Программирование на Python

Начало работы

Зачем изучать программирование?

- Программирование развивает креативность, логическое мышление, а также навыки поиска и устранения ошибок.

- Программист может создавать что-то из ничего, пользуясь логикой для составления понятных компьютеру программных конструкций, а если что-то пойдет не так, он отыщет ошибку и исправит проблему.

- Писать программы — занятие увлекательное и временами непростое, однако полученный опыт пригодится и в школе, и дома.

Почему именно Python

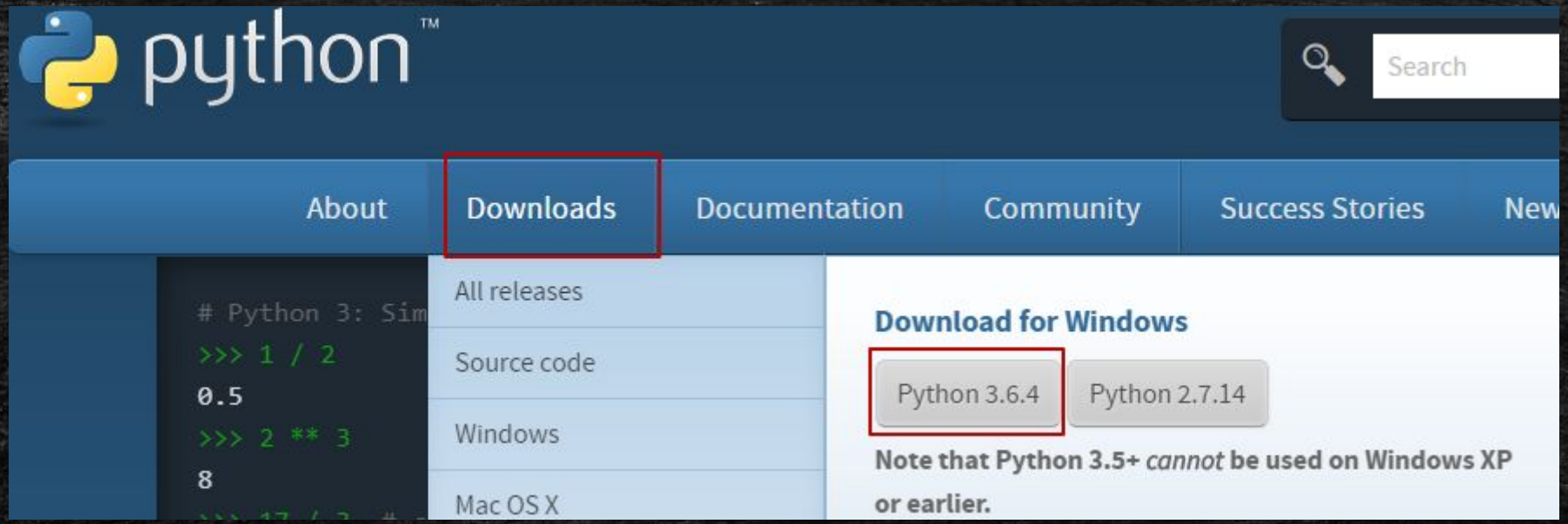
- Простой в изучении. Кроссплатформенность.
- Лаконичный синтаксис.
- Популярность и широкое применение.
  - веб-разработка
  - научные исследования
  - нейронные сети, искусственный интеллект
  - игры и пр.
- Язык используется компаниями Google, NASA, CIA и Disney.

Что можно создать на Python

- Системные утилиты
- Веб-сайты (платформы Django, Flask, Pyramid, Tornado, TurboGears)
- Приложения для научных расчетов (NumPy, SciPy)
- Приложения для Desktop (tkinter, PyQt, wxPython)
- Игры (Pygame)
- Мобильные приложения (kivy)

Что нужно для работы

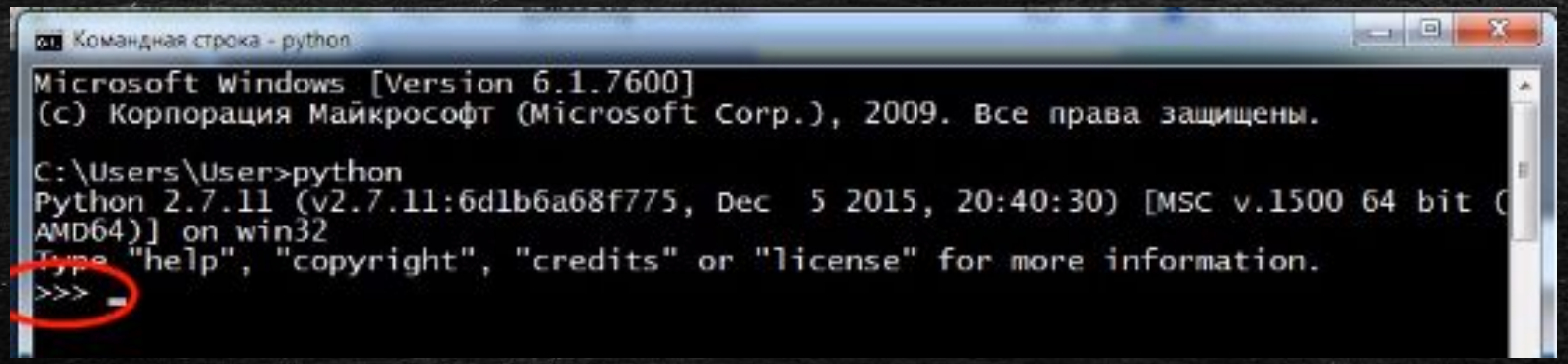
- Устанавливаем интерпретатор Python – www.python.org



