ТЕХНОАТОМ

Веб-программирование Python

Лекция 1. Введение

Михалёв Олег

python-разработчик myTarget

2009 — 2012: ОрелГТУ, программист-исследователь
2012 — 2015: БелГУ, программист-исследователь
2015 — 2016: Mail.Ru Group/My.Com, программист

Контакты

o.mikhalev@corp.mail.ru
@mhalairt

Для оперативных коммуникаций:
telegram.me/joinchat/B9Fx5UAMAn_cfcmSbajAXQ

ТЕХНОАТОМ

Сегодня

- **Организационная часть:**
- О курсе
- Итоговый проект
- Формат занятий
- **Мат. часть:**
- Идеология Python
- Рабочее окружение
- Синтаксис Python

Цель - сформировать навыки, необходимые для самостоятельной разработки веб-приложений

14 занятий

9 лекций
4 семинара
1 экзамен

Лекции 1-2, Семинар 1:
Основы программирования на Python

Лекции 3-5, Семинар 2:
Интернет и веб-программирование

Лекции 6-7, Семинар 3:
Обработка данных

Лекции 8-9, Семинар 4:
Специфика взаимодействия в веб-среде

14 занятий

8 заданий
4 рубежных контроля
1 итоговый проект

Выполнение заданий - 40 баллов
Сдача итогового проекта - 40 баллов
Активность на семинарах - 20 баллов

Удовлетворительно | 60+
Хорошо | 75+
Отлично | 90+

ТЕХНОАТОМ

«Домашняя Бухгалтерия»

Почему?

Веб-сервисы
Обработка данных
Статистика
Защита информации

Каждое следующее задание приближает
к выполнению итогового проекта

ТЕХНОАТОМ

Недостатки лекций:
- отсутствие обратной связи
- отсутствие практики

Обновление:
- диалог с аудиторией, опросы
- рабочие пятиминутки

«Семинары» будем использовать для контроля выполнения домашних заданий, разбора непонятного и обсуждения нового

Вопросы

Python - высокоуровневый язык программирования общего назначения с минималистичным синтаксисом

Python поддерживает несколько парадигм программирования:

- структурное
- объектно-ориентированное
- функциональное
- императивное
- аспектно-ориентированное

Достоинства:

- простота и удобочитаемость
- высокая скорость разработки
- платформо-независимость и переносимость
- огромный набор готовых библиотек
- широкие возможности интеграции

Недостатки:

- скорость

Области применения Python

ТЕХНОАТОМ

Системное программирование
Графические интерфейсы и веб-сценарии
Прототипирование
Научные вычисления и анализ данных
Искусственный интеллект

Программа на языке Python, в самой простой форме, – это обычный текстовый файл с расширением ".py", содержащий инструкции.

```
> print('Привет, Мир!')
```

Запуск программы осуществляется с помощью интерпретатора

```
C:\> python hello.py
```

Интерпретатор скомпилирует программу в байт код и передаст его на исполнение виртуальной машине.

Фактически виртуальная машина Python – это цикл, который выполняет перебор инструкций в байт-коде, одну за одной, и выполняет соответствующие им операции.

ТЕХНОАТОМ

CPython

Стандартная реализация языка Python

Jython

Реализация языка Python с виртуальной машиной Java

ТЕХНОАТОМ

IronPython

Реализация языка Python с виртуальной машиной .Net/Mono

Окончание поддержки Python 2 запланированно на 2020 год

Python 3.5.2
https://www.python.org/downloads/

Отличия Python 3:

- изменены механизмы сравнения
- изменены строки
- частично изменен синтаксис
- введены итераторы
- print стал функцией

PyDev
http://www.pydev.org/

PyCharm
https://www.jetbrains.com/pycharm/

Python Tools for Visual Studio
https://www.visualstudio.com/en-us/features/python-vs.aspx

Sublime Text
https://www.sublimetext.com/

PyDev
http://www.pydev.org/

PyCharm
https://www.jetbrains.com/pycharm/

Python Tools for Visual Studio
https://www.visualstudio.com/en-us/features/python-vs.aspx

Sublime Text
https://www.sublimetext.com/

Python Package Index — каталог библиотек и ПО, написанного на языке программирования Python

