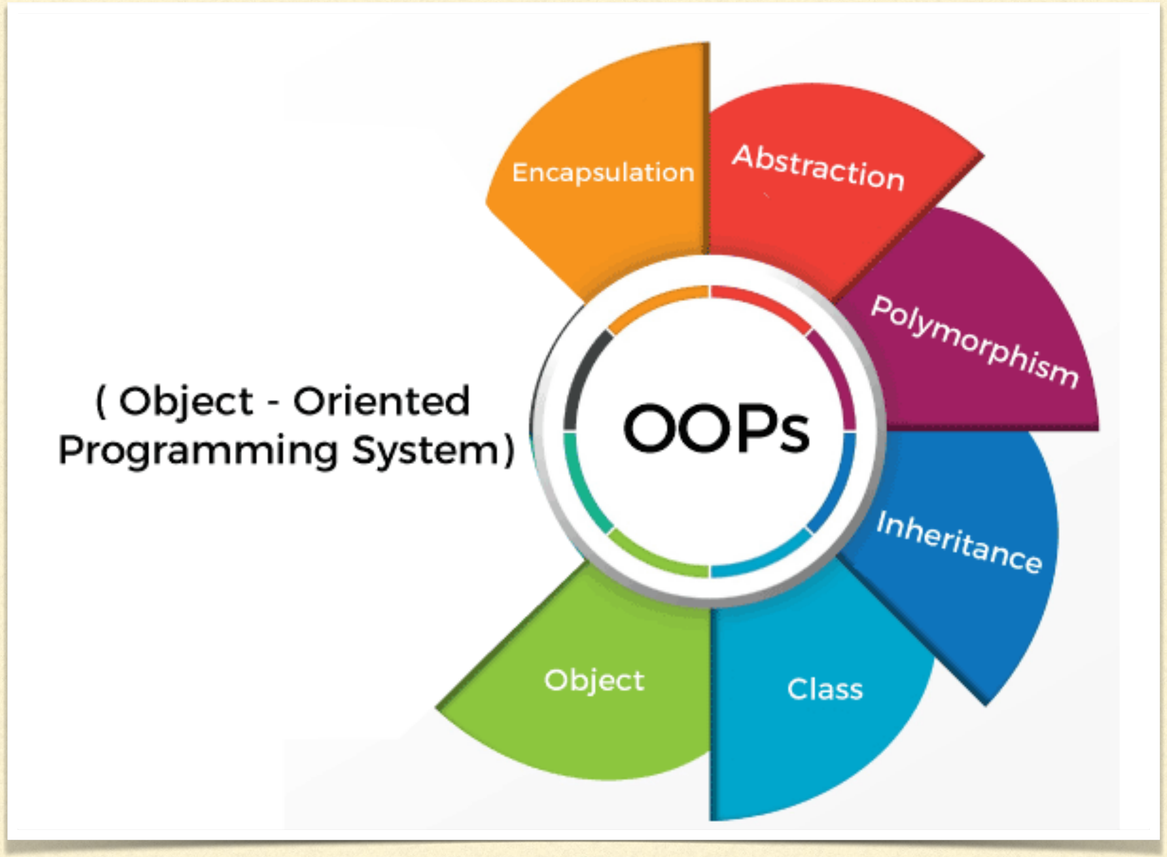
ПРОГРАММИРОВАНИЕ В PYTHON

6. Объектно-ориентированное программирование (ООП)

