Тест по лекции 1

1 2 3

Вопрос 3

В каком виде эта функция принимает аргументы?

```
def my_func(*, width=100, height=10):
    ...
```

Ваш ответ:
- ◯ Как позиционные, так и именованные.
- ◯ Только позиционные.
- ◉ Только именованные.

Ответить

1 2 3

Вопрос 2

Можно ли уже определенную функцию записать в произвольную переменную?

Ваш ответ:
- ◉ Да
- ◯ Нет

Ответить

1 2 3

Вопрос 3

С помощью какой какого ключевого слова начинается определение функции?

Ваш ответ:
- ◉ def
- ◯ func
- ◯ class

Ответить

Тест по лекции 2

1 2 3

Вопрос 1

С помощью какого ключевого слова начинается определение анонимной функции?

Ваш ответ:
- ◉ lambda
- ◯ anonymous
- ◯ adef

Ответить

1 2 3

Вопрос 2

Может ли функция вызывать сама себя?

Ваш ответ:
- ◉ Да
- ◯ Нет

Ответить

1 2 3

Вопрос 3

Можно ли анонимную функцию передавать как аргумент другой функции? А обычную функцию?

Ваш ответ:
- ◯ Да, Нет.
- ◯ Нет, Да.
- ◉ Да, Да.
- ◯ Нет, Нет.

Ответить

Тест по лекции 3

1 2 3

Вопрос 1

С помощью какой встроенной функции можно получить следующее значение из итератора/генератора?

Ваш ответ:

- ○ get()
- ◉ next()
- ○ iter()

Ответить

1 2 3

Вопрос 2

Как вычисляются значения в генераторе?

Ваш ответ:

- ◉ Каждое следующее значение вычисляется по запросу, то есть например при вызове next()
- ○ Все значения вычисляются сразу, а потом мы по ним итерируемся как по списку.

Ответить

1 2 3

Вопрос 3

Какое ключевое слово используется в генераторной функции для возвращения очередного значения?

Ваш ответ:

- ○ return
- ○ generate
- ◉ yield

Ответить

Тест по лекции 4

1 2 3

Вопрос 1

Можно ли передавать дополнительные параметры в декоратор, параметризовать его?

Ваш ответ:

- ◉ Да
- ○ Нет

Ответить

1 2 3

Вопрос 2

Какому коду эквивалентен следующий код?

```
@my_decorator2
@my_decorator1
def my_func():
    print("Hello world")
```

Ваш ответ:

◉
```
def my_func():
    print("Hello world")
my_func = my_decorator2(my_decorator1(my_func))
```

○
```
def my_func():
    print("Hello world")
my_func = my_decorator1(my_decorator2(my_func))
```

Ответить

1 2 3

Вопрос 3

Для чего может быть использован декоратор?

Ваш ответ:

- Для автоформатирования кода.
- Для добавления дополнительной логики к другой функции.
- Для переименования функции.

Ответить

Тест по лекции 5

1 2 3 4

Вопрос 1

В какой области видимости можно менять переменную с помощью ключевого слова global?

Ваш ответ:

- Глобальная
- Встроенная
- Объемлющая

Ответить

1 2 3 4

Вопрос 2

Какая область является самой узкой среди 4 областей видимости?

Ваш ответ:

- Локальная
- Глобальная
- Встроенная
- Объемлющая

Ответить

1 2 3 4

Вопрос 3

В какой области видимости можно менять переменную с помощью ключевого слова nonlocal?

Ваш ответ:

- Глобальная
- Встроенная
- Объемлющая

Ответить

1 2 3 4

Вопрос 4

Какая область является самой общей среди 4 областей видимости?

Ваш ответ:

- Локальная
- Глобальная
- Встроенная
- Объемлющая

Ответить

Тест по лекции 6

1 2 3 4

Вопрос 1

Чем отличается функции set() от frozenset()?

Ваш ответ:

- (•) В set можно добавлять и убирать элементы, из frozenset - нельзя
- ( ) В frozenset можно добавлять и убирать элементы, из set - нельзя

Ответить

1 2 3 4

Вопрос 2

Какая функция позволяет отсортировать массив?

