ЛАБОРАТОРНАЯ РАБОТА 5

Содержание

Определение функций

```
def имя_функции():
    тело_функции
```

```
def имя_функции(список_параметров):
    тело_функции
```

```
def имя_функции(список_параметров):
    тело_функции
    return возвращаемое_значение
```

Вызов функции

```
имя_функции(список_аргументов)
```

Задание № 0.1. Определить результат вызова следующих функций:

(a)
```
def print_line():
    print("-" * 20)

print_line()
```

(b)
```
def print_line(element, repeats):
    print(element * repeats)

print_line("-", 20)
print_line("*", 40)
print_line("-_", 50)
```

(c)
```
def mult(x, y):
    return x * y

print(mult(2, 2))

day_seconds = mult(24, 3600)
print(day_seconds)

res = mult("-", 40)
print(res)
```

Значения аргументов по умолчанию

Задание № 0.2. Определить результат следующих вызовов функций:

(a)
```
def print_line(element="-", repeats=40):
    print(element * repeats)

print_line(".", 20)
print_line("x")
print_line()
```

(b)
```
def print_in_frame(word, symbol="#", width=40):
    print_line(symbol, width)
    print(f"{symbol}{word.center(width-2)}{symbol}")
    print_line(symbol, width)

print_in_frame("Python")
```

Позиционные (positional) и именованные (keyword) аргументы

```
def f(pos1, pos2, /, pos_or_kwd, *, kwd1, kwd2):
    pass
```

Задание № 0.3. Определить результат следующих вызовов функций:

(a)
```
def say_hello(name, end_sign="!", repeats=1, greeting="Hello"):
    print(f"{greeting}, {name}{end_sign} " * repeats)

say_hello("Helen")
say_hello("Mary", ".", 2)
say_hello(repeats=3, end_sign="?", name="Bob")
say_hello("Arnold", greeting="Hey")
```

(b)
```
def say_hello(name, /, end_sign="!", repeats=1, *, greeting="Hello"):
    print(f"{greeting}, {name}{end_sign} " * repeats)

say_hello("Helen")
say_hello("Mary", ".", 2)
say_hello("Bob", repeats=3, end_sign="?")
say_hello("Arnold", greeting="Hey")
```

Именованные аргументы стандартной функции `print()`

Задание № 0.4. Определить результат вывода функции `print()`:

(a)
```
planets = ["Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn"]
for planet in planets:
    print(planet)
```

(b)
```
planets = ["Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn"]
for planet in planets:
    print(planet, end="; ")
```

(c)
```
x, y, z = 1, 2, 3
print(x, y, z, sep=" + ")
```

(d)
```
x, y, z = 1, 2, 3
print(x, y, z, sep=" + ", end=" = ?\n")
```

Стандартные функции `dir()`, `help()`, `range()`

Задание № 0.5. Определить результат вызова следующих функций:

(a)
```
print(dir())
```

(b)
```
import math
print(dir("math"))
```

(c)
```
help("dir")
```

(d)
```
help("math.sin")
```

(e)
```
for i in range(10):
    print(i)
```

(f)
```
my_list = list(range(10))
print(my_list)
```

(g)
```
for i in range(4, 10):
    print(i, " ", i*i)
```

(h)
```
even = list(range(2, 100, 2))
print(even)
```

(i)
```
countdown = range(10, 0, -1)
for i in countdown:
    print(i, end=" ")
print()
```

Строки документации функций

Задание № 0.6. Определить результат выполнения следующего кода:

(a)
```
def print_line(element="-", repeats=40):
    """Печатает линию из символов.

    Линия получается выводом символа element
    repeats раз.
    """

    print(element * repeats)

print(print_line.__doc__)
```

(b)
```
print(dir.__doc__)
```

Строки документации файлов

```
'''Это документация для всего кода в данном файле.

Первая строка документации считается кратким описанием.
Она отделяется от остальной документации пустой строкой.

Далее можно разместить столько текста, сколько необходимо.
'''

def print_line(element='#', repeats=65):
    '''Печатает линию из символов.

    Линия получается выводом символа element
    repeats раз.
    '''

    print(element * repeats)

def print_borders(multistring, element='#', width = 65):
    '''Выводит линию из символов слева и справа от строки.

    multistring — строка, возможно с символами переноса.
    width — итоговая ширина текста вместе с границами.
    '''

    string_list = multistring.split('\n')
    for string in string_list:
        print(element + ' ' + string.ljust(width-4) + ' ' + element)

print_line()
print_borders(__doc__)
print_line()
```

Некоторые алгоритмы

Задание № 0.7. С помощью цикла `for` реализовать алгоритм подсчета элементов в заданном списке:

1. Присвоить переменной `count` значение `0`.
2. В цикле `for` для каждого значения элемента списка увеличить значение переменной `count` на единицу.
3. Вывести значение переменной `count`.

Задание № 0.8. Предложить альтернативный способ подсчета элементов в заданном списке.

Задание № 0.9. С помощью цикла `for` реализовать алгоритм подсчета положительных элементов в заданном числовом списке:

1. Присвоить переменной `count` значение `0`.
2. В цикле `for` для каждого значения элемента списка: если данный элемент больше нуля, увеличить значение переменной `count` на единицу.
3. Вывести значение переменной `count`.

