

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ**  
**ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2.8**  
**дисциплины «Программирование на Python»**

Выполнил:  
Степанов Леонид Викторович  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение  
средств вычислительной техники  
и автоматизирование систем»,  
очная форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А., канд. техн. наук,  
доцент, доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

Тема: Работа с функциями в Python

Цель: приобретение навыков по работе с функциями при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы:

1. Создал файл (primer.py), в котором проработал пример 1, в нем из примера лабораторной номер 8:

```
PS C:\Users\Ileo\Desktop\Python-labs\lab2.8.11\prog> python3 .\primer1.py
>>> add
Фамилия и инициалы? Степанов Л.В.
Должность? Студент
Год поступления? 2021
>>> add Иванов И.И.
Неизвестная команда add иванов и.и.
>>> add
Фамилия и инициалы? Иванов И.И.
Должность? ИИ
Год поступления? 0000
>>> list
```

№	Ф.И.О.	Должность	Год
1	Иванов И.И.	ИИ	0
2	Степанов Л.В.	Студент	2021

Рисунок 1 – Результат выполнения программы primer.py

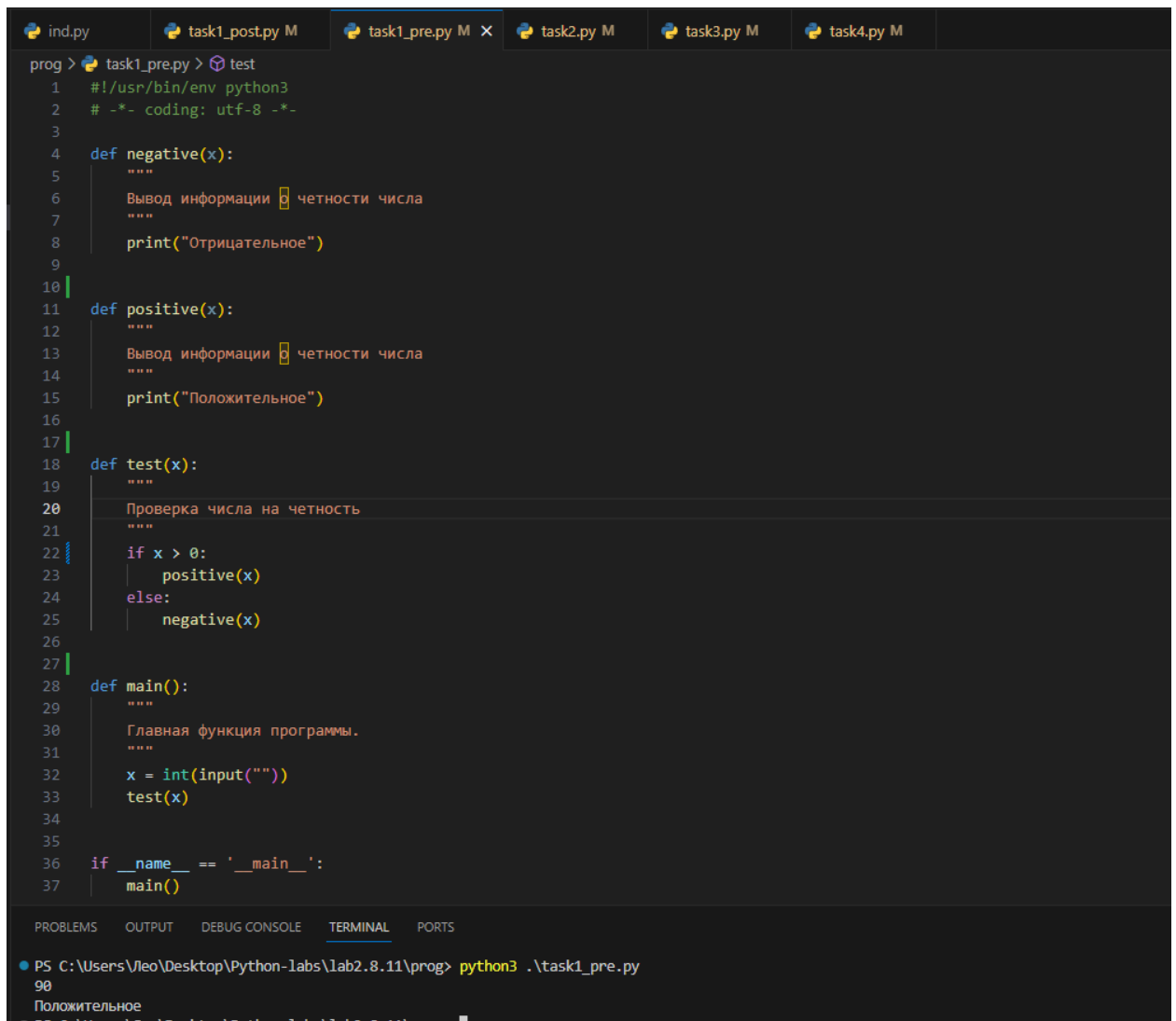
2. Создал файл (task1.py), в котором решил задачу: основная ветка программы, не считая заголовков функций, состоит из двух строки кода. Это вызов функции test() и инструкции if `__name__ == '__main__'` . В ней запрашивается на ввод целое число. Если оно положительное, то вызывается функция positive(), тело которой содержит команду вывода на экран слова "Положительное". Если число отрицательное, то вызывается функция negative(), ее тело содержит выражение вывода на экран слова "Отрицательное".

