



8-11 классы

## Программирование на Python

Презентация занятия

### **Повторение ООП. Решение задач с ООП.**

43 занятие



**инжинириум®**

МГТУ им. Н.Э. Баумана

2023

## ООП. Создание класса

### Синтаксис

```
class <Название класса>:  
    <атрибут класса 1>  
    <атрибут класса 2>  
    ...  
    <метод класса 1>  
    <метод класса 2>  
    ...
```

### Пример №1

```
1 class Car:  
2     # создаем атрибуты класса  
3     name = "c200"  
4     make = "mercedes"  
5     model = 2008  
6  
7     # создаем методы класса  
8     def start(self):  
9         print("Start")  
10  
11     def stop(self):  
12         print("End")
```

### Пример №2

```
1 class Car:  
2     # создаем атрибуты класса  
3     name = "c200"  
4     make = "mercedes"  
5     model = 2008  
6  
7     # создаем методы класса  
8     def start(self):  
9         print("Start")  
10  
11     def stop(self):  
12         print("End")  
13  
14     car_1 = Car()  
15     car_1.start()  
16     x = car_1.name  
17     print(x)
```

### Вывод

Start  
c200

**self** – общепринятое имя для ссылки на объект, в контексте которого вызывается метод.



## ООП. Атрибуты класса и экземпляра

### Пример

```
1 class Car:
2     # создаем атрибуты класса
3     car_count = 0
4
5     def start(self, a, b, c):
6         print("Start")
7         self.name = a
8         self.make = b
9         self.model = c
10        Car.car_count += 1
11
12 car_1 = Car()
13 car_1.start("Corrola", "Toyota", 2015)
14 print(car_1.name)
15 print(car_1.car_count)
```

### Вывод

```
Start
Corrola
1
```

**Атрибуты класса** делятся среди всех объектов класса, в то время как **атрибуты экземпляров** являются собственностью экземпляра.

## ООП. Статические и классовые методы

### Статические

```
1 class Car:
2     @staticmethod
3     def start():
4         print("Start")
5
6
7 Car.start()
8 car_1 = Car()
9 car_1.start()
```

### Вывод

Start  
Start

### Классовые

```
1 class Car:
2     @classmethod
3     def start(cls):
4         print("Class {}".format(cls.__name__))
5
6
7 Car.start()
```

### Вывод

Class Car

## ООП. Конструкторы

### Синтаксис

```
class <название класса>:  
    def __init__(self):  
        ...
```

### Пример

```
1 class Car:  
2     car_count = 0  
3  
4     def __init__(self):  
5         Car.car_count += 1  
6         print(Car.car_count)  
7  
8     car_1 = Car()  
9     car_2 = Car()
```

**Конструктор** — это специальный метод, который вызывается по умолчанию когда вы **создаете объект класса**

### Вывод

1  
2



## ООП. Локальные и глобальные переменные

### Пример

```
1 class Car:
2     a = "Сообщение 1"
3
4     def start(self):
5         b = "Сообщение 2"
6
7 car_1 = Car()
8 print(car_1.a)
9 print(car_1.b)
```