- При установке на Windows обязательно отметьте галочкой чекбокс **Add to PATH**

После установки

- Если всё установлено правильно, то начать работу с Python можно прямо в командной строке. Для этого достаточно набрать python



- Такая работа называется интерактивной – каждое действие выполняется сразу по нажатию клавиши Enter.
- Но писать большие программы в интерпретаторе неудобно.

Операции вычисления в Python

Тип операции	Символы в Python
Сложение	+
Вычитание	-
Умножение	*
Деление	/

Для вычислений также используйте скобки, как в обычных математических выражениях.

Операции вычисления в Python

Тип операции	Символы в Python
Возведение в степень	**
Целочисленное деление (определение частного)	//
Деление по модулю (остаток от деления)	%

Для извлечение квадратного корня используйте возведение в степень с указанием дроби, например:
>>> 9 ** (1/2)

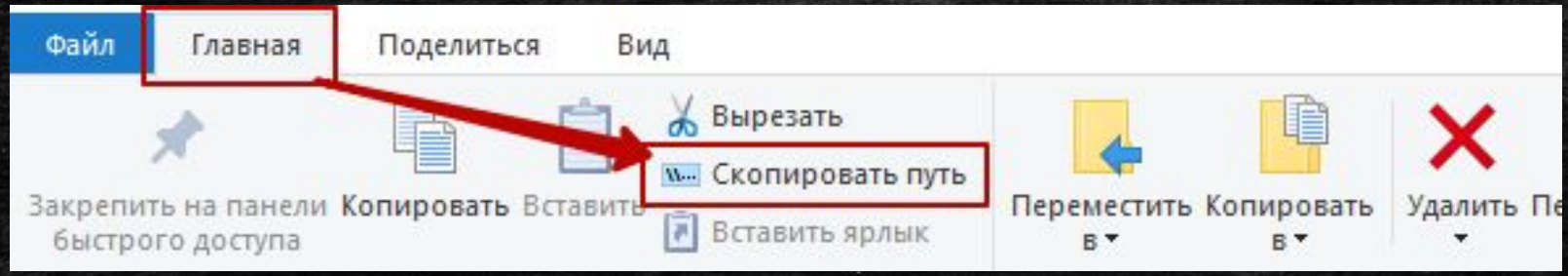
Задание

- В командной строке запустить интерпретатор Python

- Выполнить 10 математических выражений, используя изученные операторы вычислений, а также скобки.