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  def test(x):
6      """
7      Проверка числа на четность
8      """
9      if x % 2 == 0:
10         positive(x)
11     else:
12         negative(x)
13
14
15  def negative(x):
16      """
17      Вывод информации о четности числа
18      """
19      print("Отрицательное")
20
21
22  def positive(x):
23      """
24      Вывод информации о четности числа
25      """
26      print("Положительное")
27
28
29  def main():
30      """
31      Главная функция программы.
32      """
33      x = int(input(""))
34      test(x)
35
36
37  if __name__ == '__main__':
38      main()
39
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS C:\Users\leo\Desktop\Python-labs\lab2.8.11\prog> python3 .\task1_post.py
-1
Отрицательное
PS C:\Users\leo\Desktop\Python-labs\lab2.8.11\prog> 
```

Рисунок 2 – Результат выполнения программы task1\_post.py



```
ind.py task1_post.py M task1_pre.py M X task2.py M task3.py M task4.py M
prog > task1_pre.py > test
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def negative(x):
5      """
6      Вывод информации о четности числа
7      """
8      print("Отрицательное")
9
10
11 def positive(x):
12     """
13     Вывод информации о четности числа
14     """
15     print("Положительное")
16
17
18 def test(x):
19     """
20     Проверка числа на четность
21     """
22     if x > 0:
23         positive(x)
24     else:
25         negative(x)
26
27
28 def main():
29     """
30     Главная функция программы.
31     """
32     x = int(input(""))
33     test(x)
34
35
36 if __name__ == '__main__':
37     main()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Leo\Desktop\Python-labs\lab2.8.11\prog> python3 .\task1_pre.py
90
Положительное
```

Рисунок 3 – Результат выполнения программы task1\_pre.py

3. Создал файл (task2.py), в котором решил задачу: Решите следующую задачу: в основной ветке программы вызывается функция cylinder(), которая вычисляет площадь цилиндра. В теле cylinder() определена функция circle(), вычисляющая площадь круга по формуле  $\pi r^2$ . В теле cylinder() у пользователя спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле  $2\pi rh$ , или полную площадь цилиндра. В последнем случае к площади боковой поверхности цилиндра должен добавляться удвоенный результат вычислений функции circle().

