Python Start

Дмитрий Береговец, FoxmindEd,
2022

План курса

- ❑ Урок 1 - Настройка и вступление
- ❑ Урок 2 - Структуры данных
- ❑ Урок 3 - Функции и файлы
- ❑ Урок 4 - Классы и объекты
- ❑ Урок 5 - ООП
- ❑ Урок 6 - Дополнительные инструменты

foxminded

Урок 1 - Настройка

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

PyCharm Community - https://www.jetbrains.com/ru-ru/pycharm/download

foxminded

Урок 1 - Переменные

Присвоить переменной a значение 5

```
a = 5
```

Переменные с подсказкой

```
a: int = 5
b: str = “hello”
```

Основные типы

int - целочисленное число

float - дробное число

str - строка

Суммирование

```
a = 5
b = 4
c = a + b  # результат 9
```

Арифметические операции в python

https://www.w3schools.com/python/gloss_python_arithmetic_operators.asp

Конкатенация строк

```
a = “Hello”
b = “ “
c = “World!”
string = a + b + c  # результат “Hello World!”
```

foxminded

Урок 1 - Дополнительные действия со строкой

Вычленить начало строки

a = “Hello World!”

a[0] *# результат “H”*

a[:5] *# результат “Hello”*

a[3:] *# результат “lo World!”*

a[3:5] *# результат “lo”*

Повторить строку несколько раз

b = “rock”

b * 3 *# результат “rockrockrock”*

Функция print()

print(5) - *вывод в консоль числа 5 плюс перевод каретки на новую строку*

print(5, end=’‘) - *вывод в консоль числа 5 без перевода каретки*

foxminded

Урок 1 - Практика

1. Используя 3 раза подряд функцию print() вывести в консоль следующий текст. Перенос строк должен соблюдаться

Hello World!

Bye!

2. Присвоить переменной a значение 13. Присвоить переменной b значение 14. Вывести в консоль результат умножения 2 переменных. Вывести в консоль результат сведения 13 в степени 14.

foxminded

Урок 2 - Списки, кортежи

Список *(изменяемый)*

a = [12, “hello”, [3.4, 1]]

a[0] - первый элемент списка

a[1:] - срез списка начиная со второго элемента

a.append(4) - добавить 4 в конец списка

last = a.pop() - забрать последний элемент списка и присвоить в переменную

result = [3, 4] + [5, 6] - соединить 2 списка и присвоить результат в переменную

Детальнее про списки

https://docs.python.org/3.9/library/stdtypes.html#mutable-sequence-types

Кортеж *(неизменяемый)*

k = (3, “a”, 4.5)

k[:2] - срез до второго элемента включительно

result = (3, 4) + (5, 6) - суммирование кортежей

foxminded

Урок 2 - Сеты, словари

Сет *(изменяемый)*

a = {12, “hello”, 3.4}

a.add(“new”) - добавить элемент в сет

a.pop() - достать произвольный элемент из сета

a.discard(12) - удалить элемент из сета

a.intersection(b) - общие члены сета a и сета b

a.union(b) - объединить 2 сета

a & b - общие члены сета a и сета b

a | b - объединить 2 сета

Операции над сетами

https://docs.python.org/3.9/library/stdtypes.html#set-types-set-frozenset

Словарь *(изменяемый)*

s = {“name”: “Dan”, “surname”: “Brown”}

s[“name”] - получить имя

s.pop(“name”) - получить имя и удалить из словаря

s[“age”] = 34 - добавить в словарь новый ключ “age” и присвоить ему значение 34

Детальнее про словари

https://docs.python.org/3.9/library/stdtypes.html#mapping-types-dict

foxminded

Урок 2 - Логические операции, проверка if

Логические операции

and - логическое и

or - логическое или

not - логическое не

== - равно

!= - неравно

> - больше

< - меньше

>= - больше равно

<= - меньше равно

Проверка через if

```
if a == b and b == c:   # можно написать a == b == c
    print("All numbers are equal")
elif a == b:
   print("a and b equal only")
else:
    print("No equal numbers")
```

Дополнительно

https://docs.python.org/3/reference/expressions.html#value-comparisons

foxminded

Урок 2 - Практика

1. Создать свой список, кортеж, сет, словарь.
2. Произвести над ними базовые операции (добавление, обращение к элементам) плюс по одной операции, которая не упоминалась в уроке, но есть в документации.
3. При добавлении объекта в список сделать сначала проверку, что такого объекта в списке нету.

foxminded

Урок 3 - Циклы

Цикл for

Вывести в консоль элементы списка через цикл

```
a = [0, 1, 2, 3]
for elem in a:
    print(elem)
```

