ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

tpu.ru

Углубленный курс информатики

Лекция 1

Основы программирования на Python

7 февраля 2022 г.

Долганов Игорь Михайлович,
к.т.н., доцент ОХИ ИШПР

Рейтинг-план

Вид учеб. деят-ти	**Количество**	**Баллы**	**Итого**
Лекции	12	1	12
Лаб. работы	20	1	20
Отчет по ЛБ + защита	20	2	40
Тест (КН1)	1	5	5
ИДЗ	1	3	3
		Сумма:	**80**

Экзамен: max = **20** баллов; min = **11** баллов.

ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Содержание

ТОМСКИЙ
ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Язык программирования PYTHON

Язык программирования 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 кроссплатформенности написанных на нём программ.

Разработчик языка Python – голландский программист Гвидо ван Россум

Установка

1. Установить Python (`https://www.python.org/`). В OS X и Unix уже установлен.
2. Только для Windows. При установке выбрать опцию с заданием переменных окружения.

Среда разработки

1. Jupyter Notebook из пакета Anaconda (`https://www.anaconda.com/products/individual`).
2. Текстовые редакторы: VS Code, Sublime, Atom, Notepad++ и др.
3. IDE: PyCharm, Visual Studio, Spyder и др.

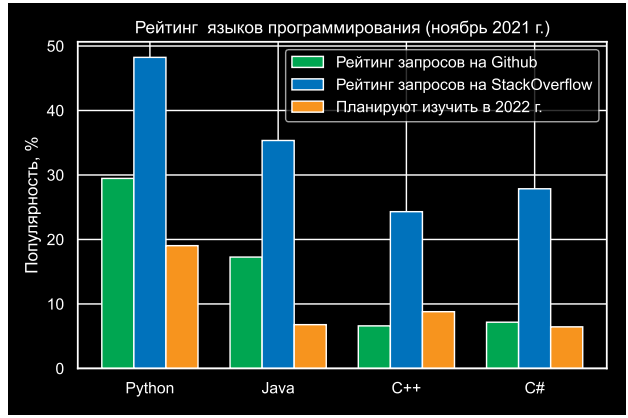
Почему PYTHON?

Кто использует Python?

- Google
- YouTube
- Dropbox
- BitTorrent
- iRobot
- Netflix и Yelp
- Intel, Cisco
- NASA

Сильные стороны

- Качество программного кода
- Продуктивность труда
- Кроссплатформенность
- Библиотеки



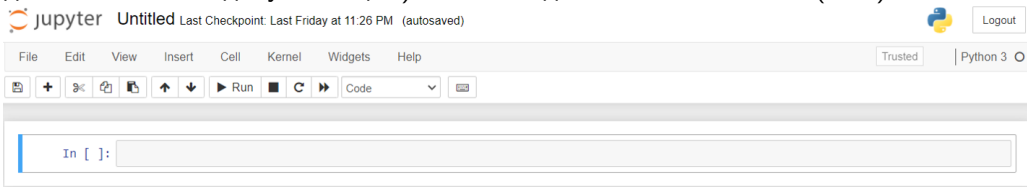
ТОМСКИЙ
ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

JUPYTER NOTEBOOK

Среда разработки Jupyter Notebook

• Чтобы начать работу с новым блокнотом щелкните по кнопке **New** и выберите ядро Python 3. Произойдет открытие новой вкладки браузера с содержанием интерфейса в котором Вы будете писать код и устанавливать соединение с ядром IPython.

• Новый блокнот содержит заголовок, панель меню и панели инструментов под которыми располагается окно ввода командной оболочки IPython, в которое нужно вводить код и команды разметки текста (например, для пояснений и документации) как последовательность ячеек (cells):



ТОМСКИЙ
ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Простейшая программа на PYTHON

Простейшая программа на Python

Простейшей программой на языке Python является пустой файл с расширением `.py`, однако рассмотрим традиционную реализацию программы, выводящей на экран сообщение «Привет Мир!»:

```
1 print("Привет Мир!")
2
```

Для вывода на экран сообщения «Привет Мир!» достаточно обратиться к стандартной функции `print()` и передать ей в качестве параметра строку `"Привет Мир!"`.

