Язык программирования Python

Введение.

Алексей Андреевич Сорокин

спецкурс, ОТИПЛ МГУ,
осенний семестр 2017–2018 учебного года
19 сентября 2017 г.

Язык программирования Python

- Скриптовый язык программирования.
- Придуман Гвидо ван Россумом в 1991.:
  - Python 0.9 — 1991.
  - Python 1.0 — 1994.
  - Python 2.0 — 2000.
  - Python 3.0 — 2008 (нет обратной совместимости с 2.x).
  - Python 2.7.10 — 2010 (ветка 2.x больше не развивается).
  - Последняя версия: Python 3.6 (2017)
- Основной язык для обработки текстов и машинного обучения.

Преимущества Python

- Высокая скорость разработки.
- Удобство чтения кода.
- Меньший объём кода по сравнению с C++, Java, etc.
- Богатая стандартная библиотека (в том числе для обработки текстов).
- Документация на `www.python.org`.
- Поддержка Unicode.
- Наличие большого количества сторонних библиотек.

Недостатки Python

- Невысокая скорость работы (в 10-1000 раз медленнее C++).
- Большие затраты памяти по сравнению с C++.

Недостатки Python

- Невысокая скорость работы (в 10-1000 раз медленнее C++).
- Большие затраты памяти по сравнению с C++.

Решения:

- Использовать части кода на других языках.
- Для численных вычислений использовать специальные библиотеки (numpy, scipy, etc.).

Философия Python

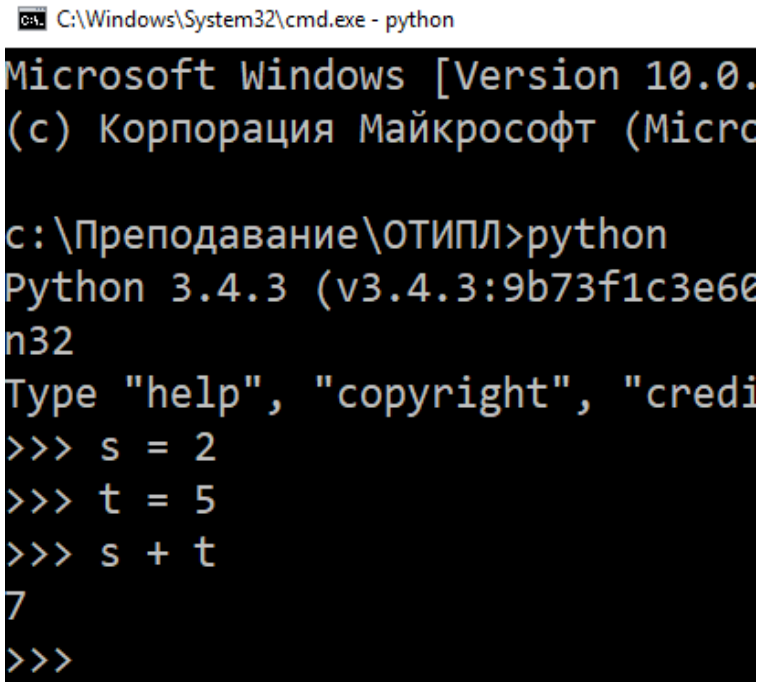
```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one -- and preferably only one -- obvious way
to do it.
```

Основные свойства Python

- Python — интерпретируемый язык (программа исполняется построчечно, а не компилируется заранее в объектный код).

