

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ  
АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО  
ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»  
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

**Отчет по лабораторной работе №11  
«Работа с функциями в языке Python»**

**по дисциплине «Основы программной инженерии»**

Выполнила:  
Первых Дарья Александровна,  
2 курс, группа ПИЖ-б-о-20-1,  
Проверил:  
Доцент кафедры инфокоммуникаций,  
Воронкин Р.А.

Ставрополь, 2021 г.

## ВЫПОЛНЕНИЕ

### 1. Практическая часть

```
i = 0
while i < 3:
    a = int(input())
    b = int(input())
    i += 1
```

Рисунок 1 – Пример использования цикла

```
1 print("Сколько бананов и ананасов для обястьян?")
2 a = int(input())
3 b = int(input())
4 print("всего", a+b, "шт.")
5
6 print("Сколько жуков и червей для ежей?")
7 a = int(input())
8 b = int(input())
9 print("всего", a+b, "шт.")
10
```

main ×

C:\Users\podar\study\anaconda\envs\11LR\python.exe "C  
Сколько бананов и ананасов для обястьян?  
12  
5  
всего 17 шт.  
Сколько жуков и червей для ежей?  
5  
7  
всего 12 шт.  
Сколько рыб и моллюсков для выдр?  
27  
12  
всего 39 шт.

Рисунок 2 – Пример длинного кода без функции

```
def countFood():
    a = int(input())
    b = int(input())
    print("Всего", a+b, "шт.")
```

Рисунок 3 – Пример использования оператора def

```

1 def countFood():
2     a = int(input())
3     b = int(input())
4     print("Всего", a + b, "шт.")
5
6
7     print("Сколько бананов и ананасов для обястьян?")
8     countFood()
9
10    print("Сколько жуков и червей для ежей?")
11    countFood()

```

main x

```

C:\Users\podar\study\anaconda\envs\11LR\python.exe "C:
Сколько бананов и ананасов для обястьян?
12
15
Всего 27 шт.
Сколько жуков и червей для ежей?
27
12
Всего 39 шт.
Сколько рыб и моллюсков для выдр?
45
37
Всего 82 шт.

```

Рисунок 4 – Пример вызова функции

```
4 figure = input("1-прямоугольник, 2-треугольник, 3-круг: ")
5
6 if figure == '1':
7     a = float(input("Ширина: "))
8     b = float(input("Высота: "))
9     print(f"Площадь: {a * b}")
10 elif figure == '2':
11     a = float(input("Основание: "))
12     h = float(input("Высота: "))
13     print(f"Площадь: {0.5 * a * h}")
14 elif figure == '3':
15     r = float(input("Радиус: "))
16     print(f"Площадь: {math.pi * r**2}")
17 else:
18     print("Ошибка ввода", file=sys.stderr)
```

main x

C:\Users\podar\study\anaconda\envs\11LR\python.exe "C:/Users/p  
1-прямоугольник, 2-треугольник, 3-круг: 2  
Основание: 5  
Высота: 3  
Площадь: 7.5

Рисунок 5 – Пример не структурированной программы

```
1 import math
2 import sys
3
4 figure = input("1-прямоугольник, 2-треугольник, 3-круг: ")
5
6 def rectangle():
7     a = float(input("Ширина: "))
8     b = float(input("Высота: "))
9     print(f"Площадь: {a * b}")
10
11
12 def triangle():
13     a = float(input("Основание: "))
14     h = float(input("Высота: "))
15     print(f"Площадь: {0.5 * a * h}")
16
17
18 main ×
19 C:\Users\podar\study\anaconda\envs\11LR\python.exe "C:/Users/podar/s
20 1-прямоугольник, 2-треугольник, 3-круг: 3
21 Радиус: 2
22 Площадь: 12.566370614359172
```

Рисунок 6 – Пример структурированной программы с функциями

```

1   result = 0
2
3   def rectangle():
4       a = float(input("Ширина: "))
5       b = float(input("Высота: "))
6       result = a * b
7
8   def triangle():
9       a = float(input("Основание: "))
10      h = float(input("Высота: "))
11      result = 0.5 * a * h
12
13     figure = input("1-прямоугольник, 2-треугольник: ")
14     if figure == '1':
15         rectangle()
16     elif figure == '2':
17         triangle()