PyPI
https://pypi.python.org/pypi

pip - пакетный менеджер Python, использующий PyPI.
Включен в поставку Python 3 с версии 3.4

```
C:\> pip install Django
C:\> pip install --upgrade Django
C:\> pip uninstall Django
```

Идентификаторы в Python – это имена, используемые для определения переменных, функций, классов, модулей и других объектов.

```
>>> x = 0
```

Идентификатор начинается с букв A-Z или a-z, либо знака подчеркивания "_", после чего следуют ноль или больше букв, знаков подчеркивания или цифр от 0 до 9.

Подробно о принятой стилистике можно прочитать в PEP8: https://www.python.org/dev/peps/pep-0008/

При обозначении границ блоков кода не используются фигурные скобки. Вместо этого – в Python есть отступы строк.

```
>>> if True:
...     print('Привет, Мир!')
```

Количество отступов в начале строки не имеет значения, но все операторы внутри такого блока должны иметь их одинаковое количество.

Области видимости не разграничены блоками кода.

Если присваивание производится внутри функции/метода/класса, переменная является:

- локальной для родительского объекта (функции/метода/класса)
- нелокальной для объектов (функций/методов/классов) определенных внутри родительского

Если присваивание производится за пределами всех объектов, она является глобальной для всего модуля.

Разрешение имен производится по следующей схеме:

- локальная область видимости (local)
- локальная область видимости объемлющего объекта 1 (enclosing)
- локальная область видимости объемлющего объекта .. (enclosing)
- локальная область видимости объемлющего объекта n (enclosing)
- глобальная область видимости (global)
- встроенная область видимости (built-in)

Подключение модуля выполняется инструкцией **import**.
В этом случае импортированный модуль переносится в глобальную область видимости.

```
>>> import math
>>> print(math.pi)
3.141592653589793
```

Условный оператор **if**, альтернативный блок выбора **else**.

```
>>> if x < 0:
...     print('Отрицательное значение')
... else:
...     print('Положительное значение')
```

Если условий и альтернатив несколько, можно использовать **elif** - аналог оператора выбора switch или case.

```
>>> if x <= 0:
...     x = 0
...     print('Отрицательное значение, изменено на ноль')
... elif x == 0:
...     print('Ноль')
... elif x == 1:
...     print('Один')
... else:
...     print('Больше')
```

Оператор **pass** не делает ничего. Он может использоваться когда синтаксически требуется присутствие оператора, но от программы не требуется действий.

```
>> if x <= 0:
...     pass
```

Оператор **while** позволяет определить простейший цикл. Блок кода в теле цикла будет выполняться пока указанное условие истинно.

```
>>> while x > 0:
...     x =- 1
```

Оператор **for** позволяет последовательно перебрать все элементы переданной последовательности.

```
>>> for x in l:
...     print(x)
```

Можно создавать свои составные типы данных и обходить их элементы в цикле for. Самым простым примером являются арифметические прогрессии.

Класс **range** поможет создавать арифметические последовательности.

```
> range(n) # последовательность от 0 до n-1
> range(e0, n, step) # последовательность от e0 до n-1 с шагом step
```

По умолчанию step равен единице (шаг можно не указывать).

```
>>> for x in range(2, 10, 2):
...     print(x)
2
4
6
8
```

Оператор **break** прерывает выполнение самого ближайшего вложенного цикла **for** или **while**. Оператор **continue**, продолжает выполнение цикла со следующей итерации.

```
>>> for x in range(10):
...     if x < 5:
...         continue
...     print(x)
```

```
>>> while True:
...     x += 1
...     if x > 5:
...         break
```

Операторы циклов могут иметь ветвь **else**. Она исполняется, когда цикл выполнил перебор до конца (в случае **for**) или когда условие становится ложным (в случае **while**), но не в тех случаях, когда цикл прерывается по **break**.

```
>>> for i in range(10):
...   if i % 10 == 0:
...     break
... else:
...     print('Расчет окончен')
```

ТЕХНОАТОМ

Спасибо за внимание!

Михалёв Олег
o.mikhalev@corp.mail.ru