Лекция 4

Объектно-ориентированное программирование

1 Классы и объекты

class имя_класса:

атрибуты класса

методы класса

имя_объекта = имя_класса()

```
14  # Создаем объект класса People
15  # под названием P1
16  P1 = People()
17
18  # Создаем объект класса People
19  # под названием P2
20  P2 = People()
```

```
1   # Создаем класс People
2   class People:
3
4       # создаем атрибуты класса
5       name = "Alex"
6       age = 20
7
8       # создаем методы класса
9       def Print_Inf(self):
10          print ("Класс People")
11      def Get_Year(self):
12          return 2019-self.age
```

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

```python
# Создаем класс People
class People:

    # создаем атрибуты класса
    name = "Alex"
    age = 20

    # создаем методы класса
    def Print_Inf(self):
        print ("Класс People")
    def Get_Year(self):
        return 2019-self.age

# Создаем объект класса People
# под названием P1
P1 = People()

# Создаем объект класса People
# под названием P2
P2 = People()

print(type(P1))

P1.Print_Inf()
print("P1.age = ", P1.Get_Year())

P2.age = 30;
print("P2.age = ", P2.Get_Year())
```

```
<class '__main__.People'>
Класс People
P1.age =  1999
P2.age =  1989
```

2 Атрибуты класса и атрибуты экземпляра

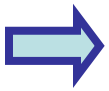
```
['Get_Year', 'Print_Inf', '__class__', '__delattr__', '__dict__', '__dir__',
'__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '_
_hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__module__',
'__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__'
, '__sizeof__', '__str__', '__subclasshook__', '__weakref__', 'age', 'name']

Get_Year
Print_Inf
__class__
__delattr__
__dict__
__dir__
__doc__
__eq__
__format__
__ge__
__getattribute__
__gt__
__hash__
__init__
__init_subclass__
__le__
__lt__
__module__
__ne__
__new__
__reduce__
__reduce_ex__
__repr__
__setattr__
__sizeof__
__str__
__subclasshook__
__weakref__
age
name
```

```python
# Создаем класс People
class People:

    # создаем атрибуты класса
    name = "Alex"
    age = 20

    # создаем методы класса
    def Print_Inf(self):
        print ("Класс People")
    def Get_Year(self):
        return 2019-self.age

# Создаем объект класса People
# под названием P1
P1 = People()

print(dir(P1),'\n')

# построчный вывод списка
for x in dir(P1): print(x)
```

dir() - просмотр всех атрибутов и методов класса или объекта

Метод __str__

```
1  class People:
2
3      name = "Alex"
4      age = 20
5
6  P1 = People();
7  print ( P1 )
```

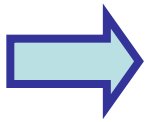


```
<__main__.People object at 0x7ff009408c50>
```

использование объекта в качестве строки вызывает метод __str__, который по умолчанию выводит локацию памяти объекта

Переопределение метода __str__

```
1  class People:
2
3      name = "Alex"
4      age = 20
5
6      def __str__(self):
7          return self.name +' '+ str(self.age)
8
9  P1 = People();
10 print ( P1 )
```



```
Alex 20
```

```python
# Создаем класс People
class People:

    name = "Alex"
    age = 20
    def Get_Year_1(self):
        return 2019-self.age
    def Get_Year_2(self):
        return 2019-People.age
    def Set_Data(self, name1, surname1, age1):
        self.name = name1
        self.surname = surname1
        self.age = age1
        People.age+=1
    def Print_Data_1(self):
        print(self.name, self.surname, self.age)
    def Print_Data_2(self):
        print(People.name, self.surname, People.age)

# Создаем объект класса People
# под названием P1
P1 = People()
P1.Set_Data('Ivan',"Ivanov", 30)
print( "строка 24:", P1.Get_Year_1() )
print( "строка 25:", P1.Get_Year_2() )
P1.Print_Data_1()
P1.Print_Data_2()
print ("строка 28:", P1.surname)
print ("строка 29:", P1.age)

P2 = People()
print ("строка 32:", P2.age)
```

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

```
строка 24: 1989
строка 25: 1998
Ivan Ivanov 30
Alex Ivanov 21
строка 28: Ivanov
строка 29: 30
строка 32: 21
```

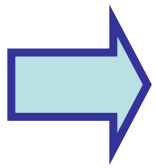
3 Методы

Статичный метод:

```
1  class People:
2
3      name = "Alex"
4      age = 20
5
6      def Set_Data(self, name1, surname1, age1):
7          self.name = name1
8          self.surname = surname1
9          self.age = age1
10
11     def Print_Data_1(self):
12         print(self.name, self.surname, self.age)
13
14     @staticmethod
15     def Print_Data_2():
16         print(People.name, People.age)
17
18 P1 = People()
19 P1.Set_Data('Ivan',"Ivanov", 30)
20
21 P1.Print_Data_1()
22 People.Print_Data_2()
```