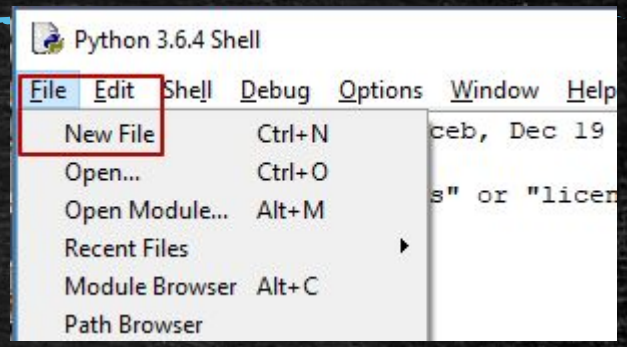
Написание программы

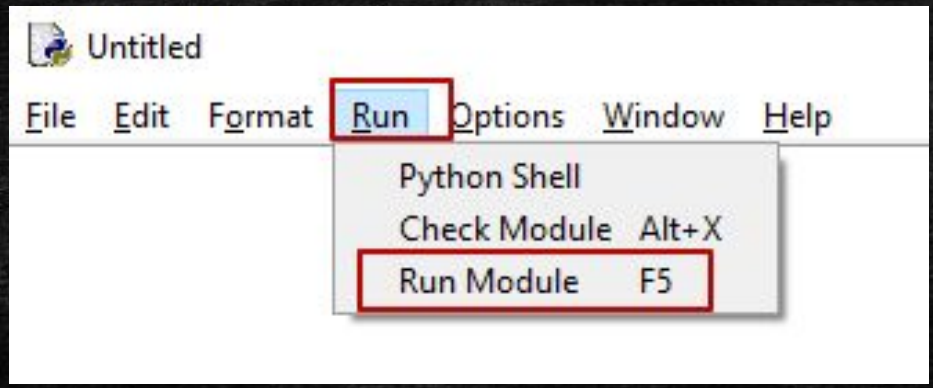
- Можно использовать текстовый редактор **IDLE** или **Atom**.
- Для открытия программы в интерпретаторе необходимо в командной строке также написать python и через пробел указать путь к файлу.
- Для удобства скопировать путь можно в Проводнике при выделенном файле



Написание программы

- В IDLE в меню выбираем File – New File (Ctrl+N).
- Сохраняем файл в свою директорию.
- Для запуска программы – меню Run – Run Module (F5)
  - При этом интерпретатор IDLE должен быть открыт – программа запустится именно в нем.





Пример простой программы

```
# Здесь любой комментарий
print('Привет, проггер')
input()
```

Редактор Atom

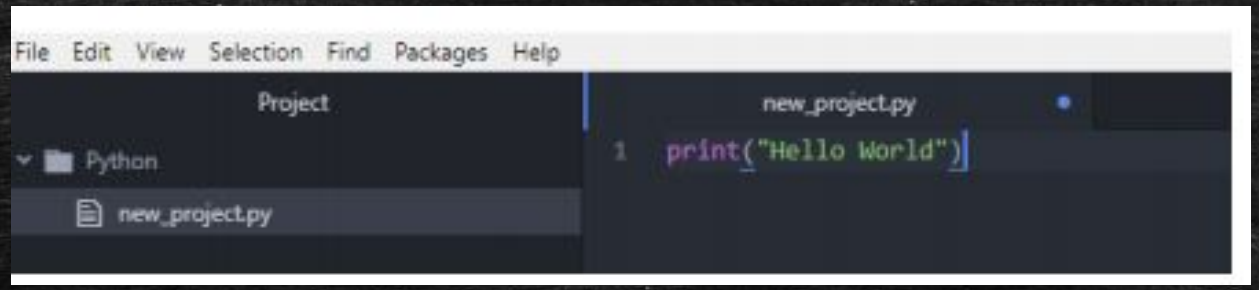
- Для удобства работы с кодом мы будем пользоваться редактором Atom.
- Первым делом добавим рабочую папку - меню File -> Add Project Folder
- И указываем путь к своей папке на компьютере
- После этого в левой части программы можно создавать файлы и папки прямо внутри рабочей папки.
- Для создания файла кликните правой клавишей мыши на папку и выберите пункт New File. Далее укажите имя файла с расширением .py

Редактор Atom

- Для работы с Python в редакторе Atom необходимо дополнительно установить два плагина.
- Меню File -> Settings раздел Install.
- В поиске набираем python. Находим и устанавливаем плагины:
  - autocomplete-python
  - atom-python-run
- После установки плагинов в редакторе Atom можно будет запускать выполнение программ клавишей **F5**.

Проверка работы в Atom

- Пишем простую программу в одну строчку



- Нажимаем F5 и проверяем запустилась ли программа

Использование Строк

- Если вы хотите использовать текст в Python, нужно создать строку.
- Строка создается путем ввода текста между парой одинарных или двойных кавычек.
- В консоли Python строки обычно отображаются с одинарными кавычками. Символы-разделители, в которые заключается строка, никак не влияют на то, как она будет отображаться.
- >>> "Python is fun!"
- 'Python is fun!'
- >>> 'Always look on the bright side of life'
- 'Always look on the bright side of life'

Использование Строк

- Некоторые символы нельзя просто так писать в строке. Например, двойные кавычки нельзя заключать в другие двойные кавычки, оформляющие строку; это приведет к тому, что программа будет преждевременно прервана.
- Чтобы использовать такие символы, нужно создавать для них исключение: ставить перед ними обратную косую черту (бэкслэш).
- Двойные кавычки нужно экранировать только в строках, заключенных в двойные кавычки; то же самое касается и одинарных кавычек.
- >>> 'Brian\'s mother: He\'s not the Messiah. He\'s a very naughty boy!’
- \n используется для перехода на новую строку

Использование Строк

- Три одинарные кавычки в начале и в конце строки. Это позволит использовать в тексте двойные и одинарные кавычки без риска ошибок. Тогда можно поместить внутрь строки любую комбинацию кавычек, кроме трех одинарных подряд.
-
- silly_string = '''"Тут что-то не так, не будь я д'Артаньян", — подумал он.'''