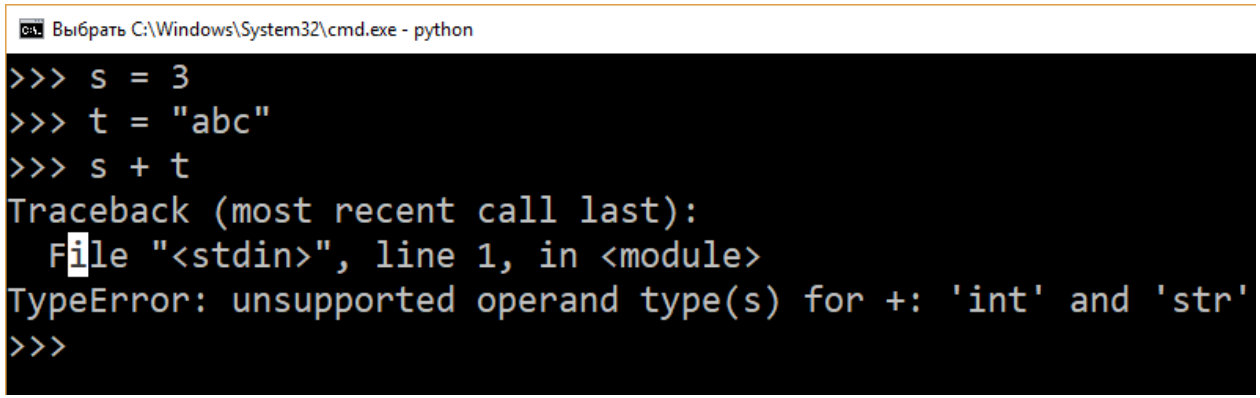
Основные свойства Python

- Python — интерпретируемый язык (программа исполняется построчечно, а не компилируется заранее в объектный код).
- Не надо заранее объявлять переменные: переменная “появляется” в момент первого использования.



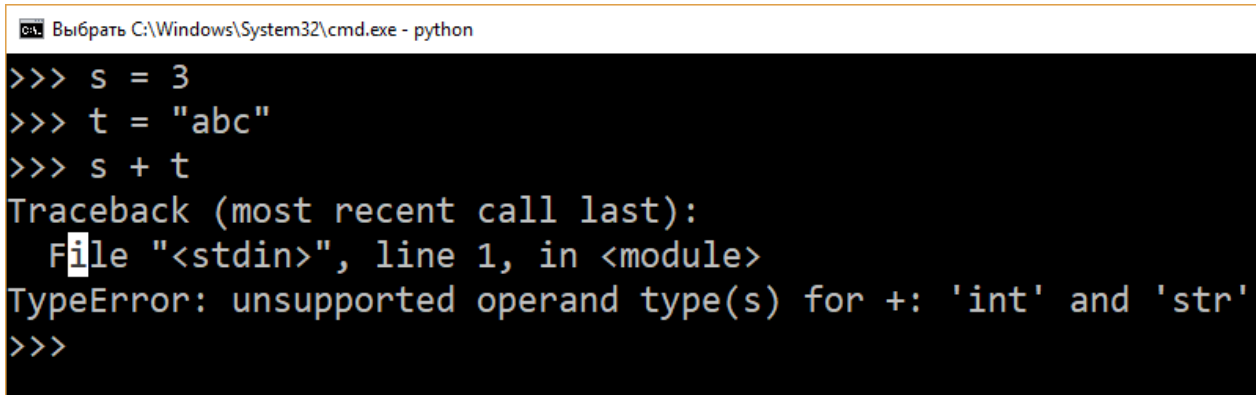
Типы переменных

- Тем не менее, переменные имеют типы.