Задание № 0.10. С помощью цикла `for` реализовать алгоритм подсчета суммы элементов в заданном числовом списке:

1. Присвоить переменной `sum` значение `0`.
2. В цикле `for` для каждого значения элемента списка увеличить значение переменной `sum` на величину данного элемента.
3. Вывести значение переменной `sum`.

Задание № 0.11. С помощью цикла `for` реализовать алгоритм определения минимального элемента в заданном числовом списке:

1. Присвоить переменной `min_elem` значение первого элемента списка.
2. В цикле `for` для каждого значения элемента списка: если данный элемент меньше значения `min_elem`, то присвоить переменной `min_elem` значение этого элемента.
3. Вывести значение переменной `min_elem`.

Требования к программам

1. Файл с исходным кодом называть по шаблону: «Фамилия_номер_задания» английским алфавитом (пример: `Ivanov_5_42.py`).
2. Файл с исходным кодом должен начинаться с многострочного описательного комментария (см. базовый шаблон ниже).
3. Имена переменных выбирать разумными.
4. Оформлять понятный ввод данных / вывод результата.
5. Имена функций и имена параметров выбирать разумными.
6. Программа должна содержать вызовы всех определенных в ней функций.

Базовый шаблон для всех лабораторных

```
'''Фамилия Имя. Номер задания

Краткая формулировка задания
'''

# Код программы
```

Пример программы

```
'''Фамилия Имя. Задание № 5.0

Определить функцию с 4 параметрами (функция, начальная точка, конечная точка,
количество разбиений отрезка), табулирующую заданную функцию.
'''
import math

def print_function(f, a, b, n):
    h = (b-a) / n

    print(f"{'x':>4s}\t{f.__name__:>10}")
    for i in range(n+1):
        x = a + i*h
        y = f(x)
        print(f"{x:4.2f}\t{y:10.4f}")

def f(d):
    return math.log((d+1)/d, 10)

print_function(math.sin, 0, 2*math.pi, 10)

print()

print_function(f, 1, 9, 8)
```

Индивидуальные задания

Задание № 5.1. Определить функцию, возвращающую количество положительных элементов в заданном числовом списке.

Задание № 5.2. Определить функцию, возвращающую сумму положительных элементов в заданном числовом списке.

Задание № 5.3. Определить функцию, возвращающую значение минимального элемента в заданном числовом списке.

Задание № 5.4. Определить функцию, возвращающую значение максимального элемента в заданном числовом списке.

Задание № 5.5. Определить функцию, возвращающую количество элементов в заданном списке строк, которые содержат заданную подстроку.

Задание № 5.6. Определить функцию, возвращающую среднее арифметическое заданного числового списка.

Задание № 5.7. Определить функцию с тремя параметрами — день, месяц, год, возвращающую строку с датой в формате `"dd.mm.yyyy"`.

Задание № 5.8. Определить функцию с тремя параметрами — часы, минуты, секунды, возвращающую строку со временем в формате `"hh:mm:ss"`.

Задание № 5.9. Определить функцию, возвращающую долю слов `"yes"` в заданном списке из слов `"yes"` и `"no"`.

Пример:

Аргумент функции	`["yes", "no", "no", "no", "yes"]`
Возвращаемое значение	`0.4`

Задание № 5.10. Определить функцию с одним параметром — строкой, возвращающую строку со словами из данной строки, но в отсортированном порядке.

Пример:

Аргумент функции	`"program file text code number line"`
Возвращаемое значение	`"code file line number program text"`

Задание № 5.11. Определить функцию с двумя параметрами — строкой со словами, разделенными пробелами, и целым положительным числом, возвращающую список со всеми словами из строки, длина которых не меньше данного числа.

Пример:

Аргументы функции	`"многообразие смысл феномен массив данные"` `7`
Возвращаемое значение	`["многообразие", "феномен"]`

Задание № 5.12. Определить функцию, возвращающую список всех слов данной строки без повторений.

Пример:

Аргумент функции	`"these boys play with those boys"`
Возвращаемое значение	`["these", "boys", "play", "with", "those"]`

Задание № 5.13. Определить функцию с двумя параметрами — списками, возвращающую список, который содержит все элементы второго списка, не входящие в первый.

Пример:

Аргументы функции	`["apple", "orange", "banana"]` `["bread", "orange", "butter", "egg"]`
Возвращаемое значение	`["bread", "butter", "egg"]`

Задание № 5.14. Определить функцию с двумя параметрами — списками, возвращающую список, который содержит все элементы, входящие только в один из списков, но не в оба одновременно.

Пример:

Аргументы функции	`["one", "two", "three", "four", "five"]` `["zero", "two", "four", "six"]`
Возвращаемое значение	`["one", "three", "five", "zero", "six"]`

Задание № 5.15. Определить функцию с двумя параметрами — список строк и строка, возвращающий новый список строк, сформированный добавлением данной строки ко всем строкам исходного списка.

Пример:

Аргументы функции	`["lab_5", "book", "lecture"]` `".pdf"`
Возвращаемое значение	`["lab_5.pdf", "book.pdf", "lecture.pdf"]`

Полезные ссылки

Официальный сайт по языку Python:

https://www.python.org/

Официальная документация по языку Python 3:

https://docs.python.org/3/

Онлайн-интерпретаторы языка Python:

https://www.onlinegdb.com/online_python_compiler

https://www.online-python.com/

https://www.programiz.com/python-programming/online-compiler/