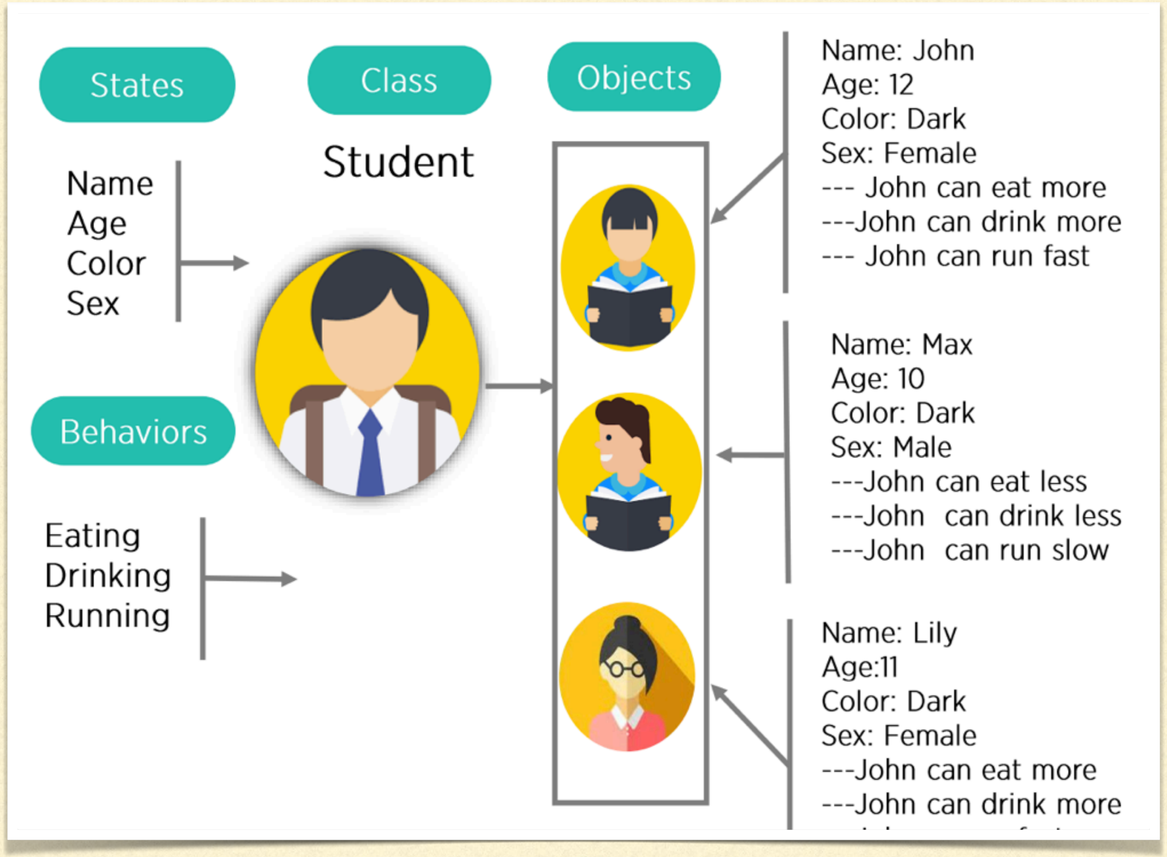
ЧТО ТАКОЕ ООП?



- ООП - методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

python™

ЧТО ТАКОЕ КЛАСС?



- **Класс** хранит некоторые элементы данных, которые являются общими для всех экземпляров этого класса
- **Экземпляры** - это созданные объекты, которые следуют определению, данному внутри класса

python™

АТРИБУТЫ ДАННЫХ И КЛАССОВ

```
class Point(object):
    def __init__(self, x, y, z):
        self.coord = (x, y, z)
```

```
class SomeClass(object):
    attr1 = 42
    attr2 = "Hello, World"
```

- **Атрибуты данных**:
  - Переменная, принадлежащая определенному экземпляру класса
  - У каждого экземпляра свое собственное значение атрибута
- **Атрибуты класса**:
  - Принадлежат всему классу
  - У всех экземпляров одно и то же значение атрибута

СОЗДАНИЕ КЛАССА

```
class Student:

    #A class representing a student
    def __init__(self,name,gpa):
        self.full_name = name
        self.gpa = gpa
    def get_gpa(self):
        return self.gpa
```

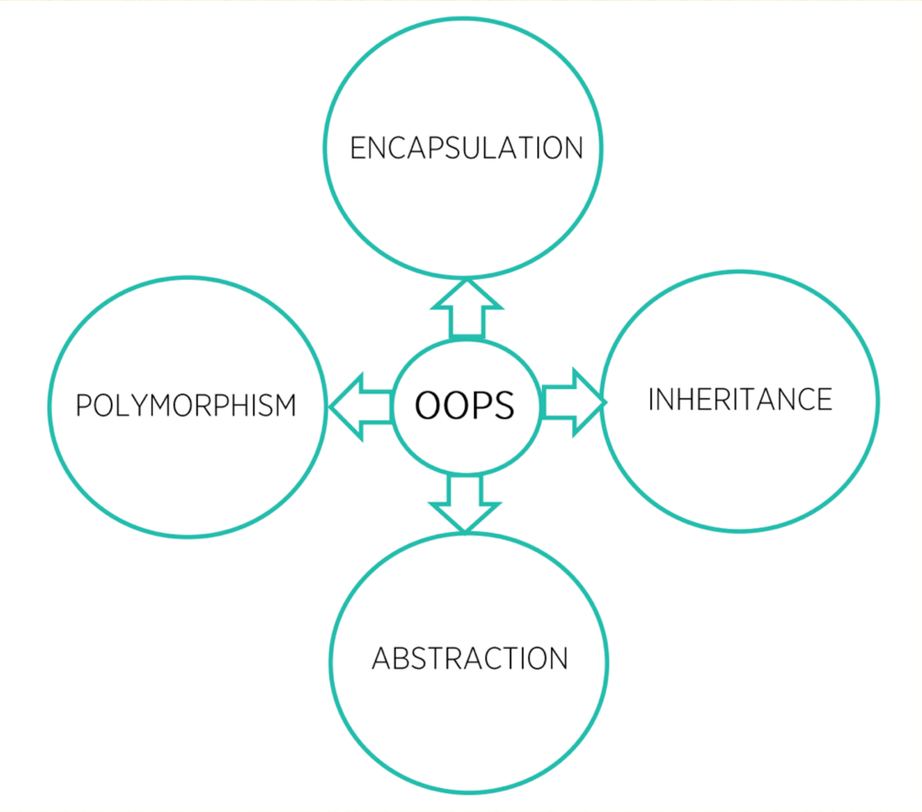
- Пишется ключевое слово **class** далее идет название класса и двоеточие. ВАЖНО ЗАМЕТИТЬ, что внутренности класса пишутся после одного отступа (1 tab).
- Конструктор **__init__**() принимает в себя любое количество аргументов, одним из которых должен быть **self**