```

Рисунок 7 – Пример неправильного использования глобальных и локальных переменных

```

R C:\Users\podar\study\PyCharm Community Edition 2021.2.3\11LR
5       b = float(input("Высота: "))
6       global result
7       result = a * b
8
9   def triangle():
10      a = float(input("Основание: "))
11      h = float(input("Высота: "))
12      global result
13      result = 0.5 * a * h
14
15     figure = input("1-прямоугольник, 2-треугольник: ")
16     if figure == '1':
17         rectangle()
18     elif figure == '2':
19         triangle()
20
main x
C:\Users\podar\study\anaconda\envs\11LR\python.exe "C:/Use
1-прямоугольник, 2-треугольник: 1
Ширина: 5
Высота: 6
Площадь: 30.00

```

Рисунок 8 – Пример преобразования глобальной переменной в локальную

```

import math

def cylinder():
    r = float(input())
    h = float(input())
    side = 2 * math.pi * r * h
    circle = math.pi * r ** 2
    full = side + 2 * circle
    return full

square = cylinder()
print(square)

```

Рисунок 9 – Пример возврата значений из функции

```

1  import math
2
3  def cylinder():
4      try:
5          r = float(input())
6          h = float(input())
7      except ValueError:
8          return
9
10     side = 2 * math.pi * r * h
11     circle = math.pi * r ** 2
12     full = side + 2 * circle
13     return full
14
15     print(cylinder())

```

main x

C:\Users\podar\study\anaconda\envs\11LF

4

5

226.1946710584651

Рисунок 10 – Пример работы эксепт обработчика исключений

```
1 import math
2
3 def cylinder():
4     r = float(input())
5     h = float(input())
6     side = 2 * math.pi * r * h
7     circle = math.pi * r ** 2
8     full = side + 2 * circle
9     return side, full
10
11 scyl, fcyl = cylinder()
12 print(f"Площадь боковой поверхности {scyl}")
13 print(f"Полная площадь {fcyl}")
```

main ×

C:\Users\podar\study\anaconda\envs\11LR\python.exe "

7

5

Площадь боковой поверхности 219.9114857512855

Полная площадь 527.7875658030853

Рисунок 11 – Пример возврата нескольких значений

```
1 import math
2
3 def cylinder(h, r=1):
4     side = 2 * math.pi * r * h
5     circle = math.pi * r ** 2
6     full = side + 2 * circle
7     return full
8
9 figure1 = cylinder(4, 3)
10 figure2 = cylinder(5)
11 print(figure1)
12 print(figure2)
```

main ×

C:\Users\podar\study\anaconda\envs\11LR\python.exe "

131.94689145077132

37.69911184307752

Рисунок 12 – Пример работы с произвольным количеством аргументов



```
1 def one_or_many(*a):
2     print(a)
3
4     one_or_many(1)
5     one_or_many('1', 1, 2, 'abc')
6     one_or_many()
```

main x

C:\Users\podar\study\anaconda\en  
(1,)  
( '1', 1, 2, 'abc' )  
( )

Рисунок 13 – Пример передачи аргумента в функцию

```
1 def func(x, y):
2     return x ** 2 + y ** 2
3
4 func = lambda x, y: x ** 2 + y ** 2
```

Рисунок 14 – Пример работы конструкции lambda

```
foo = [2, 18, 9, 22, 17, 24, 8, 12, 27]

print(list(filter(lambda x: x % 3 == 0, foo)))
print((list(map(lambda x: x * 2 + 10, foo))))
```

main x

C:\Users\podar\study\anaconda\envs\11LR\python.exe  
[18, 9, 24, 12, 27]  
[14, 46, 28, 54, 44, 58, 26, 34, 64]

Рисунок 15 – Пример хорошего применения lambda со встроенными функциями – map, filter

```
1 def kos_root():
2     """Return the pathname of the KOS root directory."""
3     global _kos_root
4     if _kos_root: return _kos_root
```

Рисунок 16 – Пример одиночной строки документации

```

4 import sys
5 from datetime import date
6
7 def get_worker():
8     """
9     Запросить данные о работнике.
10    """
11    name = input("Фамилия и инициалы? ")
12    post = input("Должность? ")

```

main x

Должность: студент  
Год поступления? 2020  
>>> add  
Фамилия и инициалы? Савин Д. А.  
Должность? старший вожатый  
Год поступления? 2019  
>>> list

№	Ф.И.О.	Должность	Год
1	Первых Д. А.	студент	2020
2	Савин Д. А.	старший вожатый	2019

Рисунок 17 – Пример № 1

```

4 import sys
5
6 def test():
7     a = int(input("Введите число: "))
8     if 0 < a:
9         positive(a)
10    elif a == 0:
11        print("Это не положительное и не отрицательное число")
12    else:
13        negative(a)
14
15 def positive(a):
16     print("Это положительное число")
17
18 def negative(a):

