Лекция 10

Язык программирования Python.

Хайрулин Сергей Сергеевич

email: s.khairulin@g.nsu.ru, s.khayrulin@gmail.com

Ссылка на материалы

План

- Лекции/практические занятия
  - Тест
- Дифференцированный зачет в конце семестра
  - Защита задания

Литература

Начальный уровень

- Mark Pilgrim. Dive into Python - http://www.diveintopython.net/
- Марк Лутц. Изучаем Python, 4-е издание // Символ-Плюс 2011.
- ...

Стандарт/Документация

- PEP-8 - https://www.python.org/dev/peps/pep-0008/
- https://www.python.org/
- https://github.com/python/cpython

Экспертный уровень

- Лучано Рамальо: Python. К вершинам мастерства
- Mitchell L. Model. Bioinformatics Programming Using Python // O'Reilly 2010.

Версии Python

- Python 2 вышел 2010 году последняя версия 2.7.16 - исправлялись только баги(ошибки) с января 2020 года поддержка прекращена.
- Python 3 в появился в 2008, является актуальной версией языка. Текущая стабильная версия 3.9, в разработке 3.10
  - Python 3 не гарантирует совместимости кода с Python 2

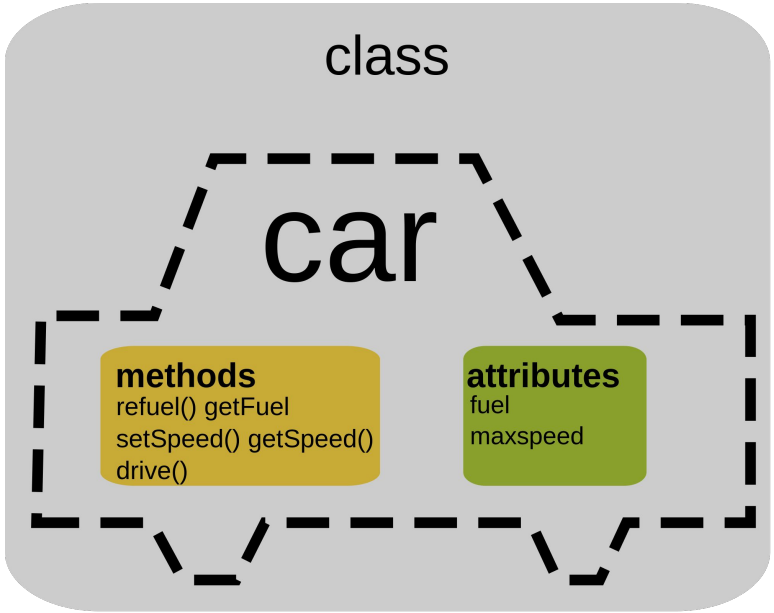
Summary

- Наследование в Python.
  - Единичное,
  - множественное наследование.
- Утиная типизация.
- Полиморфизм в Python.
- Переопределение поведения функции в зависимости от аргументов (*args, **kwargs).
- Статические метода, атрибуты класса.

ООП принципы

- Наследование
  - Возможность создание новых типов данных базирующихся на других, ранее определенных
- Полиморфизм
  - Возможность переопределения поведения базовых свойств объекта (свойств унаследованных от объектов предков)
- Инкапсуляция
  - Возможность скрывать реализацию тех или иных свойств объекта от конечного пользователя

Объект



Classname
(Identifier)
Data Member
(Static attributes)
Member Functions
(Dynamic Operations)

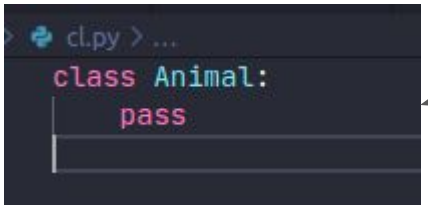
Student	Circle
name grade	radius color
getName() printGrade()	getRadius() getArea()

SoccerPlayer	Car
name number xLocation yLocation	plateNumber xLocation yLocation speed
run() jump() kickBall()	move() park() accelerate()

