Знакомство с языком программирования Python

Основы основ и начало начал

Почему Python?

Флешбеки...

Существует множество языков программирования

30 из них наиболее популярны

Почему Python?

Высокоуровневый
Первое упоминание в 1991 v3.9

Легчайший синтаксис
Невысокий порог вхождения
Востребованный на рынке
Множество библиотек (базовых хватает)
Популярен и деньгу платят

Почему Python? [help]

Кроссплатформенность

Подходит для:

- Веб-сервисов
- ML, DataSciece, Аналитики
- *Игр*
- *Написания софта*

Интерпретируемый

Почему Python?

Первые шаги написания программы

Пресловутый “Hello World”

Скуууучнаааа :(

Помните зачем это?

Основы основ

Как установить

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

Основы основ

Как установить

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

иногда на mac
alias python='python3'

Переменные

Переменные

Типы данных справедливы
int, float, boolean, str
list и др.

Python – язык с динамической типизацией

```
value = 2809
name = 'Sergey'
```

Переменные

У Python есть одна проблема

Неверно поставленный пробел сломает вашу программу

Ввод и вывод данных
Преобразование типов

Ввод и вывод данных

print() – отвечает за вывод данных

input() – отвечает за ввод данных

Ввод и вывод данных

```
print('Введите a');
a = input()
print('Введите b');
b = input()
print(a, b)
print('{} -- {}'.fotmat(a, b))
```

Ввод и вывод данных

```
print('Введите a');
a = input()                    # 10
print('Введите b');
b = input()                    # 20
c = 30
print(a, ' + ', b, ' = ', c)
```

Ввод и вывод данных

```
print('Введите a');
a = input()                    # 10
print('Введите b');
b = input()                    # 20
c = 30
print(a, ' + ', b, ' = ', c)   # 10  +  20  =  30
```

Ввод и вывод данных

```
print('Введите a');
a = int(input())
print('Введите b');
b = int(input())
c = 30
print(a, ' + ', b, ' = ', c)
print('{} + {} = {}'.format(a, b, c))
```

Ввод и вывод данных

```
print('Введите a');
a = float(input())
print('Введите b');
b = float(input())
c = ...
print(a, ' + ', b, ' = ', c)
```

Ввод и вывод данных

```
a = int(input('Введите a: '))          # 11
b = int(input('Введите b: '))          # 22
c = 33
print('{} + {} = {}'.format(a, b, c))
```

Ввод и вывод данных

```
a = int(input('Введите a: '))          # 11
b = int(input('Введите b: '))          # 22
c = 33
print('{} + {} = {}'.format(a, b, c))  # 11 + 22 = 33
```

Ввод и вывод данных

```
a = int(input('Введите \na: '))
b = int(input('Введите \nb: '))
c = a + b
print('{} + {} = {}'.format(a, b, c))
```

Демонстрация

Сегодня будет код

Арифметические операции

Арифметические операции

Важно и нужно, без них вы не напишете ни одной программы
Если помните математику – проблем не будет

+, -, *, /, %, //, **
Приоритет операций
**, ⊕, ⊖, *, /, //, %, +, -
( ) Скобки меняют приоритет

Арифметические операции

```
exp1 = 2**3 - 10 % 5 + 2*3
exp2 = 2**3 - 10 / 5 + 2*3
print(exp1)  # 14.0 или 14
print(exp2)  # 12.0 или 12
```

Арифметические операции

```
exp1 = 2**3 - 10 % 5 + 2*3
exp2 = (2**3) - (10 / 5) + (2*3)
print(exp1)  # 14.0 или 14
print(exp2)  # 12.0 или 12
```

Демонстрация

Сокращённые операции и операции присваивания

```
iter = 2
iter += 3    # iter = iter + 3
iter -= 4    # iter = iter - 4
iter *= 5    # iter = iter * 5
iter /= 5    # iter = iter / 5
iter //= 5   # iter = iter // 5
iter %= 5    # iter = iter % 5
iter **= 5   # iter = iter ** 5
```

Демонстрация

Дополнительно

```
a, b, c = 1, 2, 3
# a: 1 b: 2 c: 3
print('a: {} b: {} c: {}'.format(a, b, c))



range(*(1,5,2))
```

