Функциональное программирование

продвинутый уровень

Область видимости

```
a = 5

def randomfunc():
        a += 6
        return a

print(randomfunc());
```

Аргументы по умолчанию

```
def appenderornot(iterable=[])
    for i in range(10):
        iterable.append(i)
    return iterable
```

*args

```
def summator(*args):
    sum = 0
    for i in args:
        sum += i
    return sum
```

**kwargs

```
def printer(**kwargs):
        for key, value in kwargs.items():
                print(“ {} : {}”.format(key,value))
printer(a=10, b=10)
```

Списочные выражения

```
square_list = []

for number in range(10):

    square_list.append(number ** 2)


square_list = [number ** 2 for number in range(10)]
```

Декораторы

```
def bold(func):
    def wrapped():
        return “<b>” + func() + “</b>”
    return wrapped


@bold
def hello_world():
    return “hello world”
print(hello_world())
```

Применение нескольких декораторов

```
def bold(func):
    def wrapped():
        return “<b>” + func() + “</b>”
    return wrapped


def italic(func):
    def wrapped():
        return “<i>” + func() + “</i>”
    return wrapped
```

Генераторы

```
def createGenerator(n) :
    mylist = range(n)
    for i in mylist:
        yield i*i

mygenerator = createGenerator(7) # создаём генератор

for i in mygenerator:
     print(i)
```

Практическое задание

1. Написать декоратор добавляющий наш жирный курсивный шрифт в параграф .
2. Написать функцию суммирующую произвольное количество значений
3. Написать декоратор результат возвращающий квадрат функции сумматора.
4. Написать второй декоратор который вернет куб возвращаемого значения декорируемой функции.
5. Применить декораторы к функции в 3 задании.
6. Создать n “параграфов” где n, это результат работы функции 5.
7. Выполнить задание 6 с помощью генератора.