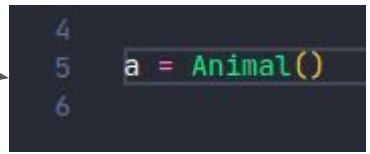
Examples of classes

Ключевое слово class

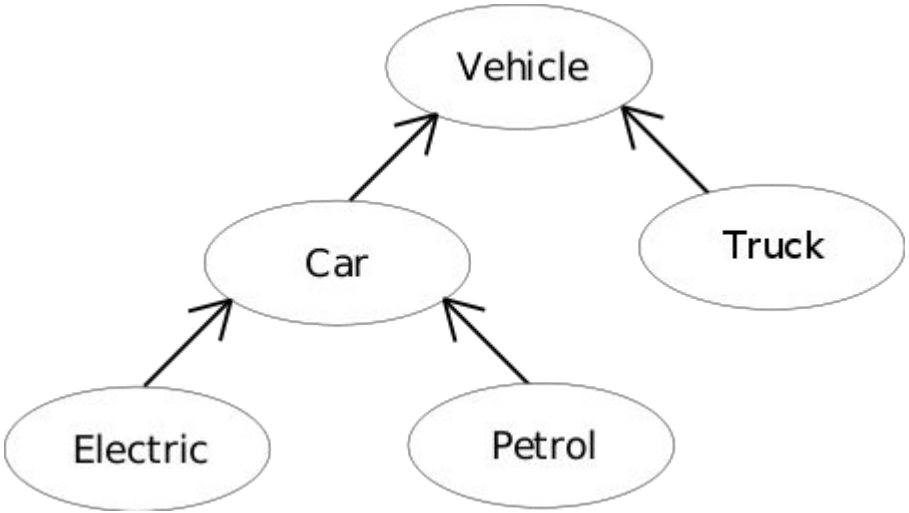
Определение класса

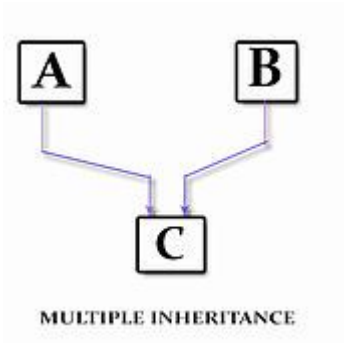


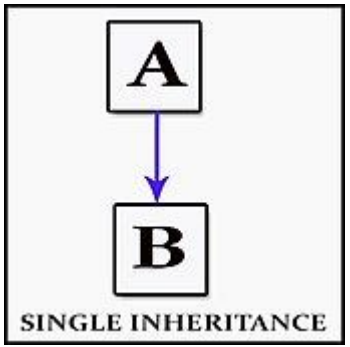
Создание экземпляра класса

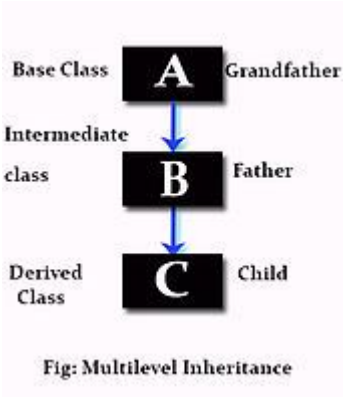


Наследование

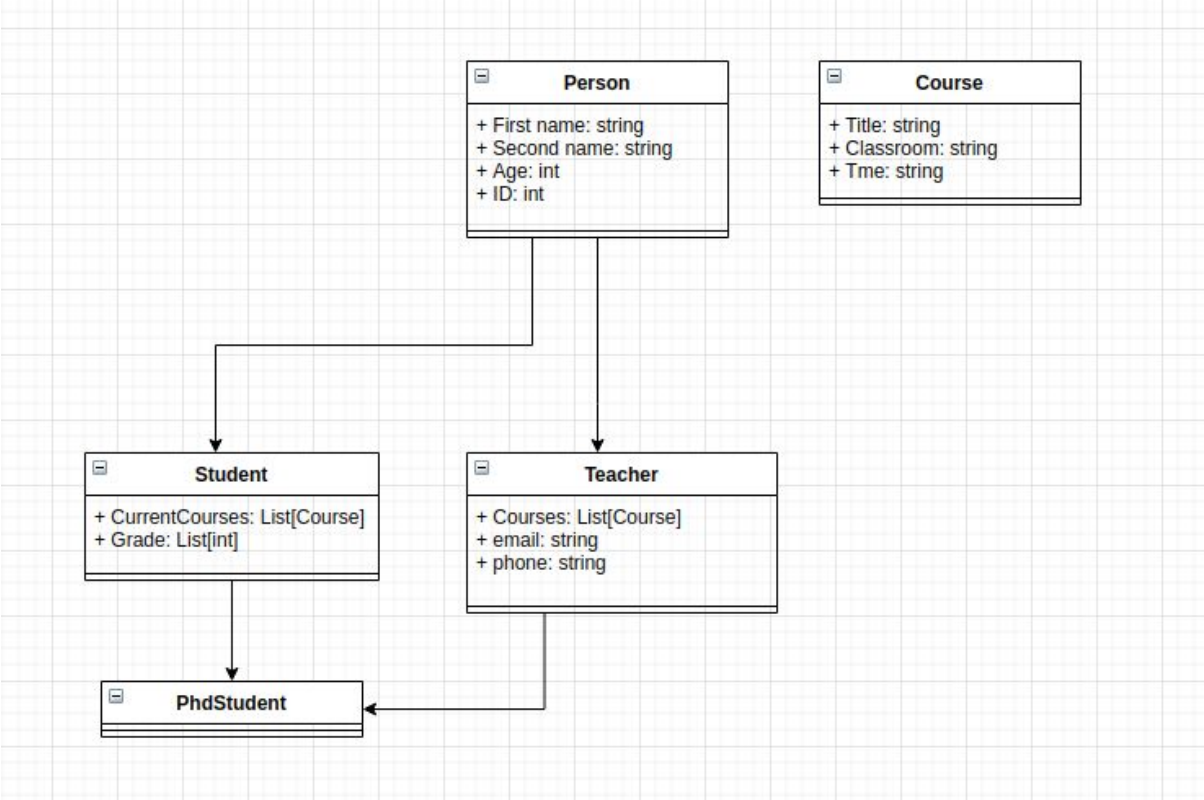








Наследование в Python.



Наследование в Python.

Наследование в Python.

```python
class Course:
    def __init__(self, title, classroom, time):
        self.title = title
        self.classroom = classroom
        self.time = time

    def get_info(self):
        return f"Course - {self.title}, place - {self.classroom}, time - {self.time}"

class Student(Person):
    def __init__(self, first_name, second_name, age, id, current_courses):
        super().__init__(first_name, second_name, age, id)
        self.current_courses = current_courses

    def get_info(self):
        base_info = super().get_info()
        info = f"Student {base_info} \n"
        info += "============Course List============\n"
        for course in self.current_courses:
            info += course.get_info()
        return info
```

Наследование в Python.

```
35
36  if __name__ == '__main__':
37      s = Student("John", "Dow", 23, 1, [Course("Differential equations", 321, "Wen 11:00 AM")])
38      print(s.get_info())
39
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
[22:30:30] serg :: serg-pc  → ~/tmp/oop»
python3 inheritance.py
Student John Dow, age - 23, ID - 1
