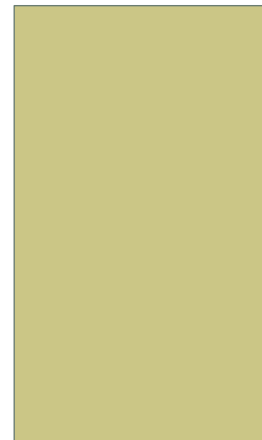


# Контрольная по Python



# Задача №1

Создайте метакласс `ClassBase`. Будучи приписан к классу этот метакласс загружает описание класса (для простоты только поля) из файла с именем совпадающим с именем класса. Если не удалось найти файл, то необходимо сформировать исключение.

*Например:*

*A.txt*

*x: 5*

*y: 7*

```
class A(metaclass = ClassBase):  
    pass
```

```
>>>a = A()  
>>>print(a.x)  
5
```

## Задача №2

**В данном задании запрещается пользоваться библиотечными функциями!**

- Напишите класс двоичного дерева поиска. Дерево должно поддерживать операции сложения (+) с другим деревом или с числом. При попытке сложить с другими типами должно выдаваться исключение.
- Помимо этого необходимо реализовать возможность итерироваться по дереву в порядке увеличения его элементов
- Также необходимо реализовать метод `__contains__` для верной работы оператора *`x in tree`*

```
for i in range(100):  
    tree = tree + 1
```

```
print(10 in tree)
```

```
for i in iter(tree):  
    print(i)
```

## Задача №3

Напишите декоратор `@spy` и функцию `bond(func)`, которая является генератором и выдает кортежи из двух элементов – времени запуска функции `func` и параметров, с которыми она была запущена. Для работы функции `bond` необходимо, чтобы функция была продекорирована, если это не так, то должно быть сгенерировано исключение.

```
@spy  
def foo(num):  
    print(num)
```

```
foo(30)  
foo("hello")  
foo(5)
```

```
for (time, parameters) in bond(foo):  
    print(time)  
    print(parameters)
```

# Задача №4

Напишите декоратор

@apply (func, testFunc), где testFunc - функция которая возвращает True или False.

При этом подразумевается, что декоратор приписан к функции, имеющей такое же количество параметров, что и функция func.

Также необходимо, чтобы функция testFunc тоже имела такое же количество параметров.

Смысл данного декоратора состоит в следующем.

Если вызывается функция func от какого-то набора параметров, то нужно найти первую продекорированную функцию у которой в качестве параметра apply указан func, такую, что соответствующий testFunc для данного набора параметров вернет true.

Если не найдется такая продекорированная функция, то необходимо вызвать исходную.

```
def test1(num):  
    return (num == 1)  
def test2(num):  
    return (num > 3)
```

```
def foo(num):  
    print("Original")
```

```
@apply (foo, test1)  
def foo2(num):  
    print("Modified")
```

```
@apply (foo, test2)  
def foo3(num):  
    print("Magic")
```

```
foo(-1)  
foo(1)  
foo(2)  
foo(4)
```

Выводит  
:  
Original  
Modified  
Original  
Magic