```
prog > task2.py > main
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  from math import pi
5
6  def cylinder(r, h):
7      """
8      Вычисляет площадь цилиндра.
9      """
10     def circle(r) -> int:
11         return pi * r * r
12     x = input("Получить площадь боковой поверхности? Да - Нет: ")
13     if x == "Да" or x == "да":
14         return 2 * pi * r * h
15
16     return 2 * circle(r) + 2 * pi * r * h
17
18
19 def main():
20     """
21     Главная функция программы.
22     """
23     print(cylinder(3, 4))
24
25
26 if __name__ == '__main__':
27     main()
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

- PS C:\Users\Leo\Desktop\Python-labs\lab2.8.11\prog> python3 .\task2.py  
Получить площадь боковой поверхности? Да - Нет: Да  
75.39822368615503
- PS C:\Users\Leo\Desktop\Python-labs\lab2.8.11\prog> python3 .\task2.py  
Получить площадь боковой поверхности? Да - Нет: Нет  
131.94689145077132

Рисунок 3 – Результат выполнения программы task2.py

4. Создал файл (task3.py), в котором решил задачу: решите следующую задачу: напишите функцию, которая считывает с клавиатуры числа и перемножает их до тех пор, пока не будет введен 0. Функция должна возвращать полученное произведение. Вызовите функцию и выведите на экран результат ее работы.

```
prog > task3.py > work_up_to
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def work_up_to():
5      """
6      Считывает и перемножает числа.
7      """
8      work = 1
9
10     while True:
11         x = int(input())
12         if x == 0:
13             break
14         work *= x
15
16     return work
17
18
19 def main():
20     """
21     Главная функция программы.
22     """
23     print(work_up_to())
24
25
26 if __name__ == "__main__":
27     main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● PS C:\Users\Leo\Desktop\Python-labs\lab2.8.11\prog> python3 .\task3.py
1
2
2
2
2
0
16
○ PS C:\Users\Leo\Desktop\Python-labs\lab2.8.11\prog> |
```

Рисунок 4 – Результат выполнения программы task3.py

5. Создал файл (task4.py), в котором решил задачу:

Решите следующую задачу: напишите программу, в которой определены следующие четыре функции:

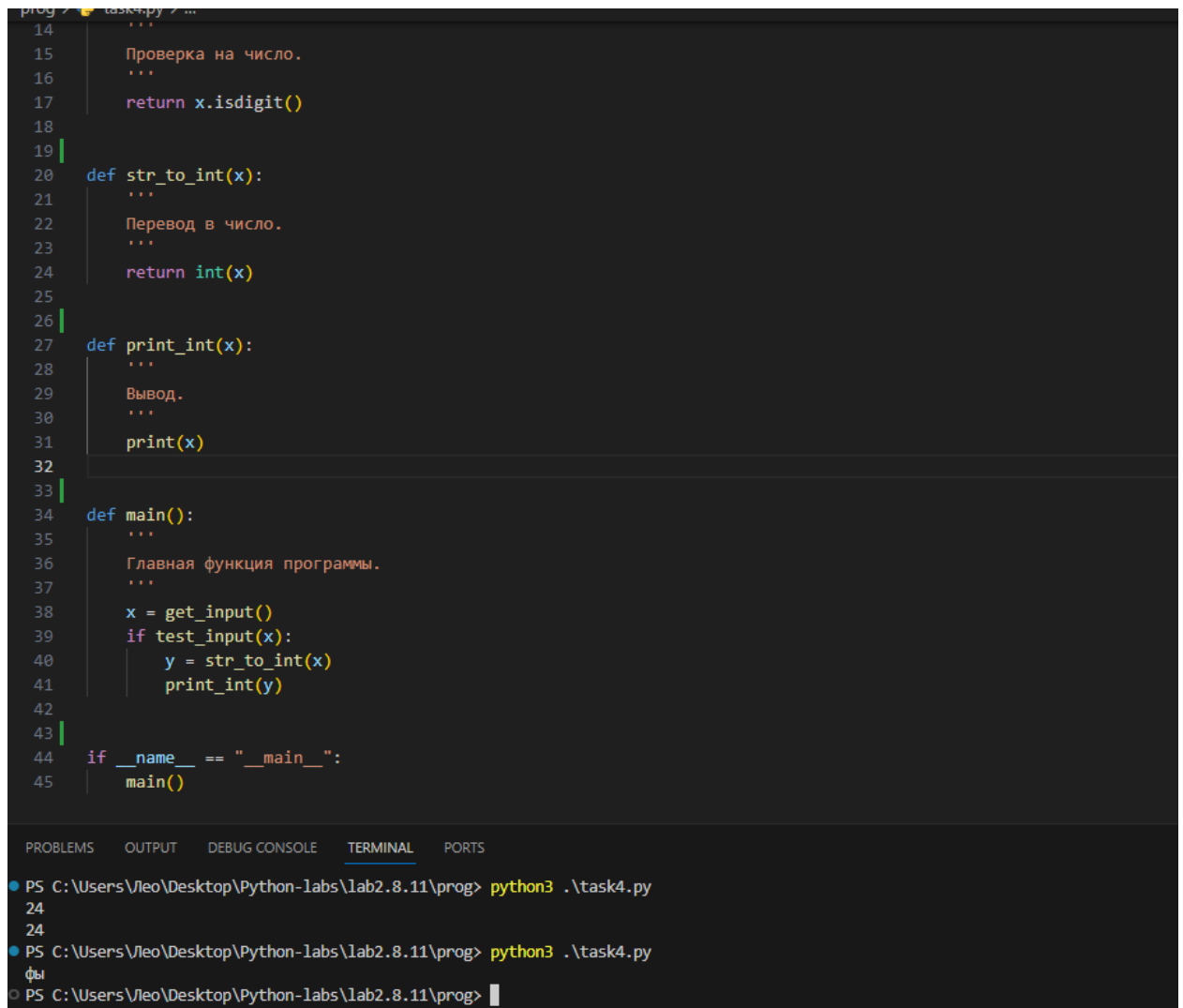
1 Функция `get_input()` не имеет параметров, запрашивает ввод с клавиатуры и возвращает в основную программу полученную строку.

2 Функция `test_input()` имеет один параметр. В теле она проверяет, можно ли переданное ей значение преобразовать к целому числу. Если можно, возвращает логическое `True`. Если нельзя – `False`.

3 Функция `str_to_int()` имеет один параметр. В теле преобразовывает переданное значение к целочисленному типу. Возвращает полученное число.

4 Функция `print_int()` имеет один параметр. Она выводит переданное значение на экран и ничего не возвращает.

В основной ветке программы вызовите первую функцию. То, что она вернула, передайте во вторую функцию. Если вторая функция вернула `True`, то те же данные (из первой функции) передайте в третью функцию, а возвращенное третьей функцией значение – в четвертую.



```
14     ...
15     Проверка на число.
16     ...
17     return x.isdigit()
18
19
20 def str_to_int(x):
21     ...
22     Перевод в число.
23     ...
24     return int(x)
25
26
27 def print_int(x):
28     ...
29     Вывод.
30     ...
31     print(x)
32
33
34 def main():
35     ...
36     Главная функция программы.
37     ...
38     x = get_input()
39     if test_input(x):
40         y = str_to_int(x)
41         print_int(y)
42
43
44 if __name__ == "__main__":
45     main()
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
PS C:\Users\Ieo\Desktop\Python-labs\lab2.8.11\prog> python3 .\task4.py
24
24
PS C:\Users\Ieo\Desktop\Python-labs\lab2.8.11\prog> python3 .\task4.py
фы
PS C:\Users\Ieo\Desktop\Python-labs\lab2.8.11\prog> |
```