```

main x

C:\Users\podar\study\anaconda\envs\11LR\python.exe "C:/Users/podar/study/  
Введите число: -2  
Это отрицательное число

```
4 import sys
5
6 def test():
7     a = int(input("Введите число: "))
8     if 0 < a:
9         positive(a)
10    elif a == 0:
11        print("Это не положительное и не отрицательное число")
12    else:
13        negative(a)
14
15 def positive(a):
16     print("Это положительное число")
17
18 def negative(a):
```

main x

C:\Users\podar\study\anaconda\envs\11LR\python.exe "C:/Users/podar/study.  
Введите число: 7  
Это положительное число

Рисунок 18 – Задание № 1

```
def circle():
    plk = math.pi * r ** 2
    return plk

numbers = input("1 - площадь боковой поверхности, 2 - полная площадь: ")
if numbers == '1':
    print(2 * math.pi * r * h)
elif numbers == '2':
    print(circle()*2)
else:
    print("Ошибка")

if __name__ == '__main__':
    cylinder()
```

main x

C:\Users\podar\study\anaconda\envs\11LR\python.exe "C:/Users/podar/study/PyCharm

Введите радиус: 7

Введите высоту: 2

1 - площадь боковой поверхности, 2 - полная площадь: 1

87.96459430051421

```
1  #!/usr/bin/env python3
2  #- coding: utf-8 -*-
3
4  import math
5  import sys
6
7  def cylinder():
8      r = float(input("Введите радиус: "))
9      h = float(input("Введите высоту: "))
10
11     def circle():
12         plk = math.pi * r ** 2
13         return plk
14
15     numbers = input("1 - площадь боковой поверхности, 2 - полная площадь: ")
16     if numbers == '1':
17         print(2 * math.pi * r * h)
18     elif numbers == '2':
19         print(circle()*2)
20     else:
21         print("Ошибка")
22
23 if __name__ == '__main__':
24     cylinder()
```

main x

C:\Users\podar\study\anaconda\envs\11LR\python.exe "C:/Us

Введите радиус: 2

Введите высоту: 7

1 - площадь боковой поверхности, 2 - полная площадь: 2

25.132741228718345

Рисунок 19 – Задание №2



```
17 def display_products(goods):
18     if goods:
19         line = '+-{}-+-{}-+-{}-+-{}-+'.format(
20             '-' * 4,
21             '-' * 30,
            get_product()
main x
Стоимость: 65
>>> add
Название продукта: мандарины
Название магазина: перекрёсток
Стоимость: 62.4
>>> add
Название продукта: пирожное
Название магазина: магнит
Стоимость: 120
>>> add
Название продукта: апельсин
Название магазина: пятёрочка
Стоимость: 100
>>> add
Название продукта: голубика
Название магазина: перекрёсток
Стоимость: 500.6
```

```
7     name = input("Название продукта: ")
8     shop = input("Название магазина: ")
9     price = float(input("Стоимость: "))
10
11     return {
12         'name': name,
13         'shop': shop,
14         'price': price,
15     }
16
17 def display_products(goods):
18     if goods:
19         line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            get_product()
main x
>>> list
+-----+-----+-----+-----+
| № | |          Продукт          | |      Магазин      | |   Цена   | |
+-----+-----+-----+-----+
| 1 | |      апельсин      | |    пятёрочка    | |  100.0   | |
| 2 | |      голубика      | |    перекрёсток  | |  500.6   | |
| 3 | |      мандарины     | |    перекрёсток  | |   62.4   | |
| 4 | |      молоко       | |      магнит     | |   65.0   | |
| 5 | |    пирожное       | |      магнит     | |  120.0   | |
| 6 | |       сыр         | |    пятёрочка    | |  120.0   | |
+-----+-----+-----+-----+
```

```

    return {
        'name': name,
        'shop': shop,
        'price': price,
    }

def display_products(goods):
    if goods:
        line = '+--{}--{}--{}--{}--{}--+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)

```

idz x

```

>>> select
Введите товар: сыр

```

№	Продукт	Магазин	Цена
1	сыр	три	23.5

```

name = input("Название продукта: ")
shop = input("Название магазина: ")
price = float(input("Стоимость: "))

return {
    'name': name,
    'shop': shop,
    'price': price,
}

def display_products(goods):
    if goods:
        line = '+--{}--{}--{}--{}--{}--+'.format(

```

display\_products()

main x

№	продукт	магазин	цена
1	апельсин	пятёрочка	100.0
2	голубика	перекрёсток	500.6
3	мандарины	перекрёсток	62.4
4	молоко	магнит	65.0
5	пироженное	магнит	120.0
6	сыр	пятёрочка	120.0

```

>>> select
Введите товар: голубика

```

Рисунок 21 – Решение идз

## Вопросы

### 1. Каково назначение функций в языке программирования Python?

Функция – это средство (способ) группирования фрагментов программного кода таким образом, что этот программный код может вызываться многократно с помощью использования имени функции.