============Course List============
Course - Differential equations, place - 321, time - Wen 11:00 AM
```

Наследование в Python.

```
36  class Teacher(Person):
37      def __init__(self, first_name, second_name, age, id, courses, email, phone):
38          super().__init__(first_name, second_name, age, id)
39          self.courses = courses
40          self.email = email
41          self.phone = phone
42
43      def get_info(self):
44          base_info = super().get_info()
45          info = f'''Teacher {base_info}, contacts:
46             mail - {self.email}
47             phone - {self.phone}
48  '''
49          info += "============Courses============\n"
50          for course in self.courses:
51              info += course.get_info() + "\n"
52          return info
53
```

Наследование в Python.

```
54
55  if __name__ == '__main__':
56      t = Teacher(
57          "John", "Dow", 47, 2, [
58              Course("Differential equations", 321, "Wen 11:00 AM"),
59              Course("Topology", "412a", "Mon 9:00 AM"),
60          ],
61          "john@dow.com",
62          "123545678"
63      )
64      print(t.get_info())
65
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
[22:41:23] serg :: serg-pc  ➜  ~/tmp/oop»
python3 inheritance.py
Teacher John Dow, age - 47, ID - 2, contacts:
           mail - john@dow.com
           phone - 123545678
============Courses============
Course - Differential equations, place - 321, time - Wen 11:00 AM
Course - Topology, place - 412a, time - Mon 9:00 AM
```

Наследование в Python.