Ваш ответ:

- (•) sorted
- ( ) reversed

Ответить

1 2 3 4

Вопрос 3

Какая функция позволяет обходить элементы одновременно во всех итерируемых объектах?

Ваш ответ:

- (•) enumerate
- ( ) zip
- ( ) reversed
- ( ) sorted

Ответить

1 2 3 4

Вопрос 4

Какие параметры принимает функция list?

Ваш ответ:

- (•) Любой итерируемый объект
- ( ) Принимает элементы списка явно

Ответить

Домашнее задание 1

Необходимо написать программу, которая будет считывать со входа данные последовательностей чисел, считать и выводить их среднее значение. Напишите сначала функцию, которая будет принимать строку, а в ответ возвращать среднее значение чисел из нее. А далее применяйте эту функцию к каждой считанной входной последовательности. На вход будут подаваться строки, в которых расположены целые числа, разделенные пробелом. Передача пустой строки будет означать конец входных данных.

```
a = list(map(int, input().split()))


while not a == []:
    print(round(sum(a) / len(a), 2))
    a = list(map(int, input().split()))
```

Домашнее задание 2

Необходимо написать программу, которая будет принимать на вход строки, преобразовывать и выводить их. Если первым символом в строке является “!”, то строку нужно привести к верхнему регистру, иначе к нижнему. Также из строки нужно удалить все символы “!”, “@”, “#”, “%”. Получившуюся строку нужно вывести. Реализуйте функцию, которая будет принимать строку, обрабатывать ее по условиям выше и возвращать обработанную строку в качестве результата. Полученную функцию применяйте к строкам подаваемым на вход.

```
signs = ["!", "@", "#", "%"]

a = input()

while not a == "":
    a = a.lower()
    if a[0] == "!":
        a = a.upper()

    for i in signs:
        a = a.replace(i, "")

    print(a)
    a = input()
```

Домашнее задание 3

напишите код, который также будет использовать функцию map() и лямбда функцию. На вход будут подаваться три аргумента для range: начало, конец и шаг числовой последовательности. Нужно вывести для каждого элемента range квадрат числа, если число нечетное, иначе вывести противоположное ему. (Решение можно реализовать в две и даже в одну строку!)

```
a, b, c = map(int, input().split())
x = range(a, b, c)
for i in map(lambda x: int(x)**2 if int(x) % 2 == 1 else -int(x), x): print(i)
```

Домашнее задание 4

Необходимо написать программу, которая будет считывать со стандартного ввода положительное целое число – порядковый номер 1 <= n <= 30, и выводить n-е по счету число Фибоначчи. Числа Фибоначчи это последовательность чисел такая, что каждое следующее число это сумма двух предыдущих. Первое и второе числа Фибоначчи это числа 1. То есть первые два числа это 1 и 1, третье число это 2 (сумма первого и второго), четвертое число это 3 (сумма второго и третьего), пятое – 5, шестое – 8 и так далее. Нужно написать этот код с помощью рекурсии.

```
a = [0, 1]


n = int(input())
for i in range(2, n + 1):
    a.append(a[i - 1] + a[i - 2])


print(a[n])
```

Домашнее задание 5

В этом задании нужно будет написать свою реализацию map(), который мы обсуждали в прошлом ДЗ. То есть нужно написать генераторную функцию, которая первым аргументом будет принимать функцию, а вторым некую последовательность. Полученный генератор должен генерировать значения из переданной последовательности, пропущенные через переданную первым аргументом функцию.

```
def map(func, seq):
    result = []
    for i in seq:
        result.append(func(i))
    return result

func_in, seq_in = eval(input()), eval(input())

for x in map(func_in, seq_in):
   print(x)
```

