Moscow Coding School

Python как первый язык

Преподаватель: Захарчук Сергей Сергеевич

Сегодня

- Структура курса
- Организационные вопросы
- Введение в Python
- Практика

Почему Python?

- Простой
- Мощный
- Распространенный
- Востребованный

О языке

- Интерпретируемый
- Динамический
- Разный
- Развивающийся

Наконец-то

https://try.jupyter.org/

https://github.com/rtridz/pythonfirst

Cygwin – if you [win]

Download and install from https://www.cygwin.com/

Type in cmd terminal

```
cd C:\cygwin64
setup-x86_64.exe -q -P wget,tar,qawk,bzip2,subversion,vim,git,openssh,nginx
```

Внимательно проверьте путь к файлу

```
echo 'alias cyg-get="/cygdrive/C/cygwin/setup-x86_64.exe -q -P"' > ~/.bash_profile
cyg-get nano
cyg-get python3
cyg-get python3-pip
pip3 isntall –U pip setuptools
```

and isntall anaconda from here https://www.continuum.io/downloads

```
echo ' export PATH="C:\ProgramData\Anaconda3:$PATH"'>~/.bash_profile
```

Типы данных в Python

- None (null)
- Числа: целые, вещественные, комплексные
- Исключения в python
- Байты (bytes и bytearray)
- Файлы.
- Множества (set и frozenset)
- Функции и их аргументы
- Кортежи (tuple)
- Словари (dict)
- Списки (list)
- Строки.

Null

Эквивалент null в Python: None

Присвоить переменной значение None очень просто:

```
In [1]: none_variable = None
        … type(none_variable)
Out[1]: NoneType
>>> type(None)
<class 'NoneType'>
```

Целые числа (int)

x + y	Сложение
x - y	Вычитание
x * y	Умножение
x / y	Деление
x // y	Получение целой части от деления
x % y	Остаток от деления
-x	Смена знака числа
abs(x)	Модуль числа
divmod(x, y)	Пара (x // y, x % y)
x ** y	Возведение в степень
pow(x, y[, z])	x^y по модулю (если модуль задан)

Битовые операции

x | y	Побитовое *или*
x ^ y	Побитовое *исключающее или*
x & y	Побитовое *и*
x << n	Битовый сдвиг влево
x >> y	Битовый сдвиг вправо
~x	Инверсия битов

Системы счисления

- **int**([object], [основание системы счисления]) - преобразование к целому числу в десятичной системе счисления. По умолчанию система счисления десятичная, но можно задать любое основание от 2 до 36 включительно.
- **bin**(x) - преобразование целого числа в двоичную строку.
- **hex**(x) - преобразование целого числа в шестнадцатеричную строку.
- **oct**(x) - преобразование целого числа в восьмеричную строку.

Вещественные числа (float)

Вещественные числа поддерживают те же операции, что и целые. Однако (из-за представления чисел в компьютере) вещественные числа неточны, и это может привести к ошибкам:

```
>>> 0.3 + 0.3 + 0.3
0.8999999999999999
```

Исключения в python.

Исключения (exceptions) - ещё один тип данных в python. Исключения необходимы для того, чтобы сообщать программисту об ошибках.

Самый простейший пример исключения - деление на ноль:

```
>>> 1 / 0
Traceback (most recent call last):
File "", line 1, in 1 / 0
ZeroDivisionError: division by zero
```

BaseException - базовое исключение, от которого берут начало все остальные.

try- except

Теперь, зная, когда и при каких обстоятельствах могут возникнуть исключения, мы можем их обрабатывать. Для обработки исключений используется конструкция **try - except**.

Первый пример применения этой конструкции:

```
>>> try:
        ... k = 1 / 0
... except ZeroDivisionError:
        ... k = 0

>>> print(k)
0
```

bytes и bytearray

Что такое байты? Байт - минимальная единица хранения и обработки цифровой информации. Последовательность байт представляет собой какую-либо информацию (текст, картинку, мелодию...).

Bytearray в python - массив байт. От типа **bytes** отличается только тем, что является изменяемым.

Файлы. Работа с файлами.

Прежде, чем работать с файлом, его надо открыть. С этим замечательно справится встроенная функция open:

```
>>> f = open('example.py', 'r')
```

Файлы. Работа с файлами.

Режим	Обозначение
'r'	открытие на чтение (является значением по умолчанию).
'w'	открытие на запись, содержимое файла удаляется, если файла не существует, создается новый.
'x'	открытие на запись, если файла не существует, иначе исключение.
'a'	открытие на дозапись, информация добавляется в конец файла.
'b'	открытие в двоичном режиме.
't'	открытие в текстовом режиме (является значением по умолчанию).
'+'	открытие на чтение и запись

Режимы могут быть объединены, то есть, к примеру, 'rb' - чтение в двоичном режиме. По умолчанию режим равен 'rt'.
И последний аргумент, encoding, нужен только в текстовом режиме чтения файла. Этот аргумент задает кодировку.