ТОМСКИЙ
ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Имена идентификаторов

Имена идентификаторов

- Переменные – это частный случай идентификаторов.
- Идентификаторы – это имена, присвоенные чему-то для его обозначения.

При выборе имен идентификаторов необходимо соблюдать следующие правила:

1. Первым символом идентификатора должна быть буква из алфавита (символ ASCII в верхнем или нижнем регистре, или символ Unicode), а также символ подчеркивания («_»);
2. Остальная часть идентификатора может состоять из букв (символы ASCII в верхнем или нижнем регистре, а также символы Unicode), знаков подчеркивания «_» или цифр $(0-9)$;
3. Имена идентификаторов **чувствительны** к регистру!

Допустимые имена: `i`, `__my_name`, `name_23`, `a1b2_c3`
Недопустимые имена: `2things`, `my-name`, `>a1b2_c3`

Имена идентификаторов

Определяемые Вами имена не могут совпадать с ключевыми словами в Python. К примеру, если Вы попытаетесь выбрать для переменной имя `class`, то Python сообщит об ошибке.

Ключевые слова в Python

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

ТОМСКИЙ
ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Динамическая типизация

Многократное присваивание переменной

- Переменные никогда не располагают какой-либо информацией о типах или связанных с ними ограничениях.
- Понятие типа связано с объектами, а не именами.
- Переменные являются обобщенными по своей сути и просто ссылаются на определенные объекты в конкретный момент времени.

```
1 x = 1
2 x = "Hello"
3 x = 1.2
4
```

- Если переменная используется в каком-то выражении, то при его вычислении она заменяется объектом, ссылкой на который она является.
- Все переменные должны быть присвоены чему либо до того, как их можно будет использовать; использование неприсвоенных переменных приведет к ошибке!

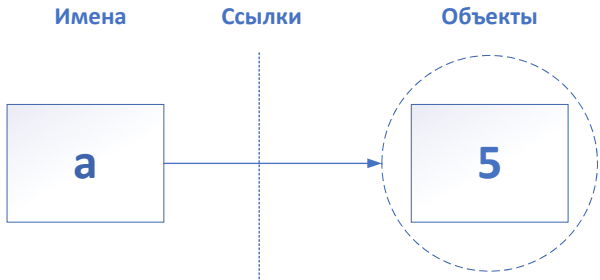
Ссылочная модель данных

В момент, когда происходит присваивание значений переменной:

```
a = 5
```

фактически происходит выполнение трех шагов:

1. Создание объекта для представления значения 5.
2. Создание переменной a.
3. Связывание переменной a с новым объектом 5.



Сборка мусора

В Python при присваивании имени нового объекта происходит освобождение области памяти, занимаемой предыдущим объектом, если на него не ссылается другое имя или объект. Данная операция известна как *сборка мусора*:

```
>>> x = 32
>>> x = "hello"   #Освободить память, занимаемую 32
                  #(если нет других ссылок)
>>> x = 3.1245    #Освободить память, занимаемую "hello"
>>> x = [0, 1, 2, 3] #Освободить память, занимаемую 3.1245
```

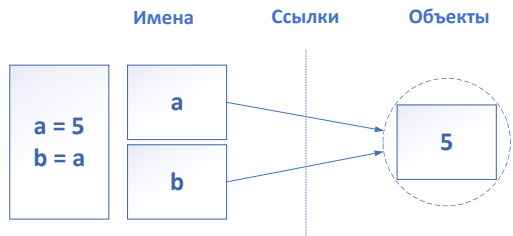
Неоспоримое преимущество сборки мусора заключается в том, что она позволяет свободно использовать объекты без необходимости выделения и освобождения памяти вручную. Python будет автоматически выделять или очищать пространство для хранения объектов при выполнении программы.

