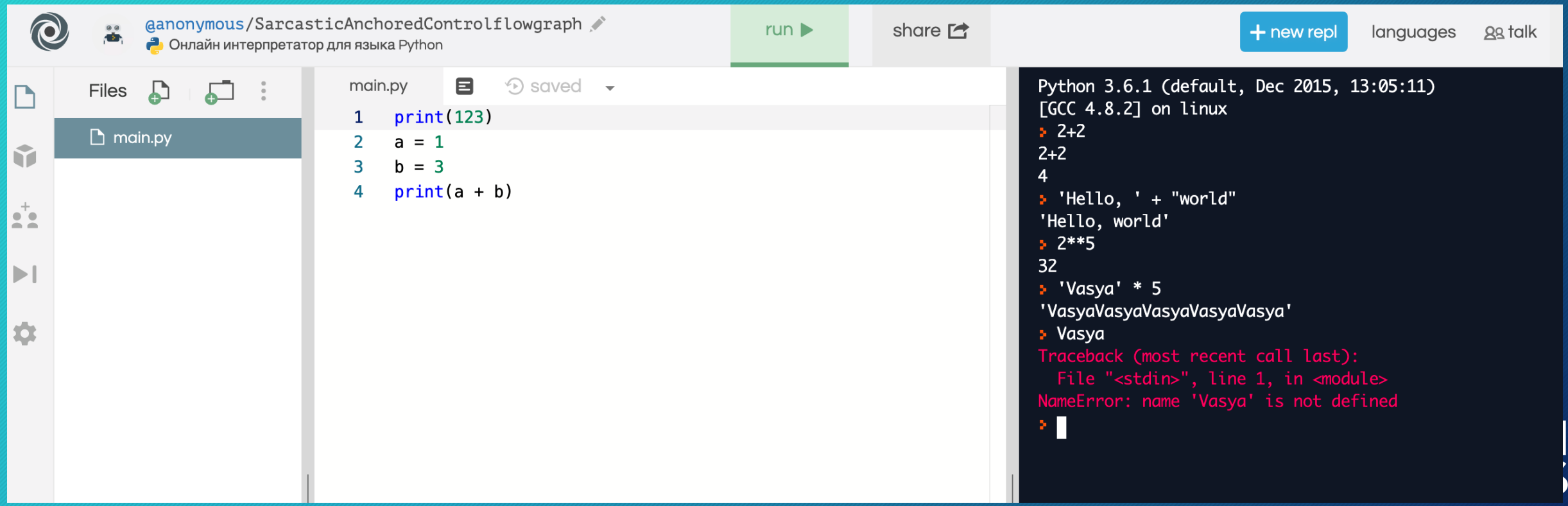
Интерпретатор Python

- Интерпретатор – это такой модуль, который исполняет другие программы. Интерпретатор Python читает программу и выполняет составляющие ее инструкции.

- В интерпретаторе Python (программе IDLE) можно работать в режиме «запрос-ответ»
- Мы задаем Python’у вопрос, а он нам отвечает на него

Работа с интерпретатором Python

- https://repl.it/languages/python3



Работа с интерпретатором Python

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
> 2+2
2+2
4
> 'Hello, ' + "world"
'Hello, world'
> 2**5
32
> 'Vasya' * 5
'VasyaVasyaVasyaVasyaVasya'
> Vasya
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Vasya' is not defined
>
```

BRAIN SKILLS

Работа с интерпретатором Python

- В первом примере мы попросили Python сложить два числа
- Во втором – склеить две строки
- В третьем – возвести 2 в степень 5
- В четвертом – повторить строку 5 раз
- А пятый пример вызвал ошибку (она выделяется красным). Это означает, что Python не может «ответить» на ваш вопрос.
  - В данном случае он просто не знает такое имя **Vasya**

Работа с интерпретатором Python

- Python – язык с **динамической** типизацией
  - Нет надобности описывать переменные
  - Переменная создается в тот момент, когда в программе она первый раз используется.
  - В этот же момент определяется тип переменной
  - Текущий тип переменной или иного объекта всегда можно узнать с помощью команды **type**