**Глобальные и локальные  
переменные отличаются своими  
областями видимости**

### Вывод

Сообщение 1

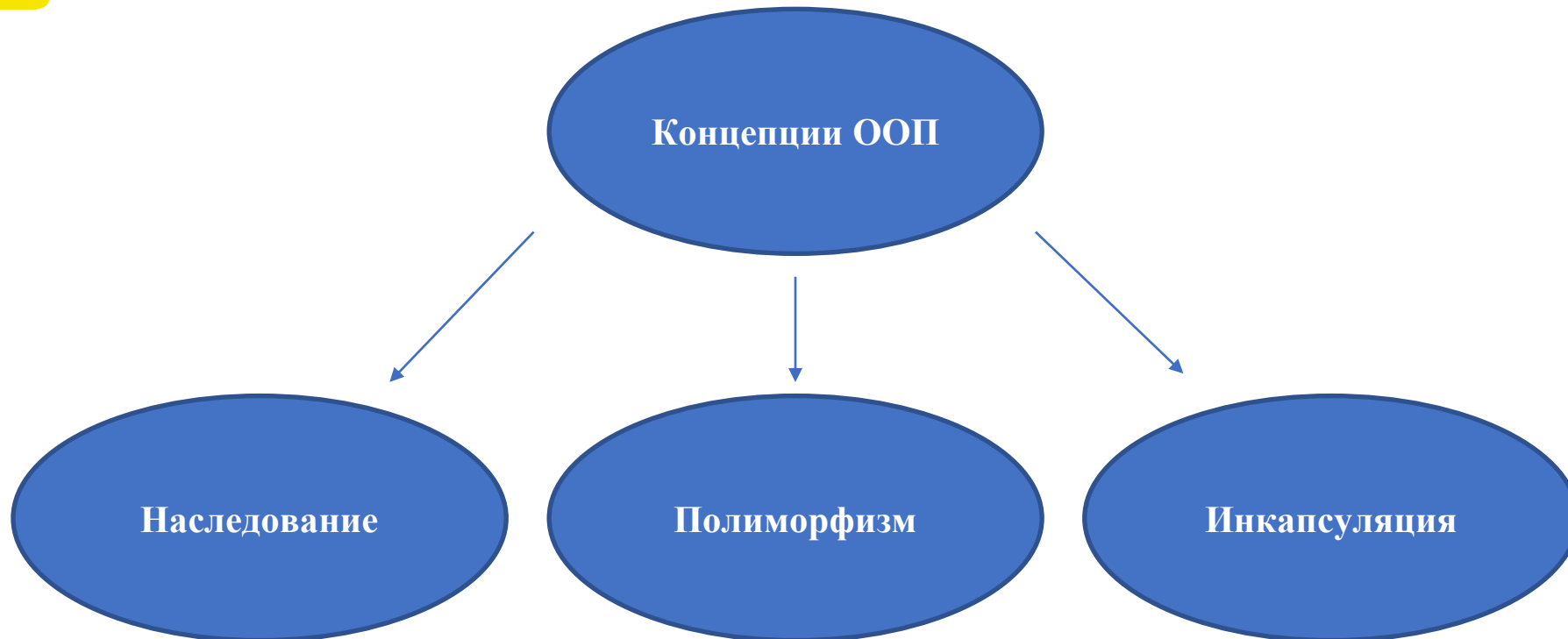
Traceback (most recent call last):

File "C:\Users\temud\PycharmProjects\pythonProject3\task\_2.py", line 9, in <module>

print(car\_1.b)

AttributeError: 'Car' object has no attribute 'b'

## Концепции ООП



## ООП. Наследование

### Пример

```
1 class Machine:
2     def function_1(self):
3         print("Это родительский метод из класса Machine")
4
5 class Car(Machine):
6     def function_2(self):
7         print("Это метод из дочернего класса Car")
8
9 car_1 = Car()
10 car_1.function_1()
```

Класс может наследовать  
характеристики другого класса

### Вывод

Это родительский метод из класса Machine

```
1 class Machine:
2     def function_1(self):
3         print("Это родительский метод из класса Machine")
4
5 class Car():
6     def function_2(self):
7         print("Это метод из дочернего класса Car")
8
9 class Toyota(Machine, Car):
10     def function_3(self):
11         print("Это метод из дочернего класса Toyota")
12
13 car_1 = Toyota()
14 car_1.function_1()
15 car_1.function_2()
```

### Вывод

Это родительский метод из класса Machine

Это метод из дочернего класса Car





## ООП. Полиморфизм

### Перегрузка метода

```
1 class Car:
2     def start(self, a, b=None):
3         if b is not None:
4             print(a + b)
5         else:
6             print(a)
7
8 car_1 = Car()
9 car_1.start(5)
```

### Вывод

5

**Перегрузка метода** относится к свойству метода вести себя по-разному, в зависимости от количества или типа параметров.