Рисунок 5 – Результат выполнения программы `task4.py`

Индивидуальное задание (17 Вариант): решить индивидуальное задание лабораторной работы 2.6, оформив каждую команду в виде отдельной функции.

```
PS C:\Users\Leo\Desktop\Python-labs\lab2.8.11\prog> python3 .\ind.py
>>> add
Имя товара: sugar
Имя магазина: magnit
Стоимость товара: 100
>>> add
Имя товара: bred
Имя магазина: magit
Стоимость товара: 35
>>> add
Имя товара: salt
Имя магазина: magnit
Стоимость товара: 10000
>>> list
+-----+-----+-----+-----+
| № | Название продукта | Имя магазина | Стоимость |
+-----+-----+-----+-----+
| 1 | sugar | magnit | 100 |
| 2 | bred | magit | 35 |
| 3 | salt | magnit | 10000 |
+-----+-----+-----+-----+
>>> info sugar
+-----+-----+-----+-----+
| № | Название продукта | Имя магазина | Стоимость |
+-----+-----+-----+-----+
| 1 | sugar | magnit | 100 |
+-----+-----+-----+-----+
>>> sd
Неизвестная команда sd
>>> exit
```

Рисунок 6 – Результат выполнения программы ind.py

Вывод: в ходе выполнения работы были приобретены навыки по работе с функциями при написании программ с помощью языка программирования Python версии 3.x.

### Контрольные вопросы

1. Каково назначение функций в языке программирования Python?

Функции в языке программирования Python используются для группировки кода, чтобы он мог быть многократно использован, делая программу более читаемой и легко управляемой. Функции также позволяют разделить большие программы на более мелкие, что упрощает разработку и управление кодом.



## 2. Каково назначение операторов def и return ?

Оператор def используется для определения функции в Python. Он указывает интерпретатору, что следующий блок кода является телом функции. Оператор return используется для возврата значения из функции. Когда интерпретатор Python достигает оператора return, он возвращает указанное значение и завершает выполнение функции.

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

Локальные переменные объявляются внутри функции и доступны только внутри этой функции. Глобальные переменные объявляются вне функций и доступны во всем коде программы. При написании функций в Python, локальные переменные используются для временного хранения данных, в то время как глобальные переменные могут быть использованы в различных частях программы.

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

Для возврата нескольких значений из функции в Python используется механизм кортежей. Функция может вернуть кортеж, содержащий несколько значений, и затем эти значения могут быть присвоены различным переменным при вызове функции.

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

Значения могут быть переданы в функцию в Python через позиционные аргументы, ключевые аргументы и аргументы по умолчанию. Позиционные аргументы передаются в порядке, в котором они определены в сигнатуре функции. Ключевые аргументы передаются с указанием имени параметра. Аргументы по умолчанию имеют значения по умолчанию и могут быть пропущены при вызове функции.

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

Для задания значения аргументов функции по умолчанию в Python используется синтаксис "переменная=значение" в сигнатуре функции. Если

значение не передается при вызове функции, будет использовано значение по умолчанию.

7. Каково назначение lambda-выражений в языке Python?

Lambda-выражения в Python представляют собой анонимные функции, которые могут содержать только одно выражение. Они обычно используются в ситуациях, когда требуется небольшая функция в одном месте кода.

8. Как осуществляется документирование кода согласно PEP257?

Документирование кода в Python согласно PEP257 включает в себя использование строк документации (docstrings) для описания модулей, функций, классов и методов. Строки документации должны быть заключены в тройные кавычки и предоставлять информацию о назначении, использовании и возвращаемых значениях функций, классов и методов.

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

Однострочные строки документации начинаются и заканчиваются тройными кавычками и предназначены для краткого описания модулей, функций, классов или методов. Они обычно используются для кратких пояснений и описаний.

Многострочные строки документации также начинаются и заканчиваются тройными кавычками, но могут занимать несколько строк и предоставлять более подробное описание. Они обычно используются для более полного и подробного документирования, включая информацию о назначении, использовании и возвращаемых значениях функций, классов и методов.