то же самое что

```
for elem in range(4):
    print(elem)
```

Цикл while

Распечатать последний элемент списка, пока список не будет пуст

```
a = [0, 1, 2, 3]
while len(a) != 0:
    print(a.pop())
```

То же самое, что

```
a = [0, 1, 2, 3]
while a:
    print(a.pop())
```

foxminded

Урок 3 - Функции

Простая функция

```
def say_hello():
    print(“hello”)
```

Функция с аргументами и возвратом значения

```
def sum_elements(a, b):
    return a + b
```

Функция с произвольными элементами

```
def print_profile(*args, **kwargs):
    profile_id = args[0]
    name = kwargs[“name”]
    print(profile_id, name)
```

Вариант с подсказками

```
def sum_elements(a: int, b: int) -> int:
    return a + b
```

Пример вызова

```
print_profile(34, name=”Dan”)
```

foxminded

Урок 3 - Файлы

Открыть файл на чтение текста, прочитать, закрыть

```
f = open('text.txt', 'r')
text = f.read()
f.close()
```

То же самое через контекстный менеджер

```
with open('text.txt', 'r') as f:
    text = f.read()
```

Открыть файл на запись, записать, закрыть

```
with open('text.txt', 'w') as f:
    f.write('some text')
```

Режимы работы с файлами

'r' - чтение текста

'rb' - чтение байтов

'w' - запись текста

'wb' - запись байтов

'a' - добавление текста в конец файла

Дополнительно

https://docs.python.org/3.9/library/functions.html?highlight=open#open

foxminded

Урок 3 - Практика

1. Создать словарь products с минимум 4 продуктами. В словаре ключами будут названия продуктов, а значениями список тегов к этому продукту. Пока в словаре есть продукты, просить у юзера вводить название продукта. Если такой продукт есть в словаре, то через цикл for распечатать теги данного продукта. После этого удалить продукт из словаря. В конце вывести сообщение, что все продукты закончились.

2. Написать функцию, которая принимает список тегов и распечатывает теги в цикле for. Использовать эту функцию в цикле while.

3. Выводить теги не в консоль, а в файл tags.txt. После чего написать юзеру что-то типа Product found. The tags are in file tags.txt.

foxminded

Урок 4 - Классы и объекты

```
class Dog:
    age: int = 3
    name: str = “Jack”
    legs_num: int = 4

    def say_gav(self):
        print('gav')


jack = Dog()
jack.say_gav()
jack.age  # вернет 3
```

```
class Dog:
    legs_num: int = 4  # общее поле для всех собак

    def __init__(self, age: int, name: str):
        self.age = age
        self.name = name

    def say_gav(self):
        print('gav')


jack = Dog(age=5, name=”Rex”)
jack.say_gav()
jack.age  # вернет 5
```

foxminded

Урок 4 - Специальные методы классов

```
class Dog:
    def __init__(self, age: int, name: str):
        self.age = age
        self.name = name

    def say_gav(self):
        print('gav')

    def __str__(self):
        return self.name

    def __repr__(self):
        return 'Dog(age={}, name={})'.format(self.age, self.name)

    def __eq__(self, other):
        return self.age == other.age

    def __gt__(self, other):
        return self.age > other.age
```

Дополнительно:

https://docs.python.org/3/reference/datamodel.html

foxminded

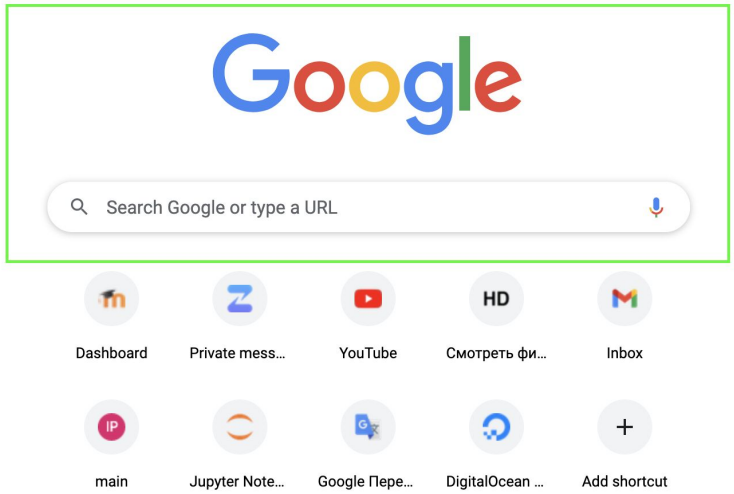
Урок 4 - Практика

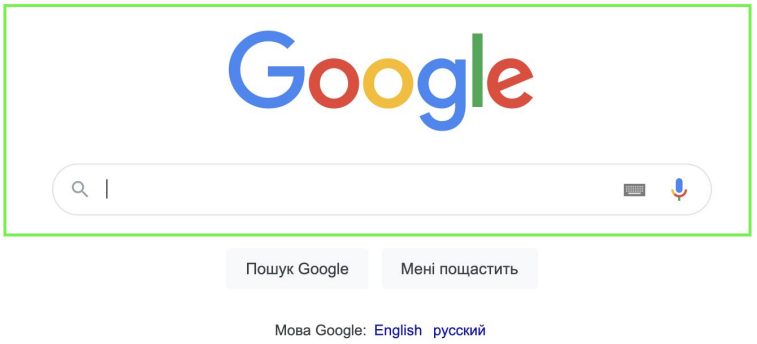
1. Создать класс Building. Добавить к классу характеристики (поля) типа высота (height), ширина (width), длина (length).

2. Добавить специальные методы, чтобы можно было распечатать объект здания. Плюс, чтобы можно было сравнить 2 здания между собой по их объему (у какого здания объем больше, то больше).

foxminded

Урок 5 - Объектно-ориентированное программирование





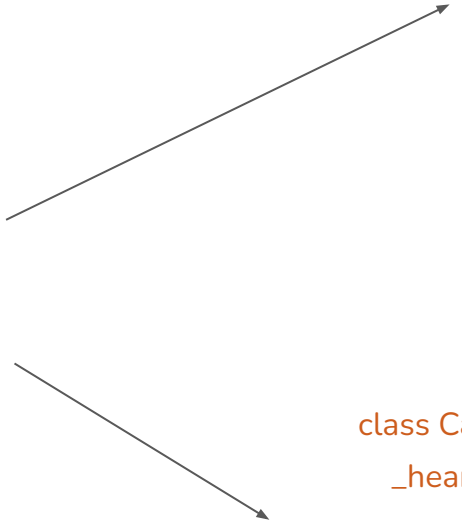
foxminded

Урок 5 - Принципы ООП

```
class Animal(ABC):
    age: int

    @abstractmethod
    def sound(self):
        pass

    def eat(self, food):
        print('Eaten', food)
```



```
class Dog(Animal):
    _heart = 'Big heart'

    def sound(self):
        print('gav')
```

```
class Cat(Animal):
    _heart = 'Small heart'

    def sound(self):
        print('may')
```

foxminded

Урок 5 - Практика

1. Придумать набор объектов и сделать свою иерархию наследования.
2. Добавить скрытые поля к объектам и переопределение методов базового класса.

foxminded

Урок 6 - Декораторы

```
def validate_num(func):

    def wrapper(a, b):

        if type(a) not in (int, float) or type(b) not in (int, float):
            raise ValueError('Value should be of type int or float')

        return func(a, b)

    return wrapper


@validate_num
def add_values(a, b):
    return a + b
```

Вариант сложнее

```
def validate_num(func):

    def wrapper(*args, **kwargs):
        values = args + tuple(kwargs.values())
        for value in values:
            if type(value) not in (int, float):
                raise ValueError('Value should be of type int or float')

        return func(*args, **kwargs)

    return wrapper
```

foxminded

Урок 6 - Контекстные менеджеры

```
class DB:

    def __enter__(self):
        print('connected to db')

    def __exit__(self, exc_type, exc_val, exc_tb):
        print('disconnected from db')


with DB() as db:
    print('some action')
```

foxminded

Урок 6 - Обработка ошибок

```
def add_values(a, b):
    return a + b


try:
    add_values(4, '5')
except TypeError:
    print('Bad arguments')
```

Базовый класс - BaseException

Рабочий базовый класс - Exception

Дополнительно

https://docs.python.org/3/library/exceptions.html

foxminded

Урок 6 - Установка пакетов

Стандартная библиотека python
https://docs.python.org/3/library/

Установка дополнительных пакетов
pip install <package-name>

Пример
pip install flask

Вывод в файл
pip list > requirements.txt

Установка из файла
pip install -r requirements.txt

foxminded

Урок 6 - Практика

1. Придумать и написать свой декоратор.
2. Написать свой контекстный менеджер для работы с файлами, который должен открывать и закрывать файл вовремя.
3. Применить обработку ошибок к открытию файла в контекстном менеджере на случай, если заданного файла на диске нету.

foxminded