**Переопределение метода** относится к наличию метода с одинаковым названием в дочернем и родительском классах.

### Переопределение метода

```
1 class Vehicle:
2     def print_details(self):
3         print("Это родительский метод из класса Vehicle")
4
5 class Car(Vehicle):
6     def print_details(self):
7         print("Это дочерний метод из класса Car")
8
9 class Toyota(Vehicle):
10    def print_details(self):
11        print("Это дочерний метод из класса Toyota")
```

```
13 car_a = Vehicle()
14 car_a.print_details()
15
16 car_b = Car()
17 car_b.print_details()
18
19 car_c = Toyota()
20 car_c.print_details()
```

Это родительский метод из класса Vehicle  
Это дочерний метод из класса Car  
Это дочерний метод из класса Toyota

### Вывод



## ООП. Инкапсуляция

### Вывод

```
1 class Car:
2     def __init__(self):
3         self.name = "corolla"
4         self.__make = "toyota"
5         self._model = 1999
6
7     car_a = Car()
8     print(car_a.name)
9     print(car_a._model)
10    print(car_a.make)
```

```
corolla
1999
```

```
Traceback (most recent call last):
```

```
File "C:\Users\temud\PycharmProjects\pythonProject3\task_2.py", line 10, in <module>
```

```
    print(car_a.make)
```

```
AttributeError: 'Car' object has no attribute 'make'
```

**Инкапсуляция** означает скрытие данных

- **attribute** – публичное свойство (**public**);
- **\_attribute** – режим доступа **protected** (служит для обращения внутри класса и во всех его дочерних классах)
- **\_\_attribute** – режим доступа **private** (служит для обращения только внутри класса).



## Практическая часть

Задания:

1) Создайте класс Juice, принимающий 1 аргумент (отвечающий за возможную добавку в сок) при инициализации объекта. В этом классе реализуйте метод my\_juice(), выводящий на печать «Сок с {название добавки}» в случае наличия добавки, а иначе отобразится следующая фраза: «Сок без добавки».

2) Напишите программу с классом Student, в котором есть три атрибута: name, group\_number и age. По умолчанию name = Ivan, age = 18, groupNumber = 10A.

Необходимо создать пять методов: getName, getAge, getGroupNumber, setNameAge, setGroupNumber.

Метод getName нужен для получения данных об имени конкретного студента.

Метод getAge нужен для получения данных о возрасте конкретного студента.

Метод setGroupNumber нужен для получения данных о номере группы конкретного студента.

Метод setNameAge позволяет изменить данные атрибутов установленных по умолчанию.

Метод setGroupNumber позволяет изменить номер группы установленный по умолчанию. В программе необходимо создать пять экземпляров класса Student, установить им разные имена, возраст и номер группы.



## Практическая часть

Задание:

1. Создайте класс *Phone*, который содержит атрибуты *number*, *model* и *weight*.
  2. Создайте три экземпляра этого класса.
  3. Выведите на консоль информацию о телефоне(`__str__` или реализовать метод `print_data`)
  4. Добавить в класс *Phone* методы:
    - *receive\_call*, имеет один параметр – имя звонящего. Выводит на консоль сообщение “Звонит {name} на {number}”.
    - *get\_number* – возвращает номер телефона.
- Вызвать эти методы для каждого из объектов.
5. Добавить конструктор в класс *Phone*, который принимает на вход три параметра для инициализации переменных класса - *number*, *model* и *weight*.
  6. Учесть все случаи вызова конструктора:
    - Вызов с 3 параметрами, *number*, *model* и *weight*
    - Вызов с 2 параметрами, *number*, *model*
    - Вызов с 1 параметром, *number*