```
>>> type(2)
<class 'int'>
>>> type(2.5)
<class 'float'>
>>> type('Vasya')
<class 'str'>
```

BRAIN SKILLS

Имена в Python

- Python различает строчные и прописные символы
- В Python переменная может иметь **любое** имя, однако оно не должно начинаться с цифры и знака препинания, но может начинаться с _, а также оно не должно совпадать с ключевыми словами Python
  - Variable - корректное имя
  - _index - корректное имя
  - Петя - корректное имя
  - !perem - некорректное имя
  - def - некорректное имя

Регистры в Python

- **Python** – регистрозависимый язык
- Vasya и vasya – это разные имена

- **НЕ ЗАБЫВАЙТЕ ЭТОТ ФАКТ**

BRAIN SKILLS

Типы в Python

Таким образом, в Python существуют следующие простые типы или классы:

- Целый числа (***int***)
- Вещественные числа (***float***)
- Строки (***str***) – указываются в одинарных или двойных кавычках
  - ‘string #1’
  - “string #2”
  - ~~‘string #3”~~

BRAIN SKILLS

Преобразование простых типов

- Целое число можно преобразовать в строку
  - str(5)
- Вещественное число можно преобразовать в строку
  - str(3.1415)
- Строку можно преобразовать в целое число
  - int(“5”)
- Строку можно преобразовать в вещественное число
  - float(“5.13”)

Первая программа на Python

```
a = 5
b = 3
c = a + b
print(c)
```

Рассмотрим эту программу построчно:

1. Определяется переменная **a** и в нее заносится значение 5
2. Определяется переменная **b** и в нее заносится значение 3
3. Определяется переменная **c** и в нее заносится сумма значений переменных **a** и **b**
4. Печатается значение переменной **c**

Операции с целыми числами в Python

A + B	– сложение
A – B	– вычитание
A * B	– умножение
A / B	– деление
A**B	– возведение в степень
-A	– «унарный» минус (отрицание)
A // B	– целочисленное деление (частное)
A % B	– остаток от деления

BRAIN SKILLS

Операции со строками в Python

S1 + S2 – склейка строк

S*n – повторить строку S n-раз

Вывод данных в Python

- Для вывода данных в Python применяется операция **print** (это мы уже видели)

```
>>> a = 1
>>> b = 2
>>> print(a, '+', b, '=', a+b)
1 + 2 = 3
```

- По умолчанию, **print** разделяет выводимые значения **пробелами**
- Для управления разделителем служит параметр **sep**

```
>>> print(a, '+', b, '=', a+b, sep="")
1+2=3
```

BRAIN SKILLS

Вывод данных в Python

- print так же по умолчанию переводит строку

```
>>> print('123')
...   print('456')
123
456
```

- Для управления переводом строки служит параметр end

```
>>> print('123', end="")
...   print('456', end="")
123456
```

BRAIN SKILLS

Ввод данных в Python

- Для организации ввода используется операция **input**
- *НО!* Результат ввода – **это всегда СТРОКА**

```
>>> input()
Привет
'Привет'
>>>
```

BRAIN SKILLS

Ввод данных в Python

- У оператора **input** может быть параметр – приглашение

```
>>> input('Введите свое имя: ')
Введите свое имя: Вася
'Вася'
>>>
```

Ввод данных в Python

- Но что же делать, если мы хотим ввести с клавиатуры число?
- Воспользуемся преобразованием типов

```
a = float(input('Введите число: '))
n = int(input('Введите показатель: '))
print('Result is', a**n)
```

BRAIN SKILLS

Операция присваивания (выводы)

- Синтаксис операции присваивания очень прост: имя переменной = значение
- После операции переменная может поменять не только свое значение, но и тип

```
a = float(input('Введите число: '))
print(a) # a – вещественное число
a = input('Введите число: ')
print(a) # a – строка (предыдущее
значение потеряно)
```

- Сначала вычисляется значение из правой части, потом оно заносится в переменную
- # - значок комментария (от него и до конца строки текст будет проигнорирован)

Условные конструкции

Условные конструкции

- Синтаксис условного оператора if требует писать код аккуратно

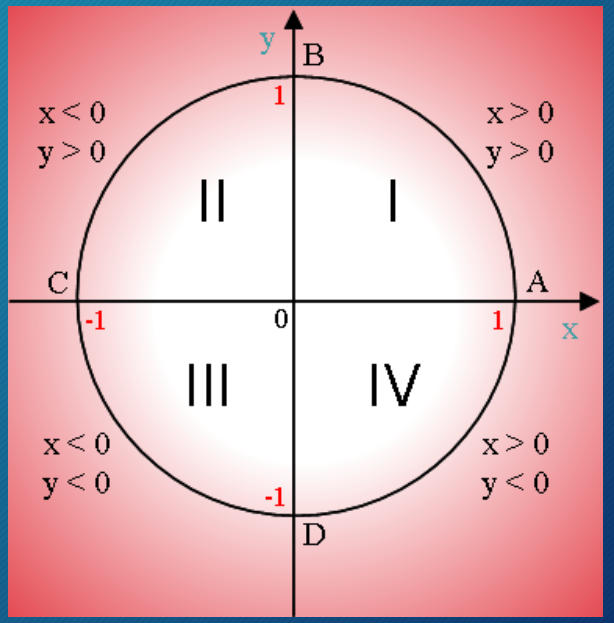
```
x = int(input())
if x > 0:
    print(x)
else:
    print(-x)
```

- Следует обратить внимание на отступы.
- Последовательность операций, имеющих одинаковый непрерывный отступ относятся к одному блоку

Условные конструкции (пример №1)

Напишем программу, определяющую четверть, в которой находится точка с координатами x и y