Файлы. Работа с файлами.

Откроем файл на запись:

```
>>> s= [str(i) for i in range(20)]
>>> f = open('text.txt', 'w')
>>> for index in l:
        ... f.write(index + '\n')
```

После окончания работы с файлом его **обязательно нужно закрыть** с помощью метода close:

```
f.close()
```

Строки.

Конкатенация (сложение)

```
>>> S1 = 'spam’
>>> S2 = 'eggs'
>>> print(S1 + S2) 'spameggs'
```

Строки.

Конкатенация (сложение)

```
>>> S1 = 'spam’
>>> S2 = 'eggs'
>>> print(S1 + S2)
 'spameggs’
```

Дублирование строки

```
>>> print('spam' * 3)
spamspamspam
```

Строки.

Длина строки (функция len)

```
>>> len('spam')
4
```

Доступ по индексу

```
>>> S = 'spam'
>>> S[0]
's'
>>> S[2]
'a'
>>> S[-2]
'a'
```

Строки.

Извлечение среза

Оператор извлечения среза: [X:Y]. X – начало среза, а Y – окончание;

символ с номером Y в срез не входит. По умолчанию первый индекс равен 0, а второй - длине строки.

```
>>> s = 'spameggs'
>>> s[3:5]
'me'
```

Строки.

При вызове методов необходимо помнить, что строки в Python относятся к категории неизменяемых последовательностей, то есть все функции и методы могут лишь создавать новую строку.

```
>>> s = 'spam'
>>> s[1] = 'b'
Traceback (most recent call last):
File "", line 1, in s[1] = 'b'
TypeError: 'str' object does not support item assignment
>>> s = s[0] + 'b' + s[2:]
>>> s
'sbam'
```

Строки.

Таблица "Функции и методы строк"

Функция или метод	Назначение
S = 'str'; S = "str"; S = '''str'''; S = """str"""	Литералы строк
S = "s\np\ta\nbbb"	Экранированные последовательности
S = r"C:\cygwin\"	Неформатированные строки (подавляют экранирование)
S = b"byte"	Строка байтов
S1 + S2	Конкатенация (сложение строк)
S1 * 3	Повторение строки
S[i]	Обращение по индексу
S[i:j:step]	Извлечение среза

Строки.

Таблица "Функции и методы строк"

len(S)	Длина строки
S.find(str, [start],[end])	Поиск подстроки в строке. Возвращает номер первого вхождения или -1
S.rfind(str, [start],[end])	Поиск подстроки в строке. Возвращает номер последнего вхождения или -1
S.index(str, [start],[end])	Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError
S.rindex(str, [start],[end])	Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError
S.replace(шаблон, замена)	Замена шаблона
S.split(символ)	Разбиение строки по разделителю

Строки.

Таблица "Функции и методы строк"

S.isdigit()	Состоит ли строка из цифр
S.isalpha()	Состоит ли строка из букв
S.isalnum()	Состоит ли строка из цифр или букв
S.islower()	Состоит ли строка из символов в нижнем регистре
S.isupper()	Состоит ли строка из символов в верхнем регистре
S.isspace()	Состоит ли строка из неотображаемых символов (пробел, символ перевода страницы ('\f'), "новая строка" ('\n'), "перевод каретки" ('\r'), "горизонтальная табуляция" ('\t') и "вертикальная табуляция" ('\v'))

Строки.

Таблица "Функции и методы строк"

S.istitle()	Начинаются ли слова в строке с заглавной буквы
S.upper()	Преобразование строки к верхнему регистру
S.lower()	Преобразование строки к нижнему регистру
S.startswith(str)	Начинается ли строка S с шаблона str
S.endswith(str)	Заканчивается ли строка S шаблоном str
S.join(список)	Сборка строки из списка с разделителем S
ord(символ)	Символ в его код ASCII
chr(число)	Код ASCII в символ

Строки.

Таблица "Функции и методы строк"

S.capitalize()	Переводит первый символ строки в верхний регистр, а все остальные в нижний
S.center(width, [fill])	Возвращает отцентрованную строку, по краям которой стоит символ fill (пробел по умолчанию)
S.count(str, [start],[end])	Возвращает количество непересекающихся вхождений подстроки в диапазоне [начало, конец] (0 и длина строки по умолчанию)
S.expandtabs([tabsize])	Возвращает копию строки, в которой все символы табуляции заменяются одним или несколькими пробелами, в зависимости от текущего столбца. Если TabSize не указан, размер табуляции полагается равным 8 пробелам
S.lstrip([chars])	Удаление пробельных символов в начале строки
S.rstrip([chars])	Удаление пробельных символов в конце строки
S.strip([chars])	Удаление пробельных символов в начале и в конце строки

Строки.