Логические операции

Логические операции

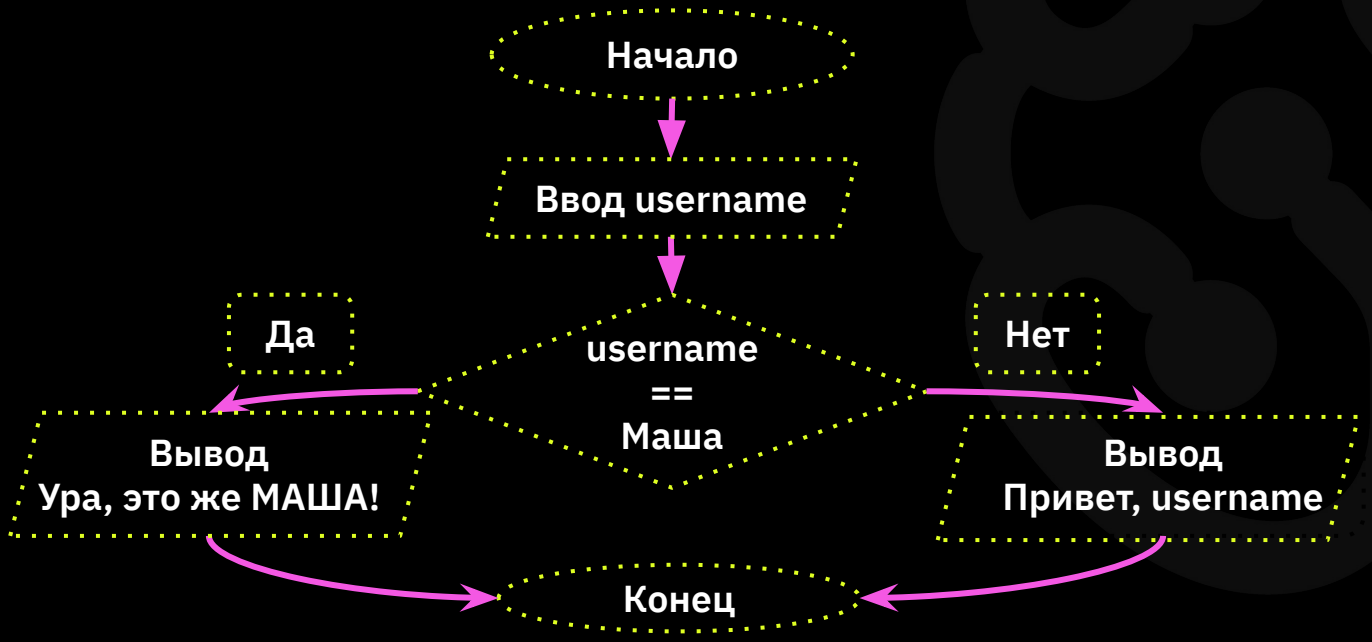
`>`, `>=`, `<`, `<=`, `==`, `!=`
`not`, `and`, `or` – не путать с `&`, `|`, `^`

Кое-что ещё: `is`, `is not`, `in`, `not in`

Управляющие конструкции: if, if-else

Управляющие конструкции: if, if-else

Условный оператор позволяет управлять ходом выполнения программы



Управляющие конструкции: if, if-else

```
if condition:
    # operator 1
    # operator 2
    # ...
    # operator n
else:
    # operator n + 1
    # operator n + 2
    # ...
    # operator n + m
```

Управляющие конструкции: if, if-else

```
if condition :
    # operator 1
    # operator 2
    # ...
    # operator n
else:
    # operator n + 1
    # operator n + 2
    # ...
    # operator n + m
```

Отступы важны!

Управляющие конструкции: if, if-else

```
username = input('Введите имя: ')
if(username == 'Маша'):
    print('Ура, это же МАША!');
else:
    print('Привет, ', username);
```

Управляющие конструкции: if, if-else

```
if condition1:
    # operator
elif condition2:
    # operator
elif condition3:
    # operator
else:
    # operator
```

Управляющие конструкции: if, if-else

```
username = input('Введите имя: ')
if username == 'Маша':
    print('Ура, это же МАША!')
elif username == 'Марина':
    print('Я так ждала Вас, Марина!')
elif username == 'Ильнар':
    print('Ильнар - топ)')
else:
    print('Привет, ', username)
```

Управляющие конструкции: while

Управляющие конструкции: while

Цикл позволяет выполнить блок операторов какое-то количество раз

```
while condition:
    # operator 1
    # operator 2
    # . . .
    # operator n
```

Управляющие конструкции: while

Цикл позволяет выполнить блок операторов какое-то количество раз

```
while condition:
    # operator 1
    # operator 2
    # . . .
    # operator n
```

Отступы важны!

Управляющие конструкции: while

```
original = 23
inverted = 0
while original != 0:
    inverted = inverted * 10 + (original % 10)
    original //= 10
print(inverted)
```

Управляющие конструкции: while-else

```
while condition:
    # operator 1
    # operator 2
    # . . .
    # operator n
else:
    # operator n + 1
    # operator n + 2
    # . . .
    # operator n + m
```

Управляющие конструкции: while-else

```
while condition:
    # operator 1
    # operator 2
    # . . .
    # operator n
else:
    # operator n + 1
    # operator n + 2
    # . . .
    # operator n + m
```

Отступы важны!