Использование функций в программах на Python даёт следующие взаимосвязанные преимущества:

- избежание повторения одинаковых фрагментов кода в разных частях программы;
- уменьшение избыточности исходного кода программы. Как следствие, уменьшение логических ошибок программирования;
- улучшенное восприятие исходного кода программы в случаях, где вместо блоков многочисленных инструкций (операторов) вызываются имена готовых протестированных функций. Это, в свою очередь, также уменьшает количество ошибок;
- упрощение внесения изменений в повторяемых блоках кода, организованных в виде функций. Достаточно внести изменения только в тело функции, тогда во всех вызовах данной функции эти изменения будут учтены;
- с помощью функций удобно разбивать сложную систему на более простые части. Значит, функции – удобный способ структурирования программы;
- уменьшение трудозатрат на программирование, а, значит, повышение производительности работы программиста.

### 2. Каково назначение операторов def и return?

Оператор def, выполняемый внутри определения функции, определяет локальную функцию, которая может быть возвращена или передана. Свободные переменные, используемые во вложенной функции, могут обращаться к локальным переменным функции, содержащей def.

Оператор return [выражение] возвращает результат из функции. Оператор return без аргументов аналогичен return None

### 3. Каково назначение локальных и глобальных переменных при написании функций в Python?

Все variables Python, которые доступны в какой - то момент в коде либо в локальной области видимости или в глобальном масштабе.

Объяснение состоит в том, что локальная область действия включает в себя все переменные, определённые в текущей функции, а глобальная область действия включает переменную, определённую вне текущей функции.

### 4. Как вернуть несколько значений из функции Python?

С помощью оператора return из функции можно вернуть одно или несколько значений. Возвращаемым объектом может быть: число, строка, None. Чтобы вернуть несколько значений, нужно написать их через запятую.



## 5. Какие существуют способы передачи значений в функцию?

Существует два способа передачи параметров в функцию: по значению и по адресу. При передаче по значению на месте формальных параметров записываются имена фактических параметров. При вычислении функции в стек заносятся копии значений фактических параметров, и операторы функции работают с этими копиями.

## 6. Как задать значение аргументов функции по умолчанию?

В Python аргументам функции можно присваивать значения по умолчанию. Мы можем предоставить аргументу значение по умолчанию, используя оператор присваивания =. Вот пример: `def greet(name, msg="Доброе утро!"):` `"""` Эта функция выводит для человека с именем `name` сообщение `msg`.

## 7. Каково назначение lambda-выражений в языке Python

Лямбда-выражения на Python - конструкторы простых безымянных однострочных функций. Могут быть использованы везде, где требуется.

## 8. Как осуществляется документирование кода согласно PEP257?

Документирование кода в python - достаточно важный аспект, ведь от неё порой зависит читаемость и быстрота понимания вашего кода, как другими людьми, так и вами через полгода. PEP 257 описывает соглашения, связанные со строками документации python, рассказывает о том, как нужно документировать python код. Цель этого PEP - стандартизировать структуру строк документации: что они должны в себя включать, и как это написать (не касаясь вопроса синтаксиса строк документации). Этот PEP описывает соглашения, а не правила или синтаксис.

## 9. В чем особенность однострочных и многострочных форм строк документации?

Одиночные строки документации предназначены для действительно очевидных случаев.

```
def kos_root():  
    """Return the pathname of the KOS root directory."""  
    global _kos_root  
    if _kos_root: return _kos_root
```

Многострочные строки документации состоят из однострочной строки документации с последующей пустой строкой, а затем более подробным описанием. Первая строка может быть использована автоматическими средствами индексации, поэтому важно, чтобы она находилась на одной строке и была отделена от остальной документации пустой строкой. Первая строка может быть на той же строке, где и открывающие кавычки, или на следующей строке. Вся документация должна иметь такой же отступ, как кавычки на первой строке.

```
def complex(real=0.0, imag=0.0):  
    """Form a complex number.  
  
    Keyword arguments:  
    real -- the real part (default 0.0)  
    imag -- the imaginary part (default 0.0)  
  
    """  
    if imag == 0.0 and real == 0.0: return complex_zero  
    ...
```