Python поддерживает множественное наследование, множественное наследование позволяет объединять в одном объекте свойства разных объектов.

Наследование в Python.

```python
class Worker:
    def __init__(self, sallary, exp="inf"):
        self.sallary = sallary
        self.exp = "inf"

    def is_expired_contract(self):
        if self.exp == "inf":
            return False
        elif datetime.now() > self.exp:
            return True
        return False


class Teacher(Person, Worker):
    def __init__(self, first_name, second_name, age, id, courses, email, phone, sallary):
        Person.__init__(self, first_name, second_name, age, id)
        Worker.__init__(self, sallary)
        self.courses = courses
        self.email = email
        self.phone = phone

    def get_info(self):
        base_info = super().get_info()
        info = f'''Teacher {base_info}, contacts:
           mail - {self.email}
           phone - {self.phone}
'''
        info += "============Courses============\n"
        for course in self.courses:
            info += course.get_info() + "\n"
        return info

```

Наследование в Python.

```
71
72  if __name__ == '__main__':
73      phd = Teacher(
74          "John", "Dow", 47, 2, [
75              Course("Differential equations", 321, "Wen 11:00 AM"),
76              Course("Topology", "412a", "Mon 9:00 AM"),
77          ],
78          "john@dow.com",
79          "123545678",
80          1000
81      )
82      print(phd.get_info())
83      print(phd.is_expired_contract())
84
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS 1

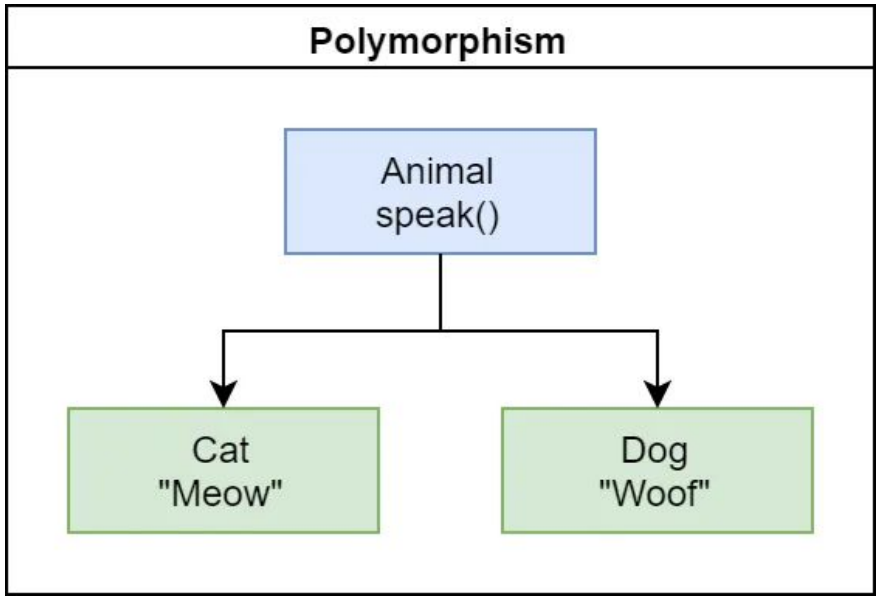
```
[0:29:40] serg :: serg-pc  ➜  ~/tmp/oop»
python3 inheritance.py
Teacher John Dow, age - 47, ID - 2, contacts:
           mail - john@dow.com
           phone - 123545678
============Courses============
Course - Differential equations, place - 321, time - Wen 11:00 AM
Course - Topology, place - 412a, time - Mon 9:00 AM