Домашнее задание 6

В этом задании нужно будет написать свою реализацию filter(), то есть генераторную функцию, которая принимает функцию и последовательность и фильтрует последовательность в зависимости от вердикта переданной функции. Воспользуйтесь своей реализацией, чтобы применить лямбда функцию, поданную на вход, к поданной на вход последовательности.

```
def filter(func, seq):
   result = []
   for i in seq:
      if func(i) == True:
         result.append(i)
   return result

func_in, seq_in = eval(input()), eval(input())

for x in filter(func_in, seq_in):
   print(x)
```

Домашнее задание 7

Напишите декоратор, который будет кэшировать вызовы функции, которая будет передана на вход. То есть декоратор должен проверить нет ли в кэше (например, словаре) значения, и если нет, то вычислить и запомнить результат, если есть, то взять это значение.

```
def cache_deco(func):
    cache = {}

    def cached_func(*args):
        if args not in cache:
            cache[args] = func(*args)
        return cache[args]

    return cached_func



code = []

while data := input():
    code.append(data)

code = "\n".join(code)
exec(code)
```

Домашнее задание 8

Напишите декоратор, который будет принимать натуральное число n – число повторений – и будет повторять вызов декорированной функции n раз, а также возвращать значение из последнего вызова.

```
def repeat_deco(n=1):
    def decorator(func):
        def wrapper(*args, **kwargs):
            result = None
            for _ in range(n):
                result = func(*args, **kwargs)
            return result

        return wrapper

    return decorator



code = []

while data := input():
   code.append(data)

code = "\n".join(code)
exec(code)
```

Домашнее задание 9

Напишите функцию f, в которой глобальная переменная a типа int, полученная со стандартного ввода, увеличивается на 10.

```
a = int(input())


def f():
   global a
   a += 10


f()


print(a)
```

Домашнее задание 10

Напишите функцию g вложенную в f, в которой переменная b типа int, полученная со стандартного ввода в объемлющей области видимости увеличивается на 10.

```
def g():
   global b
   b = int(input())
   def h():
      global b
      b += 10
   h()
   print(b)
g()
```

Домашнее задание 11

Представьте, что у вас есть словарь с ключами и их частотами (то есть насколько часто встречался каждый ключ) в качестве значений. Напишите функцию make_most_common_keys, которая принимает словарь частот и выводит отсортированный (например через функцию sorted) по убыванию частот (то есть значений ключей) список ключей.

```
from typing import List, Dict


def make_most_common_keys(d: Dict[int, int]) -> List[int]:
   sorted_freq = sorted(d.items(), key=lambda x: x[1], reverse=True)
   most_common_keys = [key for key, _ in sorted_freq]
   return most_common_keys


code = []
while data := input():
    code.append(data)


code = "\n".join(code)
exec(code)
```

Домашнее задание 12

Написать функцию get_indexes которая принимает два списка и возвращает список индексов, в которых элемент из первого списка меньше элемента из второго списка по данному индексу. Желательно проходиться сразу по двум массивам одновременно (вспомните функцию zip). Для нахождения индексов можно использовать enumerate вместе с zip.

```
from typing import List


def get_indexes(nums1: List[int], nums2: List[int]) -> List[int]:
   indexes = [i for i, (x, y) in enumerate(zip(nums1, nums2)) if x < y]
   return indexes


code = []
while data := input():
   code.append(data)
code = "\n".join(code)
exec(code)
```

Домашнее задание 13

Необходимо написать генераторную функцию solution, которая будет фильтровать данные из последовательности data функцией func_filter, к полученным данным применять функцию func_map и возвращать в качестве значения каждый второй элемент полученной последовательности. Нужно пользоваться здесь концепцией генератора, то есть обрабатывать не все данные разом, а только те, что необходимы для возвращения следующего значения.

```
def cache_deco(func):
    cache = {}

    def wrapper(*args):
        if args not in cache:
            cache[args] = func(*args)
        return cache[args]

    return wrapper

n = 0

def solution(func_map, func_filter, data):
    global n
    for key, value in enumerate(data):
        if func_filter(value):
            if n % 2 == 0:
                yield func_map(value)
            else:
                func_map(value)
            n += 1

code = []
while data := input():
    code.append(data)
code = "\n".join(code)
exec(code)
```