Разделяемые ссылки

Рассмотрим создание разделяемых ссылок:

```
>>> a = 5
>>> b = a
```

Такой контекст в Python со множеством переменных, ссылающихся на один и тот же объект, называется *разделяемой ссылкой* (иногда говорят *разделяемый объект*).
Стоит обратить внимание, что при этом имена a и b не связаны друг с другом напрямую.

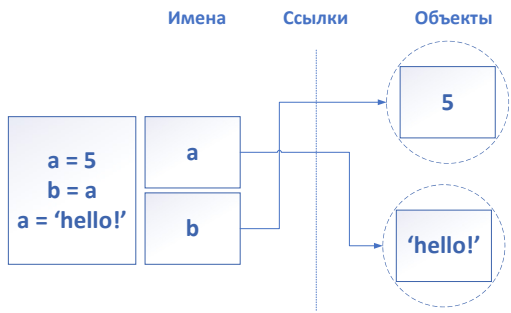


Разделяемые ссылки

Добавим еще один оператор:

```
>>> a = 5
>>> b = a
>>> a = "hello!"
```

Последний оператор просто создает новый объект для представления строкового значения "hello!" и переопределяет ссылку a на этот новый объект. Но все это не отражается на значении b, она по-прежнему продолжает ссылаться на первоначальный объект, целое число 5.



ТОМСКИЙ
ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Запись комментариев

Запись комментариев

Комментарии – это то, что пишется после # и представляет интерес лишь как заметка для читающего программу. Текст программы говорит о том, **КАК**, а комментарии должны объяснять, **ПОЧЕМУ**.

```
1 print("Привет Мир!") #print - это функция
2
```

- Комментарии могут встречаться в тексте программы как отдельные строки или могут быть размещены справа от операторов в строке.
- Текст после символов # пропускается интерпретатором как примечание, предназначенное для человека, читающего данную программу.
- При копировании кода Вы можете игнорировать комментарии, т.к. они носят исключительно информативный характер.
- Грамотное использование комментариев может в значительной степени повысить читаемость, облегчить понимание и, как следствие, улучшить поддерживаемость Вашего программного кода.

ТОМСКИЙ
ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Типы данных

Классификация объектов

Основные типы объектов в Python:

Тип объекта	Категория	Изменяемый?
Числа (все)	Числовые	Нет
Строки	Последовательности	Нет
Списки	Последовательности	Да
Словари	Отображения	Да
Кортежи	Последовательности	Нет
Файлы	Расширения	-
Множества	Множества	Да

ТОМСКИЙ
ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Числовые типы

Числовые литералы

Числа в Python могут быть трех типов:

1. Целые числа (`int`);
2. Числа с плавающей точкой (`float`);
3. Комплексные числа (`complex`).

Литерал	**Расшифровка**
1234, −24, 0, 99999999999999999	Целые числа (неограниченный размер)
1.23, 1., 3.17e-10, 4E210, 4.0e+210	Числа с плавающей точкой
3+4j, 3.0+4.0j, 3J	Литералы комплексных чисел

Целые числа и числа с плавающей точкой

- Целые числа записываются в виде строк десятичных цифр.
- Числа с плавающей точкой имеют десятичную точку и/или необязательный показатель степени со знаком, вводимый посредством нотации e или E.
- Целые числа можно создавать, используя встроенную функцию `int`, а числа с плавающей точкой – при помощи встроенной функции `float`:

```
>>> x = 2.5
>>> y = 1
>>> z = 2.
>>> type(x)  #Число с плавающей точкой
<class 'float'>
>>> type(y)  #Целое число
<class 'int'>
>>> type(z)  #Тоже число с плавающей точкой
<class 'float'>
>>> int(x)   #Получение целого числа отбрасыванием дробной части
2
>>> float(y) #Приведение целого числа к типу float
1.0
```

Комплексные числа

Комплексные числа в Python записываются как

$$\texttt{действительная\_часть} + \texttt{мнимая\_часть}$$

где `мнимая_часть` заканчивается символом j или J.