Возврат множественных значений из метода:

```
1  class Square:
2
3      @staticmethod
4      def get_squares(a, b):
5          return a*a, b*b
6
7      def get_xyz(self, x, y, z):
8          return x+10, y+20, z+30
9
10 print(Square.get_squares(3, 5))
11 s = Square()
12 print(s.get_xyz(1,2,3))
```



```
(9, 25)
(11, 22, 33)
```

4 Конструктор

Конструктор класса определяется методом __init__

```
1  class People:
2
3      name = "Alex"
4      age = 20
5
6      def __init__(self, name1, age1):
7          self.name = name1
8          self.age = age1
9
10     def __str__(self):
11         return self.name +' '+ str(self.age)
12
13 P1 = People("Ivan", 40);
14 print ( P1 )
15
16 P2 = People("Olga", 20);
17 print ( P2 )
```

```
Ivan 40
Olga 20
```

```
1  class People:
2
3      name = "Alex"
4      age = 20
5
6      def __init__(self, name1 = "Helen", age1 = 10 ):
7          self.name = name1
8          self.age = age1
9
10     def __str__(self):
11         return self.name +' '+ str(self.age)
12
13 P1 = People("Ivan", 40);
14 print ( P1 )
15
16 P2 = People("Olga");
17 print ( P2 )
18
19 P3 = People();
20 print ( P3 )
```

```
Ivan 40
Olga 10
Helen 10
```

5 Инкапсуляция

Публичный — **public**

Приватный — **private** (префикс двойного нижнего подчеркивания)

Защищенный — **protected** (префикс нижнего подчеркивания)

```
1  class People:
2
3      # публичный атрибут
4      name = "Alex"
5
6      # приватный атрибут
7      __age = 20
8
9      # защищенный атрибут
10     _surname = 'Ivanov'
```

```python
class People:

    name = "Ivan"
    _age = 20

    # открытый метод
    def F1(self):
        print("Открытый метод F1")

    # закрытый метод
    def __F2(self):
        print("Закрытый метод F2")

    # открытый метод
    def F3(self):
        print("Открытый метод F3 с вызовом закрытого")
        self.__F2()

P1 = People()
P1.F1()

# ошибка
# P1.F2()

P1.F3()
```

```
Открытый метод F1
Открытый метод F3 с вызовом закрытого
Закрытый метод F2
```

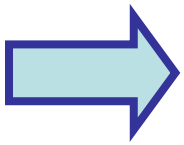
```python
# создаем класс Car
class Car:

    # создаем конструктор класса Car
    def __init__(self, model):
        # Инициализация свойств.
        self.model = model

    # создаем свойство модели
    @property
    def model(self):
        return self.__model

    # Сеттер для создания свойств
    @model.setter
    def model(self, model):
        if model < 2000:
            self.__model = 2000
        elif model > 2018:
            self.__model = 2018
        else:
            self.__model = model

    def getCarModel(self):
        return "Год выпуска модели " + str(self.model)

carA = Car(2088)
print(carA.getCarModel())
```

```
Год выпуска модели 2018
```

6 Наследование

class имя_наследника (имена_родительских_классов) :
атрибуты
методы

```
1  class People:
2      name = "Ivan"
3      _age = 20
4
5      def __init__(self, name1 = "Ivan", age1 = 20):
6          self.name = name1
7          self.age = age1
8
9      def Print_People(self):
10         print(self.name, self.age)
11
12 class Manager(People):
13     id = 1
14
15     def Print_Meneger(self):
16         self.Print_People()
17         print(self.id)
18
19 P = People("Petr",30)
20 P.Print_People()
21 print()
22 M = Manager()
23 M.name = "Olga"
24 M.Print_Meneger()
25 M.Print_People()
```



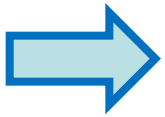
```
Petr 30

Olga 20
1
Olga 20
```

7 Полиморфизм

Полиморфи́зм — возможность объектов с одинаковой спецификацией иметь различную реализацию.

Переопределение метода

```
1  class People:
2      name = "Ivan"
3      _age = 20
4
5      def Inf(self):
6          print('Класс People, метод Inf')
7
8      def Print(self):
9          print('Класс People, метод Print')
10
11 class Student (People):
12     id = 1
13
14     def Print(self):
15         print("Класс Student, метод Print")
16
17 P = People()
18 P.Print()
19
20 S = Student()
21 S.Inf()
22 S.Print()
```



```
Класс People, метод Print
Класс People, метод Inf
Класс Student, метод Print
```