Типы переменных

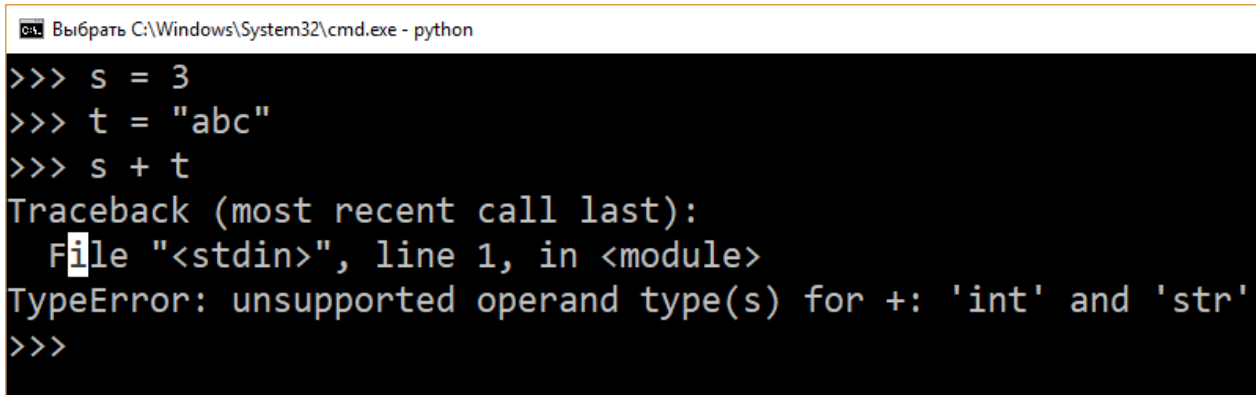
- Тем не менее, переменные имеют типы.



- Нельзя сложить число со строкой.

Типы переменных

- Тем не менее, переменные имеют типы.



- Нельзя сложить число со строкой.
- Зато можно строку со строкой.

Типы переменных

- Основные типы переменных:
  - Целочисленный (**int**).
  - С плавающей точкой (**float**).
  - Булев (**bool**).
  - Строковый (**str**).

Типы переменных

- Основные типы переменных:
  - Целочисленный (**int**).
  - С плавающей точкой (**float**).
  - Булев (**bool**).
  - Строковый (**str**).
- Тип переменной фиксирован, но может меняться в ходе работы (строгая динамическая типизация).

Типы переменных

- Основные типы переменных:
  - Целочисленный (**int**).
  - С плавающей точкой (**float**).
  - Булев (**bool**).
  - Строковый (**str**).
- Тип переменной фиксирован, но может меняться в ходе работы (строгая динамическая типизация).
- **type** — функция, возвращающая тип переменной.

Стандартный ввод

- Стандартный вход считывается функцией **input**().
- input() возвращает строку (до перевода строки), для получения числа нужно применить приведение типов.

```
>>> s = input()
1
>>> type(s)
<class 'str'>
>>> s+2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object to str implicitly
>>> t = int(s)
>>> t + 2
3
```

Стандартный вывод

- Печать призводится функцией **print**().
- **print** печатает объект любого типа (неявно преобразует всё в строки):

```
>>> s = 2
>>> t = "abc"
>>> print(s+2)
4
>>> print(t + "a")
abca
```

Стандартный вывод

- Печать призводится функцией **print**().
- **print** печатает объект любого типа (неявно преобразует всё в строки):

```
>>> s = 2
>>> t = "abc"
>>> print(s+2)
4
>>> print(t + "a")
abca
```

- Можно печатать несколько объектов сразу:

```
>>> print(s, t)
2 abc
```

Стандартный вывод

- Печать призводится функцией **print**().
- **print** печатает объект любого типа (неявно преобразует всё в строки):

```
>>> s = 2
>>> t = "abc"
>>> print(s+2)
4
>>> print(t + "a")
abca
```

- Можно печатать несколько объектов сразу:

```
>>> print(s, t)
2 abc
```

- Можно “красиво” печатать с помощью функции **format**().

```
>>> s = int(input())
5
>>> print("You've entered the number ", s)
You've entered the number  5
>>> print("You've entered the number {}".format(s))
You've entered the number 5
>>> print("The half of your number is {:.2f}".format(s/2))
The half of your number is 2.50
```

Арифметические операции

Основные арифметические операции:

- $+, -, *, /$, возведение в степень $**$.

Арифметические операции

Основные арифметические операции:

- $+, -, *, /$, возведение в степень $**$.
- Деление нацело $//$ и остаток от деления $\%$.