False
```

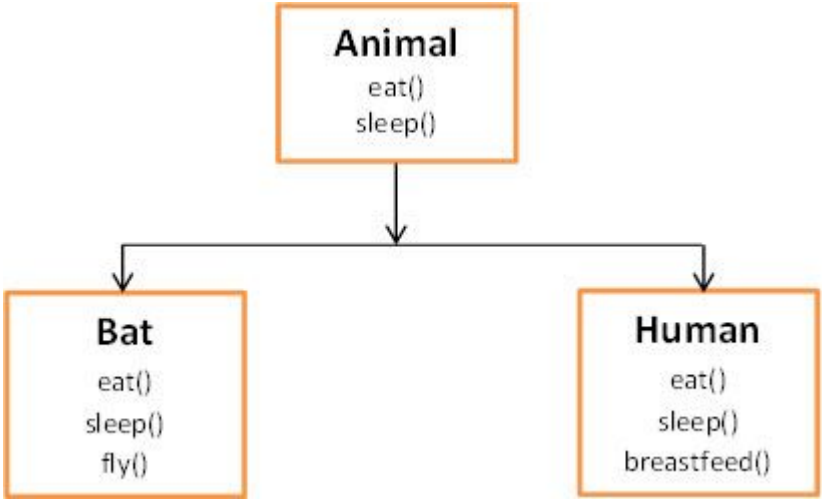
Утиная типизация

Принцип звучит сл. образом - если объект ходит, как утка, плавает, как утка, и крякает, как утка, значит, он утка - это возможно благодаря динамической типизации типов.

Утиная типизация

Полиморфизм





gp.py > Shape > area

```python
class Shape:
    def area(self):
        raise NotImplementedError("Too abstract figure dont't know how to calculate area for it.")


class Circle(Shape):
    def __init__(self, r, x=0, y=0):
        self.center_x = x
        self.center_y = y
        self.radius = r

    def area(self):
        return math.pi * self.radius ** 2


if __name__ == '__main__':
    c = Circle(5)
    print(c.area())

```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS 1

```
[11:49:56] serg :: serg-pc  ➜  ~/tmp/oop»
python3 gp.py
78.53981633974483
```

Полиморфизм

```
17
18
19  class Rectangle(Shape):
20      def __init__(self, h, w, x=0, y=0):
21          self.center_x = x
22          self.center_y = y
23          self.h = h
24          self.w = w
25
26      def area(self):
27          return self.h * self.w
28
29
30  if __name__ == '__main__':
31      r = Rectangle(5, 5)
32      print(r.area())
33
```

```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS 1

[11:56:15] serg :: serg-pc  → ~/tmp/oop»
python3 gp.py
25
```

Переопределение поведения функции в зависимости от аргументов (*args, **kwargs).

Для определение поведения функции в зависимости от аргументов, можно воспользоваться подобной записью см ниже. Где *args - дает доступ к списку неименованных аргументов в тоже время **kwargs - словарь key:value где key - аргумента value - его значение. Таким образом в функцию можно передавать произвольное количество аргументов.

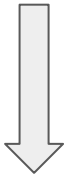
```
1  def f(arg1, arg2, *args, **kwargs):
2      print("Mandatory args ", arg1, arg2)
3      print("List of no named args")
4      for a in args:
5          print(a)
6
7      print("List named args")
8      for a, v in kwargs.items():
9          print(f"{a} = {v}")
```

Переопределение поведения функции в зависимости от аргументов (*args, **kwargs).

*args, **kwargs - могут быть пустыми

```
In [5]: f(1,2)
Mandatory args  1 2
List of no named args
List named args
```

НО обязательные аргументы должны присутствовать всегда



```
In [6]: f(1)
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-281ab0a37d7d> in <module>
----> 1 f(1)

TypeError: f() missing 1 required positional argument: 'arg2'
```

Переопределение поведения функции в зависимости от аргументов (*args, **kwargs).

```
In [8]: f(1,2,3,4,5,6, a=1, b=2)
Mandatory args  1 2
List of no named args
3
4
5
6
List named args
a = 1
b = 2
```

Статические метода, атрибуты класса.

Иногда требуется определить одинаковое поведение для всех экземпляров класса. Это можно сделать с помощью статических полей и методов класса. Статические методы не имеют ссылку на конкретный экземпляр класса - отсюда статический метод никак не может изменить состояние какого-либо конкретного экземпляра этого класса.

Статические метода, атрибуты класса.

Статические метода, атрибуты класса.

stat.py > ...

```python
class A:
    COUNTER = 0  # static field

    def __init__(self):
        A.COUNTER += 1

    @staticmethod
    def do_common_work():
        print(f"Currently {A.COUNTER} examplars was created.")


if __name__ == '__main__':
    a1 = A()
    A.do_common_work()
    a2 = A()
    A.do_common_work()
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
[12:41:03] serg :: serg-pc  ➜ ~/tmp/oop»
python3 stat.py
Currently 1 examplars was created.
Currently 2 examplars was created.
```

Практическая часть

1. Наследование