СОЗДАНИЕ ЭКЗЕМПЛЯРА КЛАССА

```
class Point(object):
    def __init__(self, x, y, z):
        self.coord = (x, y, z)

p = Point(13, 14, 15)
p.coord # (13, 14, 15)
```

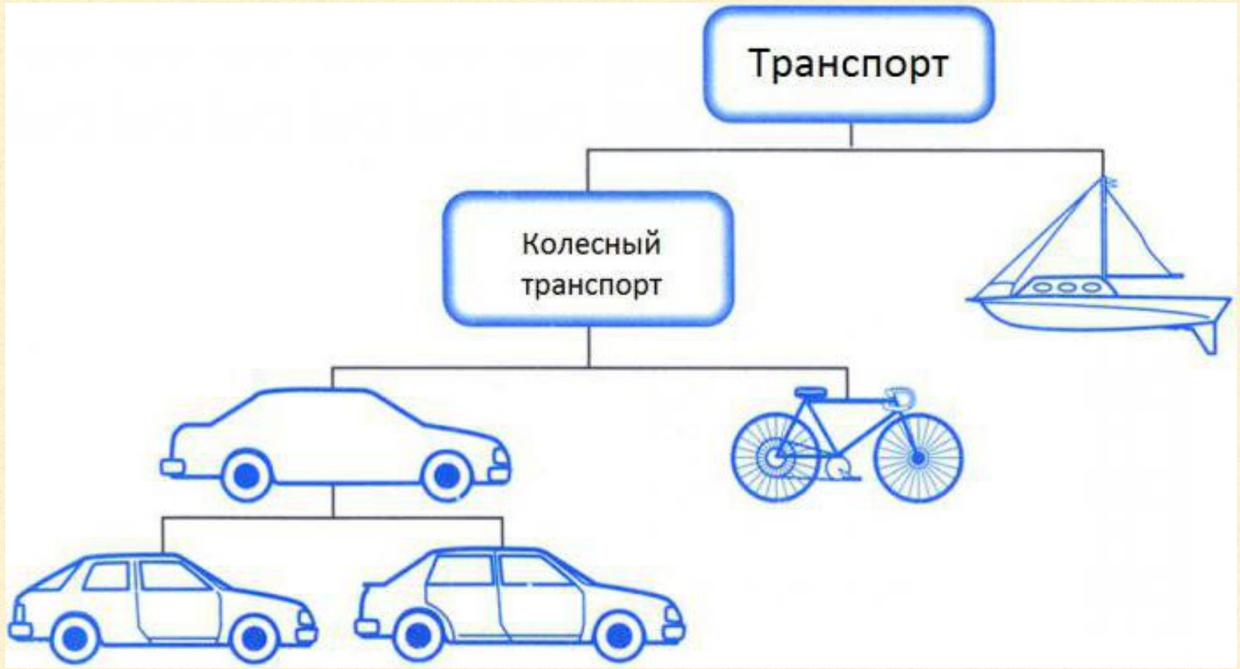
ОСНОВНЫЕ ПРИНЦИПЫ ООП



- Наследование
- Инкапсуляция
- Полиморфизм
- Абстракция

python™

НАСЛЕДОВАНИЕ



- Используя наследование, мы может определить ряд дочерних классов. Они же будут наследовать свойства и методы базового. Таким образом, мы расширяем функциональность базового класса.