Арифметические операции

Основные арифметические операции:

- $+, -, *, /$, возведение в степень $**$.
- Деление нацело $//$ и остаток от деления $\%$.
- Максимум, минимум: **max**, **min**.

```
>>> s = 3
>>> t = 5
>>> s + t
8
>>> t / s
1.6666666666666667
>>> t // s
1
>>> s ** t
243
>>> t % s
2
>>> max(s, t)
5
>>> min(s, t, 2)
2
```

Логические операции

- bool — специальный логический тип, имеющий только два значения (логические константы): **True** (истина), **False** (ложь).

Логические операции

- bool — специальный логический тип, имеющий только два значения (логические константы): **True** (истина), **False** (ложь).
- Логические значения возвращаются операторами сравнения ($>, <, >=, <=, ==, !=$).

```
>>> s = 5
>>> a = (s > 3)
>>> a
True
>>> b = (s == 4)
>>> b
False
```

Логические операции

Логические операции:

- логическое И: **and**, &.
- логическое ИЛИ: **or**, |.
- логическое НЕ: **not**.

```
>>> a & b
False
>>> a or b
True
>>> not b
True
>>> a | b
True
>>> a and b
False
```

Условный оператор

- Логические значения позволяют проверять условия.

Условный оператор

- Логические значения позволяют проверять условия.
- Синтаксис условного оператора:
  if ⟨условие⟩:
  ⟨TAB⟩ ⟨операция_1⟩
  ⟨TAB⟩ ⟨операция_2⟩

```
>>> s = 5
>>> t = 8
>>> if s % 2 == 1:
...     s = s - 1
...
>>> if t % 2 == 1:
...     t = t - 1
...
>>> print(s, t)
4 8
```

Условный оператор

- В условном операторе можно задавать альтернативное условие:
  if ⟨условие⟩:
  ⟨TAB⟩ ⟨операция_1⟩
  ⟨TAB⟩ ...
  ⟨TAB⟩ ⟨операция_m⟩
  else:
  ⟨TAB⟩ ⟨операция_1⟩
  ⟨TAB⟩ ...
  ⟨TAB⟩ ⟨операция_n⟩

```
>>> s = input("Print your age\n")
Print your age
28
>>> s = int(s)
>>> if s >= 18:
...     print("You are 18 or older")
... else:
...     print("You are younger than 18")
...
You are 18 or older
```

Условный оператор

- Можно задавать несколько альтернативных условий:
  if ⟨условие_1⟩:
  ⟨TAB⟩ ⟨блок_инструкций_1⟩
  elif ⟨условие_2⟩:
  ⟨TAB⟩ ⟨блок_инструкций_2⟩
  ...
  else:
  ⟨TAB⟩ ⟨блок_инструкций⟩

Цикл **while**

- Оператор **while** выполняет операцию, пока истинно некоторое условие.

Цикл **while**

- Оператор **while** выполняет операцию, пока истинно некоторое условие.
- Синтаксис:
  while ⟨условие⟩:
  ⟨TAB⟩ ⟨операция_1⟩
  ⟨TAB⟩ ...
  ⟨TAB⟩ ⟨операция_m⟩

```
s = int(input())
d = 1
while d < s:
        d *= 2
print("The smallest degree of 2 not smaller than {}"
    " is {}".format(s, d))
```

Цикл **while**

```
c:\Преподавание\ОТИПЛ\Python\2017_осень>python while_example.py
4
The smallest degree of 2 not smaller than 4 is 4

c:\Преподавание\ОТИПЛ\Python\2017_осень>python while_example.py
125
The smallest degree of 2 not smaller than 125 is 128

c:\Преподавание\ОТИПЛ\Python\2017_осень>python while_example.py
15
The smallest degree of 2 not smaller than 15 is 16

c:\Преподавание\ОТИПЛ\Python\2017_осень>python while_example.py
19
The smallest degree of 2 not smaller than 19 is 32
```