Комплексные числа можно создать при помощи встроенной функции `complex(real, imag)`:

```
>>> x = 2.5
>>> y = 0.85
>>> complex(x, y)
(2.5+0.85j)
```

ТОМСКИЙ
ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Арифметические операции с числами

Арифметические операции с числами

Основные арифметические операции с числами:

Приоритет	Операция	Описание
1	`x + y`	Сложение
2	`x - y`	Вычитание
3	`x * y`	Умножение
4	`x % y`	Остаток от деления (деление по модулю)
5	`x / y`, `x // y`	Настоящее деление, деление нацело (с округлением в меньшую сторону)
6	`-x`, `+x`	Противоположность, идентичность
7	`x ** y`	Возведение в степень

- Если операция используется с двумя операндами, она называется **бинарной**.
- Существуют также операции с одним операндом, называемые **унарными**.

Составные выражения

Подобно другим языкам программирования, сложные выражения в Python записываются путем объединения арифметических операций. Например, определение суммы двух произведений можно представить в виде комбинации переменных и операций:

$$a \cdot b + c \cdot d$$

Значения подобных и еще более сложных выражений интерпретатор Python будет вычислять в соответствии с *приоритетом операций*.

Круглые скобки в выражениях

Заключение подвыражений в круглые скобки переопределяет приоритеты операций Python; выражения в круглых скобках во всех случаях вычисляются первыми и затем их результаты используются в охватывающих выражениях.

$$(x + y) \cdot z$$
$$x + (y \cdot z)$$

- В первом случае операция `+` применится к переменным `x` и `y` первой, потому что это действие помещено в круглые скобки.
- Во втором случае первой выполняется операция `*`, так же, как если бы круглые скобки отсутствовали.
- В большинстве случаев добавление скобок в объемные составные выражения является хорошим тоном, т.к. не только обеспечивает правильный порядок его вычисления, но и помогает в обеспечении лучшей читабельности.

Операции с числами разных типов

В Python допускается использовать в выражениях различные числовые типы. Например, можно сложить целое число и число с плавающей точкой:

$$30 + 3.14$$

Целые числа проще чисел с плавающей точкой, которые проще, чем комплексные числа. Таким образом, если операция выполняется над целым числом и числом с плавающей точкой, то целое число приводится к значению с плавающей точкой и результат вычисляется тоже как число с плавающей точкой:

```
>>> 30 + 3.14      #Преобразование целого числа в число с плавающей точкой
                   #результат вычисления тоже число с плавающей точкой
33.14
```

Типы также можно преобразовать вручную, вызвав встроенные функции:

```
>>> int(30.145)    #Сокращение числа с плавающей точкой до целого
30
>>> float(6)       #Преобразование целого числа в число с плавающей точкой
6.0
```

Операции деления

- **Классическое (настоящее) деление** – деление с сохранением остатка:

$$x \ / \ y$$

- **Деление с округлением в меньшую сторону** – данная операция всегда усекает дробные остатки, округляя в меньшую сторону не зависимо от типов операндов. Тип результата зависит от типов операндов.

$$x \ // \ y$$

```
>>> 100 / 40      #Сохраняет остаток
2.5
>>> 100 / 40.0
2.5
>>> 100 // 40     #Усекает остаток, возвращает целое число
2
>>> 100 // 40.0   #Округляет в меньшую сторону, вовращает число с
                  #плавающей точкой
2.0
```

Составные операторы присваивания

В Python доступен набор дополнительных форматов операторов присваивания. Данный набор по большей части является всего лишь сокращением, совмещающем в себе бинарную операцию и присваивание.