Спецсимолы

- Используйте символ \n для переноса текста на новую строчку.
- Параметр end=‘ ‘ в конце функции print() отменяет перевод на новую строку

```
print("Hello", end=' ')
print("World")
```

- Параметр sep=‘, ‘ добавляет разделитель между другими параметрами вывода

```
print("small", "medium", "large")
print("small", "medium", "large", sep="")
print("small", "medium", "large", sep=", ")
```

Подстановка значений

- Для подстановки заготовленных значений при выводе используются переменные и символ {}.

```
a = 1
b = 0
print(f"Pupils = {a} Students={b}")
```

- Обратите внимание, что перед кавычками стоит буква f, указывающая на особое форматирование с подстановкой.

Задание

- Создайте файл text.py
- Выведите на экран текст в несколько строк
- В самом тексте используйте двойные и одинарные кавычки
- Используйте символ перевода строки между словами
- Запустите программу в интерпретаторе

Переменные в Python

- Переменная - именованное место для хранения данных (чисел, текста, списков). Также переменную можно рассматривать как ярлык, которым помечены некие данные.
- Чтобы создать переменную с именем fred, нужно указать имя, поставить знак «равно» (=) и ввести соответствующие данные.
- >>> fred = 100
- >>> print(fred)

Типы переменных

- Имена переменных пишутся латинскими буквами и цифрами. Первый символ всегда буква.
- Есть разные типы переменных – отличаются хранимой информацией.

Тип переменной	Название	Пример
`int`	Целое число	-150 0 13
`float`	Вещественное число	-12.0 1.1 150.5
`str`	Символьная строка	"Hello" "My name is.."
`bool`	Логический тип данных	True False

Объединение строк

- Подобно целым и дробным числам, строки в Python можно сложить с помощью операции объединения (конкатенации).
- При объединении строк не имеет значения, созданы ли они с одинарными или двойными кавычками.
  - >>> "Spam" + 'eggs'
  - 'Spameggs'
  - >>> print("First string" + ", " + "second string")
  - First string, second string

* Обратите внимание, что при выводе можно добавлять дополнительные символы в кавычках

Объединение строк

- Даже если ваши строки содержат числа, они будут объединяться как строки, а не числа.
  - >>> "2" + "2"
  - '22'
- Сложение строки с числом выдаст ошибку: несмотря на сходство, это два разных объекта.
  - >>> 1 + '2' + 3 + '4'
  - Traceback (most recent call last): File "<stdin>", line 1, in <module> TypeError: unsupported operand type(s) for +: 'int' and 'str'

Объединение строк

- Строки также можно умножать на целые числа. Это приводит к повторению первоначальной строки. Порядок следования не имеет значения, но строка, как правило, следует первой.
  - >>> print("spam" * 3)
  - spamspamspam
- Строки нельзя умножать на другие строки. Строки также нельзя умножать на числа с плавающей запятой, даже если это целые числа.
  - >>> '17' * '87'
  - TypeError: can't multiply sequence by non-int of type 'str'

Ввод данных

- Чтобы получить данные от пользователя, в Python используется функция с интуитивным названием **input** (англ. ввод).
- Функция приглашает пользователя ввести данные, после чего введенный текст возвращается в виде строки.
- >>> input(“Введите что-нибудь пжт: “)

```
name = input("Your name: ")
print("Hello, " + name)
```

Задание

- Создайте программу, которая бы запрашивала у пользователя его имя, а затем приветствовала по имени.
- При выводе сообщения на экран используйте опцию подстановки значений в {}

Домашнее задание

- Скачать и установить интерпретатор Python
- Выполнить в интерактивном режиме действия:
  - 10/3 ; 10//3 ; 10%3; 10** 3 – разобраться, что они делают
  - 20*4+2 и 20*(4+2)
  - Сделать скриншот вычислений
- Придумать своей программе название, вывести его перед строкой приветствия, добавить строку ожидания ввода пользователя. Программу сохранить как dz01.py
- В систему обучения загрузить архив с вычислениями в интерактивном режиме и файл с программой dz01.py

Преобразование типа

- В Python возможность выполнения операции зависит от типов данных. Например, нельзя сложить две строки, содержащие числа 2 и 3, если вы хотите получить число 5: программа будет интерпретировать их как строки и вернет результат 23.
- В этом случае нужно преобразовать тип данных.
- В предыдущем примере нужно использовать функцию **int**
- >>> "2" + "3"
- '23'
- >>> int("2") + int("3")
- 5

Преобразование типа

- Мы уже познакомились с такими типами данных, как целые и дробные числа и строки. Для преобразования в эти типы данных используются соответственно функции int, float и str.
- Другим примером преобразования типа является конвертация пользовательского ввода (строки) в числа (целые или дробные) с целью выполнения вычислений.
- >>> float(input("Enter a number: ")) + float(input("Enter another number: "))
- Enter a number: 40
- Enter another number: 2
- 42.0