**КЛАССЫ**

|  |  |
| --- | --- |
|  |  |
| **def \_\_init\_\_(self, name, pay):**  **self.name = name**  **self.pay = pay** | Функция КОНСТРУКТОР экземпляра класса |
| **def \_\_call\_\_(self, label):**  **print(label, self.state)**  **self.state += 1** | *Функция автоматически вызывается при вызове экземпляра класса в виде функции. Например Clss\_ekz(“Hello world”)* |
| **Def \_\_add\_\_(self, other):**  **return ThirdClass(self.data + other)** | **\_\_add\_\_** выполняется, когда экземпляр ThirdClass присутствует в выражении +; *Вызывается для self + other* |
| **Def \_\_str\_\_(self):**  **return ’[ThirdClass:%s]'%self.data** | **\_\_str\_\_** выполняется, когда объект выводится (формально при его преобразовании  в отображаемую строку встроенной функцией str или ее внутренним эквивалентом  Python).  *# Вызывается для print (self) , str()* |
|  |  |
|  |  |
|  |  |
| **class Worker:**  **def \_\_init\_\_(self, name, pay):**  **self.name = name**  **self.pay = pay**  **def lastName(self):**  **return self.name.split()[-1]**  **def giveRaise(self, percent):**  **self.pay \*= (1.0 + percent)** | *# Инициализация при создании*  *# Конструктор*  *# self – это сам объект*  *# Разбить строку по символам пробела*  *# Обновить сумму выплат* |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# CLASS

class Ludi():

     def \_\_init\_\_(self,name,age):

        self.name = name

        self.\_age = age      # Скрытие вне класса переменной

        self.god\_rogd = 2020-age

    def god\_rogd1(self):

         self.god\_rogd= 2020 - self.age

         return self.god\_rogd

    @property       Дает возможность прочитать скрытую в классе переменную, переназначив ее

    def age(self):

        return self.\_age

    @age.setter Блок отрабатывает при прямом изменнии переменной

    def age(self, age):

        if age <0: print("не корректное значение")

        elif age <18: print("Замечен нe совершеннолетний")

        elif age >60: print("Дедуля, шли бы вы")

        else: print("Приветствуем тебя пользователь")

    def strL(self):

        x = f":{self.god\_rogd}:Мальчику {self.name}  {self.\_age} лет, он родился в {self.god\_rogd1()} году. Значение переменной {self.god\_rogd}"

        return x

a = Ludi("Мамед",70)

print(a.name)

print(a.age)

a.age=15

print(a.god\_rogd1())

a.strL()

**Перегрузка операторов**

**\_\_init\_\_**(self[, ...]) - как уже было сказано выше, конструктор.

**\_\_del\_\_**(self) - вызывается при удалении объекта сборщиком мусора.

**\_\_repr\_\_**(self) - вызывается встроенной функцией repr; возвращает "сырые" данные, использующиеся для внутреннего представления в python.

**\_\_str\_\_**(self) - вызывается функциями str, print и format. Возвращает строковое представление объекта.

**\_\_bytes\_\_**(self) - вызывается функцией bytes при преобразовании к [байтам](https://pythonworld.ru/tipy-dannyx-v-python/bajty-bytes-i-bytearray.html).

**\_\_format\_\_**(self, format\_spec) - используется функцией format (а также методом format у строк).

**\_\_lt\_\_**(self, other) - x < y вызывает x.\_\_lt\_\_(y).

**\_\_le\_\_**(self, other) - x ≤ y вызывает x.\_\_le\_\_(y).

**\_\_eq\_\_**(self, other) - x == y вызывает x.\_\_eq\_\_(y).

**\_\_ne\_\_**(self, other) - x != y вызывает x.\_\_ne\_\_(y)

**\_\_gt\_\_**(self, other) - x > y вызывает x.\_\_gt\_\_(y).

**\_\_ge\_\_**(self, other) - x ≥ y вызывает x.\_\_ge\_\_(y).

**\_\_hash\_\_**(self) - получение хэш-суммы объекта, например, для добавления в словарь.

**\_\_bool\_\_**(self) - вызывается при проверке истинности. Если этот метод не определён, вызывается метод \_\_len\_\_ (объекты, имеющие ненулевую длину, считаются истинными).

**\_\_getattr\_\_**(self, name) - вызывается, когда атрибут экземпляра класса не найден в обычных местах (например, у экземпляра нет метода с таким названием).