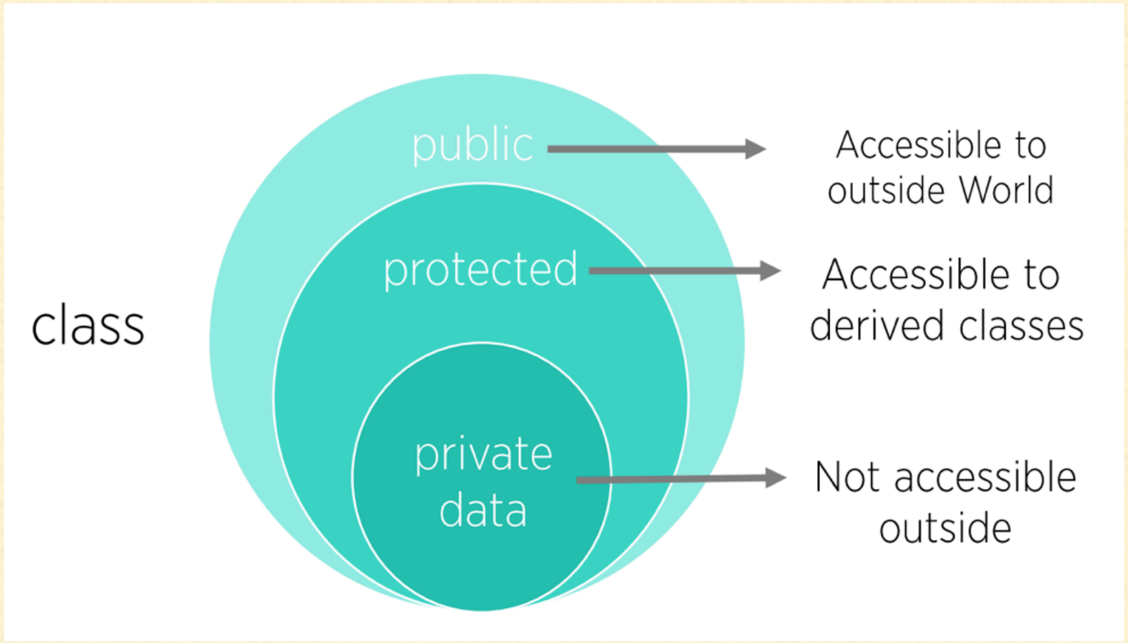
```
class Mammal():
    className = 'Mammal'

class Dog(Mammal):
    species = 'Canis lupus'

dog = Dog()
dog.className # Mammal
```

python™

ИНКАПСУЛЯЦИЯ



- Инкапсуляция - это механизм, который ограничивает прямой доступ к данным и методам объектов

- публичный (`public` , нет особого синтаксиса, `publicBanana`);
- защищенный (`protected` , одно нижнее подчеркивание в начале названия, `_protectedBanana`);
- приватный (`private` , два нижних подчеркивания в начала названия, `__privateBanana`).

```
class Phone:
    username = "Kate"                   # public variable
    __how_many_times_turned_on = 0      # private variable

    def call(self):                     # public method
        print( "Ring-ring!" )

    def __turn_on(self):                # private method
        self.__how_many_times_turned_on += 1
        print( "Times was turned on: ", self.__how_many_times_turned_on )
```

ПОЛИМОРФИЗМ

```
class Animal(object):
    def __init__(self, name):
        self.name = name
    def talk(self):
        raise NotImplementedError
class Dog(Animal):
    def talk(self):
        return "Bow...Bow..."
class Cat(Animal):
    def talk(self): return "Meow...Meow..."
```

- Дочерние классы могут их переопределять и решать одну и ту же задачу разными путями, а конкретная реализация будет выбрана только во время исполнения программы.

python™

АБСТРАКЦИЯ

```
from abc import abstractmethod, ABC

class Vehicle(ABC):
    def __init__(self, speed, year):
        self.speed = speed
        self.year = year

    def start(self):
        print("Starting engine")

    def stop(self):
        print("Stopping engine")

    @abstractmethod
    def drive(self):
        pass

class Car(Vehicle):
    def __init__(self, canClimbMountains, speed, year):
        Vehicle.__init__(self, speed, year)
        self.canClimbMountains = canClimbMountains

    def drive(self):
        print("Car is in drive mode")
```

- Абстракция - это методология программирования, в которой детали программных кодов скрыты от пользователя, и пользователю отображаются только самые важные вещи
- Для работы с абстрактными классами необходимо подключить специальный модуль ABC

ВРЕМЯ ПРАКТИКИ!