

**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 по дисциплине:
основы программной инженерии**

Выполнила:

студент группы ПИЖ-б-о-20-1

Лазарева Дарья Олеговна

Проверил:

доцент кафедры инфокоммуникаций

Романкин Р.А.

Ставрополь, 2021 г.

Ход работы:

1. Функции в программировании

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

modul1 x

C:\Users\79616\anaconda3\python.exe

5
6
11
4
3
7
8
6
14

```
print("Сколько бананов и ананасов для обезьян?")
a = int(input())
b = int(input())
print("Всего", a+b, "шт.")
print("Сколько жуков и червей для ежей?")
a = int(input())
b = int(input())
print("Всего", a+b, "шт.")
print("Сколько рыб и моллюсков для выдр?")
a = int(input())
b = int(input())
print("Всего", a+b, "шт.")
```

modul1 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/11/modul1.

Сколько бананов и ананасов для обезьян?

13

5

Всего 18 шт.

Сколько жуков и червей для ежей?

7

8

Всего 15 шт.

Сколько рыб и моллюсков для выдр?

12

13

Всего 25 шт.

2. Оператор def. Вызов функции

```
def countFood():  
    a = int(input())  
    b = int(input())  
    print("Всего", a+b, "шт.")  
  
print("Сколько бананов и ананасов для обезьян?")  
countFood()  
print("Сколько жуков и червей для ежей?")  
countFood()  
print("Сколько рыб и моллюсков для выдр?")  
countFood()
```

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/11/modul

Сколько бананов и ананасов для обезьян?

53

20

Всего 73 шт.

Сколько жуков и червей для ежей?

44

22

Всего 66 шт.

Сколько рыб и моллюсков для выдр?

87

56

Всего 143 шт.

3. Вывод ошибки

```
print("Сколько бананов и ананасов для обезьян?")
countFood()
print("Сколько жуков и червей для ежей?")
countFood()
print("Сколько рыб и моллюсков для выдр?")
countFood()
```

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

countFood()

modul1 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/11/modul1.py

Traceback (most recent call last):

File "D:/УЧЕБА/ОПИ/11/modul1.py", line 2, in <module>

countFood()

NameError: name 'countFood' is not defined

Сколько бананов и ананасов для обезьян?

4. Структура программы путем введения функции

```
import math
import sys

def rectangle():
    a = float(input("Ширина: "))
    b = float(input("Высота: "))
    print(f"Площадь: {a * b}")

def triangle():
    a = float(input("Основание: "))
    h = float(input("Высота: "))
    print(f"Площадь: {0.5 * a * h}")

def circle():
```

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОП
1-прямоугольник, 2-треугольник, 3-круг:
2
Основание: 3
Высота: 4
Площадь: 6.0

1-прямоугольник, 2-треугольник, 3-круг:
3
Радиус: 2
Площадь: 12.566370614359172

5. Локальные и глобальные переменные

```
def triangle():  
    a = float(input("Основание: "))  
    h = float(input("Высота: "))  
    print(f"Площадь: {0.5 * a * h}")  
  
figure = input("1-прямоугольник, 2-треугольник: ")  
if figure == '1':  
    rectangle()  
elif figure == '2':  
    triangle()
```

modul1 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/11/modul1

1-прямоугольник, 2-треугольник: 1

Ширина: 3

Высота: 6

Площадь: 18.0

6. Обращение из функции к глобальным переменным

```
import math
import sys

def rectangle():
    a = float(input("Ширина %s: " % figure))
    b = float(input("Высота %s: " % figure))
    print(f"Площадь: {a * b}")

def triangle():
    a = float(input("Основание %s: " % figure))
    h = float(input("Высота %s: " % figure))
    print(f"Площадь: {0.5 * a * h}")

elif figure == '2'
```

modul1 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/11/m
1-прямоугольник, 2-треугольник: 2
Основание 2: 4
Высота 2: 2
Площадь: 4.0

7. Применение команды global

```
def triangle():  
    a = float(input("Основание: "))  
    h = float(input("Высота: "))  
    global result  
    result = 0.5 * a * h  
  
    print("1-прямоугольник, 2-треугольник: ")  
    figure = input()  
    if figure == '1':  
        rectangle()  
    elif figure == '2':  
        triangle()  
    print("Площадь: %.2f" % result)
```

triangle()

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ

1-прямоугольник, 2-треугольник:
1
Ширина: 5
Высота: 3
Площадь: 15.00

8. Применение оператора return

```
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 full
10
11 square = cylinder()
12 print(square)
```

cylinder()

modul1 x

C:\Users\79616\anaconda3\python.exe D:

3

7

188.4955592153876

```
try:
    r = float(input())
    h = float(input())
except ValueError:
    return

side = 2 * math.pi * r * h
circle = math.pi * r**2
full = side + 2 * circle
return full

print(cylinder())
```

cylinder()

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/11

3

7

188.4955592153876

9. Возврат нескольких значений

```
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 side, full

scyl, fcyl = cylinder()
print(f"Площадь боковой поверхности {scyl}")
print(f"Полная площадь {fcyl}")
```

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/11/modul1

3
7

Площадь боковой поверхности 131.94689145077132
Полная площадь 188.4955592153876

10. Произвольное количество аргументов

```
def one_or_many(*a):
    print(a)

one_or_many(1)
one_or_many('1', 1, 2, 'abc')
one_or_many()
```

one_or_many()

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/11/modul1

(1,)
('1', 1, 2, 'abc')
()

11. lambda-функции

```
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)))
```

modul1 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/11/mod
[18, 9, 24, 12, 27]
[14, 46, 28, 54, 44, 58, 26, 34, 64]

Пример 1:

12. Вывод списка всех команд

```
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
exit - завершить работу с программой.
```

13. Результат применения команды add (добавление)

```
>>> add
Фамилия и инициалы? Иванов И.П.
Должность? директор
Год поступления? 2015
>>> add
Фамилия и инициалы? Себастьянов С.С.,
Должность? охранник
Год поступления? 2018
```

14. Вывод всех работников

```
>>> List
```

No	Ф.И.О.	Должность	Год
1	Иванов И.П.	директор	2015
2	Себостьянов С.С,	охранник	2018

15. Вывод работников со стажем от 5 лет

```
>>> select 5
```

No	Ф.И.О.	Должность	Год
1	Иванов И.П.	директор	2015

Выполнение индивидуального задания: Решить индивидуальное задание лабораторной работы 2.6, оформив каждую команду в виде отдельной функции.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import datetime
from datetime import date

import sys

if __name__ == '__main__':
    workers = []

    while True:
        command = input(">>> ").lower()

        if command == 'exit':
            break

        elif command == 'add':
            surname = input("Фамилия: ")
            name = input("Имя: ")
            number = int(input("Номер телефона: "))
            date_obj = input("Дата рождения: ").split('.')

            worker = {
                'surname': surname,
                'name': name,
                'number': number,
                'date_obj': date_obj,
            }

            workers.append(worker)
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get('name', ''))

        elif command == 'list':
            for num, elem in enumerate(workers):
                print(f"{num+1}. \n{str(elem['surname'])} \n"
                    f"Номер телефона: {str(elem['number'])} \n Дата"
                    f"рождения: {elem['date_obj']}")

        elif command.startswith('select'):
            surname = input("Введите фамилию: ")
            for elem in workers:
                if elem['surname'] == surname:
                    print(f"Имя {str(elem['name'])} \nНомер телефона:"
                        f" {str(elem['number'])} \n"
                        f"Дата рождения: {elem['date_obj']}")
                    break
            else:
                print("Фамилии не найдено")

        elif command == 'help':
            print("Список команд:\n")
            print("add - добавить человека;")
            print("list - вывести список всех людей;")
            print("select - найти данные по фамилии;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")
```

```
else:  
    print(f"Неизвестная команда {command}", file=sys.stderr)
```

16. Вывод списка всех команд

```
>>> help  
Список команд:  
  
add - добавить человека;  
list - вывести список всех людей;  
select - найти данные по фамилии;  
help - отобразить справку;  
exit - завершить работу с программой.
```

16. Выполнение команды add

```
>>> add  
Фамилия: mironov  
Имя: v  
Номер телефона: 3456  
Дата рождения: 12.06.2000  
>>> add  
Фамилия: kipelova  
Имя: m  
Номер телефона: 5678  
Дата рождения: 12.12.2002  
>>> add  
Фамилия: lazarev  
Имя: e  
Номер телефона: 5555  
Дата рождения: 31.01.2001
```

17. Вывод всех людей с их данными

```
>>> list
1.
lazareva d
Номер телефона: 1
Дата рождения: ['01', '02', '3003']
2.
lazarev e
Номер телефона: 5555
Дата рождения: ['31', '01', '2001']
3.
kipelova m
Номер телефона: 5678
Дата рождения: ['12', '12', '2002']
4.
lazarev s
Номер телефона: 1234
Дата рождения: ['01', '02', '2003']
5.
mironov v
Номер телефона: 3456
Дата рождения: ['12', '06', '2000']
>>>
```

18. Вывод человека по заданной фамилии

```
>>> select
Введите фамилию: lazareva
Имя d
Номер телефона: 1
Дата рождения: ['01', '02', '3003']
```


Контрольные 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  
    ...
```