ООП:

Основное:

Основные понятия:

Класс- шаблон на основе которого будет создаваться объект. По своим свойствам напоминает тип данных. Пример создания класса человек.

```
class Person:
        pass
```

Объект - переменная (конкретно в языке Python любая переменная - объект), созданная на основе шаблона. Пример создания объекта.

```
person=Person()
# person объект, а Person() -класс
```

Пока выглядит бесполезно . Нужно добавить функциональности.

Метод- функция, определенная лишь в конкретном типе данных. Что это значит и чем это отличается от функции ? Разберем на конкретном примере.

```
simple_list=list([1,2,3]) # создаем переменную с типом данных list он же класс list
 # simple_list объект, а list() -класс
```

Допустим, мы хотим добавить в наш список еще один элемент. Что мы сделаем ? Воспользуемся методом .append.

```
simple_list.append(4)
print(simple_list)
# output : [1,2,3,4]
```

Попробуем воспользоваться этим же методом с другим типом данных, например, int.

```
a=int(10)
a.append(20)
# output : AttributeError: 'int' object has no attribute 'append'
```

Какой из этого можно сделать вывод ? **Метод может быть вызван только с конкретным типом данных/ классом, тогда как функция может быть вызвана для любого типа.**

Вернемся к нашему классу Person.

```
class Person:
        def move(self):
                print('move')
```

move - метод условного движения , методы создаются также как и функции, но должны иметь обязательный первый аргумент self. Его мы использовать не будем, но его **обязательно** нужно указать в скобках, такой синтаксис.

```
person.move() # вызываем метод move
```

Поле/Атрибут- переменная внутри класса. В чем ее преимущество ? Разберем на примере.

Допустим, мы хотим описать человека. Какие данные нам нужны ? Пусть это будут имя и возраст . Сначала реализуем это уже знакомыми нам средствами.

```
name='Misha'
age=22
```

Мы создали 2 переменные. Кажется, что с помощью этого мы можем описать человека, но нет. Что если нам нужно описать 10, 1000, 100000 человек ? Для этого нам стоит структурировать данные с помощью ООП.

```
class Person:
    name='Misha'
    age=22 # name и age поля класса
    def move(self):
        print('move')
person1=Person()# person - это объект, а Person -класс
person2=Person()
person2.name='Vasya'
person2.age=31
print(person1.name,person1.age)
print(person2.name,person2.age)
# output : Misha 22 \n Vasya 31
```

Конструкторы:

```
class Person:
    def __init__(self,name,age): # __init__ - конуструктор.Его имя обязательно должно быть таким  . self.name -поле класса, name -аргумент
        self.name=name
        self.age=age
person=Person('Misha',22)# person - это объект, а Person -класс
print(person.name,person.age) # результат - Misha 22
```

Исчерпывающий источник : https://www.yuripetrov.ru/edu/python/ch_10_01.html