Управляющие конструкции: while-else

```
original = 23
inverted = 0
while original != 0:
    inverted = inverted * 10 + (original % 10)
    original //= 10
else:
    print('Пожалуй')
    print('хватит )')
print(inverted)
# Пожалуй
# хватит )
# 32
```

Управляющие конструкции: for

Управляющие конструкции: for

Когда мы знаем, что хотим

```
for i in enumeration:
    # operator 1
    # operator 2
    # . . .
    # operator n
```

Управляющие конструкции: for

```
for i in 1, -2, 3, 14, 5:
    print(i)


# 1
# -2
# 3
# 14
# 5
```

Управляющие конструкции: for

Знакомьтесь – range

```
r = range(5)            # range(0, 5)
r = range(2, 5)         # range(2, 5)
r = range(-5, 0)        # range(-5, 0)
r = range(1, 10, 2)     # range(1, 10, 2)
r = range(100, 0, -20)  # range(100, 0, -20)
```

Управляющие конструкции: for

Знакомьтесь – range

```
r = range(100, 0, -20) # range(100, 0, -20)

for i in r:
    print(i)
# 100 80  60 40 20

for i in range(5):
    print(i)
# 0 1 2 3 4
```

Управляющие конструкции: for

Вложенные циклы

```
line = ""
for i in range(5):
    line = ""
    for j in range(5):
        line += "*"
    print(line)
```

Немного о строках

Немного о строках

```
text = 'съешь ещё этих мягких французских булок'
print(len(text))                    # 39
print('ещё' in text)                # True
print(text.isdigit())               # False
print(text.islower())               # True
print(text.replace('ещё','ЕЩЁ'))    #

for c in text:
    print(c)
```

Немного о строках

```
text = 'съешь ещё этих мягких французских булок'
print(text[0])   # с
print(text[1])   # ъ
print(text[len(text)-1])  # к
print(text[-5])  # б
print(text[:])   # print(text)
print(text[:2])  # съ
print(text[len(text)-2:])  # ок
print(text[2:9])  # ешь ещё
print(text[6:-18])  # ещё этих мягких
print(text[0:len(text):6])  # сеикакл
print(text[::6])  # сеикакл
text = text[2:9] + text[-5] + text[:2] # ...
```

Списки: введение

Списки: введение

Список – пронумерованная, изменяемая коллекция объектов ~~произвольных~~ типов

```
numbers = [1, 2, 3, 4, 5]
print(numbers)  # [1, 2, 3, 4, 5]

numbers = list(range(1, 6))
print(numbers)  # [1, 2, 3, 4, 5]

numbers[0] = 10
print(numbers)  # [10, 2, 3, 4, 5]

for i in numbers:
    i *= 2
    print(i)    # [20, 4, 6, 8, 10]
print(numbers)  # [10, 2, 3, 4, 5]
```

Списки: введение

```
colors = ['red', 'green', 'blue']

for e in colors:
    print(e)      # red green blue

for e in colors:
    print(e*2)    # redred greengreen blueblue

colors.append('gray') # добавить в конец
print(colors == ['red', 'green', 'blue', 'gray']) # True
colors.remove('red') #del colors[0] # удалить элемент
```

Функции

Функции

Это фрагмент программы, используемый многократно

```
def function_name(x):
    # body line 1
    # . . .
    # body line n
    # optional return
```

Функции

```
def f(x):
    return x**2
```

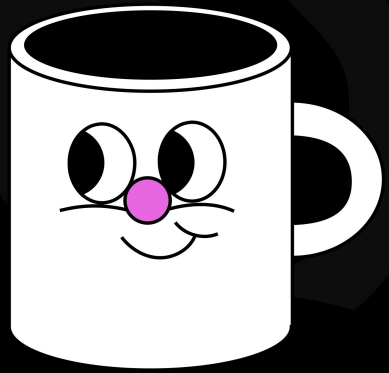
```
def f(x):
    if x == 1:
        return 'Целое'
    elif x == 2.3:
        return 23
    else:
        return
```

```
print(f(1))          # Целое
print(f(2.3))        # 23
print(f(28))         # None
print(type(f(1)))    # str
print(type(f(2.3)))  # int
print(type(f(28)))   # NoneType
```

Итоги

Итоги

Как настроить окружение
Переменные в python: не менять тип в процессе
Операторы ввода и вывода данных
Арифметические операции и проблемы
Условный оператор и его вариации
Цикл while и вариация while-else
Цикл for, range
Вложенные циклы
Строки, списки



Спасибо за внимание