```
x = int(input())
y = int(input())
if x > 0:
    if y > 0:                    # x>0, y>0
        print("Первая четверть")
    else:                        # x>0, y<=0
        print("Четвертая четверть")
else:
    if y > 0:                    # x<=0, y>0
        print("Вторая четверть")
    else:                        # x<=0, y<=0
        print("Третья четверть")
```



BRAIN SKILLS

Операции отношения

Обозначение	Суть	Пример
<	меньше	a < b
>	больше	a > b
<=	меньше равно	a <= b
>=	больше равно	a >= b
==	равно ли	a == b
!=	не равно	a != b

Результат операции отношения – это или **истина** (True) или **ложь** (False).

Логические операции (связки)

- Для объединения нескольких операций отношения в одну применяются логические связки:
  - not
  - and
  - or

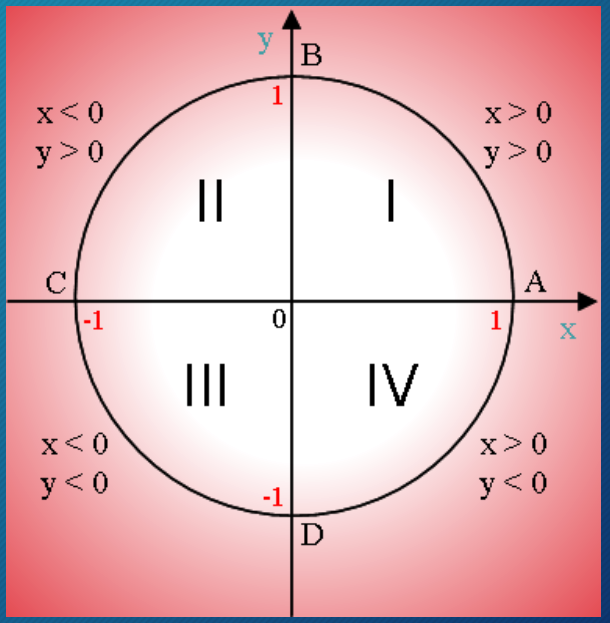
```
if (a > b) and (a <= 10):
    print(a)
if (a == 100) or (b == 0):
    a = 50
if not(a == 5):
    print("Ошибка")
```

BRAIN SKILLS

Условные конструкции (пример №2)

Напишем иначе программу, определяющую четверть, в которой находится точка с координатами **x** и **y**

```
if (x > 0) and (y > 0):
     print ('1 четверть')
else:
     if (x < 0) and (y > 0):
          print ('2 четверть')
else:
     if (x < 0) and (y < 0):
          print ('3 четверть')
else:
     if (x > 0) and (y < 0):
          print ('4 четверть')
else:
    print ('Оси')
```

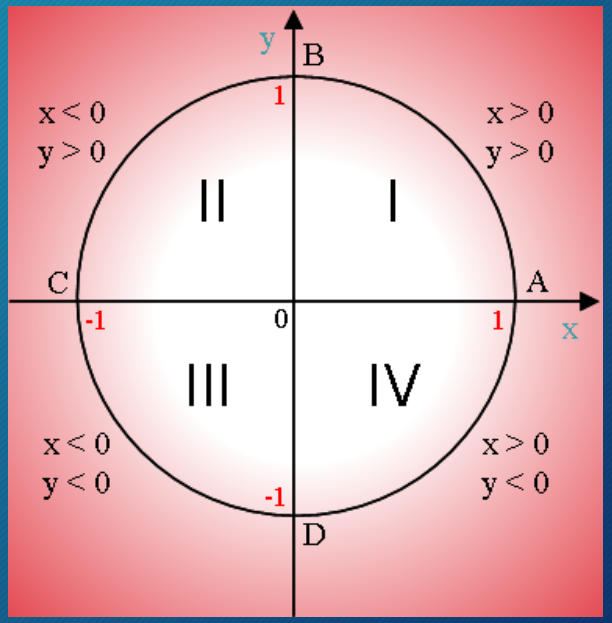


BRAIN SKILLS

Условные конструкции (пример №3)

А теперь упростим эту громоздкую запись:

```
if (x > 0) and (y > 0):
    print ('1 четверть')
elif (x < 0) and (y > 0):
    print ('2 четверть')
elif (x < 0) and (y < 0):
    print ('3 четверть')
elif (x > 0) and (y < 0):
    print ('4 четверть')
else:
    print ('Оси')
```



BRAIN SKILLS

Синтаксический сахар (операции отношения)

Синтаксический сахар - это конструкция, которая никак не влияет на работу программы, но делает ее более удобной для программиста

```
x = 5
if (x > 0) and (x < 10):
    print('По старинке')
```

```
x = 5
if (10 > x > 0):
    print ('По новому')
```

BRAIN SKILLS

Цикл с предусловием

- Цикл – это набор операций, которые повторяются необходимое количество раз.
- Это количество может быть определено условием, а может быть задано константно.
- Синтаксис цикла с предусловием:

```
while <условие>:
    операция №1
    операция №2
    …
```

- Цикл будет продолжаться пока условие **истинно**
- Опять обратите внимание на отступы

BRAIN SKILLS

Цикл с предусловием (пример №0)

- Подсчет квадратов первых 10 чисел:

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

Цикл с предусловием (пример №1)

- Подсчет количества цифр целого числа

```
n = int(input())
k = 0
while (n > 0):
    k = k + 1
    n = n // 10
print(k)
```

BRAIN SKILLS

Цикл с предусловием (пример №2)

- Вычисление факториала целого числа

$$n! = \begin{cases} 1, n = 0 \\ n * (n-1) * (n-2) * \cdots * 1, n > 0 \end{cases}$$

```
n = int(input())
f = 1
while (n > 0):
    f = f * n
    n = n - 1
print(f)
```

BRAIN SKILLS

Break

- Оператор **break** – мгновенный выход из цикла
- Задача: ввести числа до нуля, посчитать сумму положительных чисел. Как только ввели отрицательное число – закончить программу

```
a = int(input())
s = 0
while a != 0:
    if a < 0:
        break
    s += a
    a = int(input())
```

BRAIN SKILLS

Continue

- Оператор **continue** – продолжение цикла сверху
- Задача: ввести числа до нуля, посчитать сумму положительных чисел

```
a = int(input())
s = 0
while a != 0:
    if a < 0:
        continue
    s += a
    a = int(input())
```

BRAIN SKILLS

Синтаксический сахар (арифметические операции)

```
a = a + 1      →      a += 1
a = a - x      →      a -= x
n = n // 3     →      n //= 3
```

```
x = x OP y     →      x OP= y
```

Диапазоны

- Идея цикла с параметром (**for**) в Python заключается в том, что мы перебираем значения из некоторого пула, который может быть представлен **диапазоном**
- **Диапазон** – новый для нас тип данных языка
- **Диапазоны** могут быть созданы оператором **range**

Запись и примеры диапазонов

Запись	Пояснение
range(n)	Целые числа в интервале [0..n)
range(a, b)	Целые числа в интервале [a..b)
range(a, b, s)	Целые числа в интервале [a..b) с шагом s

Запись	Пояснение
range(5)	0, 1, 2, 3, 4
range(2, 5)	2, 3, 4
range(-3, 7, 2)	-3, -1, 1, 3, 5
range(10, 1, -2)	10, 8, 6, 4, 2
range(2, 5, -1)	ПУСТО

Цикл FOR

- Цикл for обходит значения из предлагаемого диапазона
- Следующий пример выводит через пробел целые числа от 0 до 9

```
for i in range(10):
    print(i, end=" ")
```

- Переменная i последовательно примет все значения из диапазона
- Внутри цикла может находиться много операций. Все они должны иметь одинаковый отступ

BRAIN SKILLS

Примеры цикла FOR

```
>>> n = int(input())
>>> for i in range(n):
…         print(i, end=' ')
0 1 2 3 4


>>> for i in range(2, 70, 7):
…         print(i, end=' ')
2 9 16 23 30 37 44 51 58 65
```

BRAIN SKILLS

«Красивая» программа. Факториал

```
n = int(input("Введите число: "))
f = 1
for i in range(1, n+1):
    f *= i
print("Факториал числа", n, "равен", f)
```

BRAIN SKILLS

Проход по значениям

```
>>> for i in 1, 2, 3, 'one', 'two', 'three':
…       print(i)
1
2
3
'one'
'two'
'three’
```

BRAIN SKILLS

Приоритеты операций над числами (по возрастанию)

Операция	Смысл
x + y	Сумма
x - y	Разность
x * y	Произведение
x / y	Деление
x // y	Деление нацело
x % y	Остаток
-x	Унарный минус
+x	Унарный плюс
x ** y	Возведение в степень

BRAIN SKILLS

Вопросы?