**\_\_setattr\_\_**(self, name, value) - назначение атрибута.

**\_\_delattr\_\_**(self, name) - удаление атрибута (del obj.name).

**\_\_call\_\_**(self[, args...]) - вызов экземпляра класса как [функции](https://pythonworld.ru/tipy-dannyx-v-python/vse-o-funkciyax-i-ix-argumentax.html).

**\_\_len\_\_**(self) - длина объекта.

**\_\_getitem\_\_**(self, key) - доступ по индексу (или ключу).

**\_\_setitem\_\_**(self, key, value) - назначение элемента по индексу.

**\_\_delitem\_\_**(self, key) - удаление элемента по индексу.

**\_\_iter\_\_**(self) - возвращает итератор для контейнера.

**\_\_reversed\_\_**(self) - итератор из элементов, следующих в обратном порядке.

**\_\_contains\_\_**(self, item) - проверка на принадлежность элемента контейнеру (item in self).

**Перегрузка арифметических операторов**

**\_\_add\_\_**(self, other) - сложение. x + y вызывает x.\_\_add\_\_(y).

**\_\_sub\_\_**(self, other) - вычитание (x - y).

**\_\_mul\_\_**(self, other) - умножение (x \* y).

**\_\_truediv\_\_**(self, other) - деление (x / y).

**\_\_floordiv\_\_**(self, other) - целочисленное деление (x // y).

**\_\_mod\_\_**(self, other) - остаток от деления (x % y).

**\_\_divmod\_\_**(self, other) - частное и остаток (divmod(x, y)).

**\_\_pow\_\_**(self, other[, modulo]) - возведение в степень (x \*\* y, pow(x, y[, modulo])).

**\_\_lshift\_\_**(self, other) - битовый сдвиг влево (x << y).

**\_\_rshift\_\_**(self, other) - битовый сдвиг вправо (x >> y).

**\_\_and\_\_**(self, other) - битовое И (x & y).

**\_\_xor\_\_**(self, other) - битовое ИСКЛЮЧАЮЩЕЕ ИЛИ (x ^ y).

**\_\_or\_\_**(self, other) - битовое ИЛИ (x | y).

Пойдём дальше.

**\_\_radd\_\_**(self, other),

**\_\_rsub\_\_**(self, other),

**\_\_rmul\_\_**(self, other),

**\_\_rtruediv\_\_**(self, other),

**\_\_rfloordiv\_\_**(self, other),

**\_\_rmod\_\_**(self, other),

**\_\_rdivmod\_\_**(self, other),

**\_\_rpow\_\_**(self, other),

**\_\_rlshift\_\_**(self, other),

**\_\_rrshift\_\_**(self, other),

**\_\_rand\_\_**(self, other),

**\_\_rxor\_\_**(self, other),

**\_\_ror\_\_**(self, other) - делают то же самое, что и арифметические операторы, перечисленные выше, но для аргументов, находящихся справа, и только в случае, если для левого операнда не определён соответствующий метод.

Например, операция x + y будет сначала пытаться вызвать x.\_\_add\_\_(y), и только в том случае, если это не получилось, будет пытаться вызвать y.\_\_radd\_\_(x). Аналогично для остальных методов.

Идём дальше.

**\_\_iadd\_\_**(self, other) - +=.

**\_\_isub\_\_**(self, other) - -=.

**\_\_imul\_\_**(self, other) - \*=.

**\_\_itruediv\_\_**(self, other) - /=.

**\_\_ifloordiv\_\_**(self, other) - //=.

**\_\_imod\_\_**(self, other) - %=.

**\_\_ipow\_\_**(self, other[, modulo]) - \*\*=.

**\_\_ilshift\_\_**(self, other) - <<=.

**\_\_irshift\_\_**(self, other) - >>=.

**\_\_iand\_\_**(self, other) - &=.

**\_\_ixor\_\_**(self, other) - ^=.

**\_\_ior\_\_**(self, other) - |=.

**\_\_neg\_\_**(self) - унарный -.

**\_\_pos\_\_**(self) - унарный +.

**\_\_abs\_\_**(self) - модуль (abs()).

**\_\_invert\_\_**(self) - инверсия (~).

**\_\_complex\_\_**(self) - приведение к complex.

**\_\_int\_\_**(self) - приведение к int.

**\_\_float\_\_**(self) - приведение к float.

**\_\_round\_\_**(self[, n]) - округление.

**\_\_enter\_\_**(self), **\_\_exit\_\_**(self, exc\_type, exc\_value, traceback) - реализация менеджеров контекста.