Составные операторы присваивания

`x += y`	`x &= y`	`x -= y`	`x |= y`
`x *= y`	`x ^= y`	`x /= y`	`x >>= y`
`x %= y`	`x <<= y`	`x **= y`	`x //= y`

К примеру, рассмотрим два способа добавления 10 к имени:

```
>>> x = 1
>>> x = x + 10   #Традиционное присваивание
>>> x
11
>>> x += 10      #Составное присваивание
>>> x
21
```

Operator	Description	Syntax
&	Bitwise AND	x & y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

Другие встроенные операции для работы с числами

Python предлагает встроенные *функции* и стандартные *модули* для работы с числами. Например, встроенные функции `pow` и `abs` вычисляют степень и абсолютное значение, соответственно.

```
>>> import math
>>> math.pi, math.e                                  #Общие константы
(3.141592653589793, 2.718281828459045)
>>> math.sin(2 * math.pi / 180)                      #Синус, косинус, тангенс
0.03489949670250097
>>> math.sqrt(121), math.sqrt(3)                     #Квадратный корень
(11.0, 1.7320508075688772)
>>> pow(3, 4), 3 ** 4, 3.0 ** 4.0                    #Возведение в степень
(81, 81, 81.0)
>>> abs(-24.0), sum((1, 2, 3, 4, 5, ))               #Абсолютное значение,
                                                     #суммирование
(24.0, 15)
>>> min(3, 1, 2, 4, 5), max(3, 1, 2, 4, 5)           #Минимум, максимум
(1, 5)
```

Округление и усечение чисел с плавающей точкой

Существуют также несколько способов отбрасывания десятичных цифр из чисел с плавающей точкой:

```
>>> math.floor(2.512), math.floor(-2.512)    #Округление
                                             #до меньшего целого
(2, -3)
>>> math.trunc(2.512), math.trunc(-2.512)    #Усечение (отбрасывание
                                             #десятичных цифр)
(2, -2)
>>> int(2.512), int(-2.512)  #Преобразование в целое число
(2, -2)
>>> round(2.512), round(2.462), round(2.516, 2) #Округление
(3, 2, 2.52)
```

Распространенные математические функции модуля `math`

Функция	Описание
`math.ceil(x)`	округление до ближайшего большего числа
`math.floor(x)`	округление вниз
`math.trunc(x)`	усекает значение `x` до целого
`math.exp(x)`	e^x
`math.log(x, [base])`	логарифм `x` по основанию `base`, если `base` не указан, вычисляется натуральный логарифм
`math.acos(x)`	арккосинус `x` в радианах
`math.asin(x)`	арксинус `x` в радианах
`math.atan(x)`	арктангенс `x` в радианах
`math.cos(x)`	косинус `x` (`x` указывается в радианах)
`math.sin(x)`	синус `x` (`x` указывается в радианах)
`math.tan(x)`	тангенс `x` (`x` указывается в радианах)

ТОМСКИЙ
ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Оператор ввода данных в PYTHON

Оператор ввода данных

Для ввода пользовательских данных в Python предусмотрена стандартная функция `input()`:

```
x = input("Введите x: ")
y = input("Введите y: ")

print(x + y)
```

```
Введите x: 5
Введите y: 7
57
```

Функция `input()`: возвращает строку, поэтому при вычислении выражения `x + y` в результате получается 57, т.к. происходит конкатенация двух строк.

Оператор ввода данных

При необходимости ввода числовых значений следует явно приводить результат функции `input()` к желаемому типу:

```
1 x = int(input("Введите x: "))
2 y = int(input("Введите y: "))
3
4 print(x + y)
```

```
Введите x: 5
Введите y: 7
12
```

ТОМСКИЙ
ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Пример

Пример программирования арифметического выражения

Вычислите выражение при $x = 1$:

$$\left(\arctan\left(\sqrt{x} - 2.7 \times 10^{-3}\right)\right)^2 + \frac{e^{-x}}{\cos(x)}$$

```
1  import math
2
3
4  x = 1
5
6  y = math.atan(x ** 0.5 - 2.7e-3) ** 2 \
7      + math.exp(-x) / math.cos(x)
8
9  print(y)   #1.2956057140344002
```

ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Контакты

Долганов Игорь Михайлович,
к.т.н., доцент ОХИ ИШПР

Учебный корпус №2, ауд. 136
dolganovim@tpu.ru

Благодарю за внимание!