Таблица "Функции и методы строк"

S.capitalize()	Переводит первый символ строки в верхний регистр, а все остальные в нижний
S.center(width, [fill])	Возвращает отцентрованную строку, по краям которой стоит символ fill (пробел по умолчанию)
S.count(str, [start],[end])	Возвращает количество непересекающихся вхождений подстроки в диапазоне [начало, конец] (0 и длина строки по умолчанию)
S.expandtabs([tabsize])	Возвращает копию строки, в которой все символы табуляции заменяются одним или несколькими пробелами, в зависимости от текущего столбца. Если TabSize не указан, размер табуляции полагается равным 8 пробелам
S.lstrip([chars])	Удаление пробельных символов в начале строки
S.rstrip([chars])	Удаление пробельных символов в конце строки
S.strip([chars])	Удаление пробельных символов в начале и в конце строки

Строки.

Таблица "Функции и методы строк"

S.partition(шаблон)	Возвращает кортеж, содержащий часть перед первым шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий саму строку, а затем две пустых строки
S.rpartition(sep)	Возвращает кортеж, содержащий часть перед последним шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий две пустых строки, а затем саму строку
S.swapcase()	Переводит символы нижнего регистра в верхний, а верхнего – в нижний
S.title()	Первую букву каждого слова переводит в верхний регистр, а все остальные в нижний
S.zfill(width)	Делает длину строки не меньшей width, по необходимости заполняя первые символы нулями
S.ljust(width, fillchar=" ")	Делает длину строки не меньшей width, по необходимости заполняя последние символы символом fillchar
S.rjust(width, fillchar=" ")	Делает длину строки не меньшей width, по необходимости заполняя первые символы символом fillchar
S.format(*args, **kwargs)	Форматирование строки

Модули в Python

- Помимо стандартных выражений для работы с числами (а в Python их не так уж и много), в составе Python есть несколько полезных модулей.
- Модуль math предоставляет более сложные математические функции.

Создаём Telegram бота

- Необходимо установить приложение Telegram. https://telegram.org/
- Добавляем к себе в контакт-лист бота с именем **BotFather**
- Запускаем процедуру "общения" с ботом нажатием кнопки *Start*. Далее перед нами предстанет список команд точно как на скриншоте.
- Для того, чтобы создать нового бота необходимо выполнить команду */newbot* и следовать инструкциям. Обратите внимание, что **username для бота должен всегда содержать в конце слово bot**. Например, nickname_bot.

Создаём Telegram бота

После создания бота, обратите внимание на строку с текстом:

Use this token to access the HTTP API:383941154:FAH7QYaqdYQQXBSKh0I7TZKpYy7IVWOfr9g

Создаём Telegram бота

За которой следует т.н. token по которому мы будем манипулировать нашим ботом. Помимо функции создания telegram бота, BotFather также имеет ряд других возможностей:

- Присвоить боту описание
- Установить аватар
- Поменять token

Создаём Telegram бота

Можно обойтись и обычным Python скриптом, правда в этом случае необходимо будет периодически опрашивать Telegram на предмет новых запросов от пользователей бота (используя метод *getUpdates*) и увеличивая offset для получения самых последних данных без повторений. В Telegram существует два взаимоисключающих метода получения команд/сообщений для вашего бота.

- Использование вызова API метода *getUpdates*
- Установка Webhook

Установка Webhook заключается в передаче боту специального URL адреса на который будет поступать POST запрос каждый раз, когда кто-то начнёт посылать сообщения боту. Именно этот вариант мы и будем использовать для взаимодействия между ботом и его пользователем.

Подробнее о *getUpdates* и *setWebhook* можно почитать соответственно здесь:

https://core.telegram.org/bots/api#getting-updates

Создаём Telegram бота

```
$ pip install telepot
import telepot
token = '123456'
TelegramBot = telepot.Bot(token)
print (TelegramBot.getMe())
```

Создаём Telegram бота

Процесс общения с telegram ботом происходит по HTTPS; для передачи данных используется JSON. Метод *getUpdates* возвращает список/массив из объектов типа Update.

Имеется параметр update_id, который служит в качестве offset параметра при вызове метода getUpdates. То есть update_id+1 вернёт все сообщения, поступившие после последнего update_id, при этом все предыдущие сообщения будут удалены.