

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

**Отчет о лабораторной работе №11 по дисциплине основы программной
инженерии**

Выполнил:
Выходцев Егор Дмитриевич,
2 курс, группа ПИЖ-б-о-20-1,

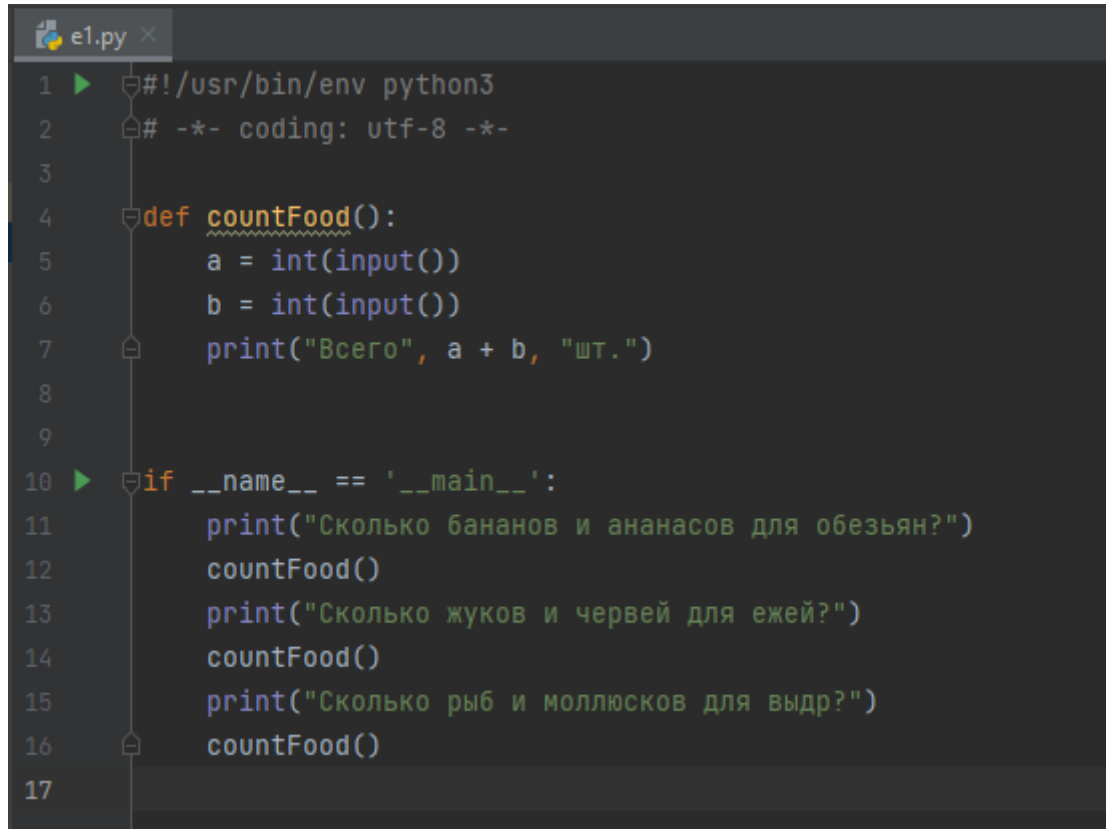
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г.

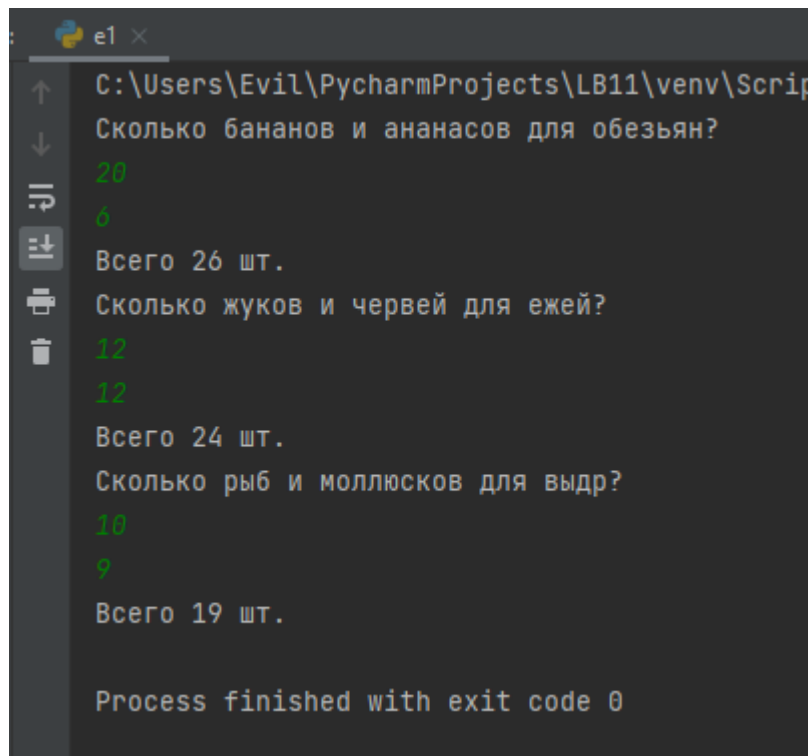
1. Работа с функциями в языке Python

Примеры из методических указаний

Определение функции. Оператор def. Вызов функции



```
e1.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      def countFood():
5          a = int(input())
6          b = int(input())
7          print("Всего", a + b, "шт.")
8
9
10  ▶  if __name__ == '__main__':
11      print("Сколько бананов и ананасов для обезьян?")
12      countFood()
13      print("Сколько жуков и червей для ежей?")
14      countFood()
15      print("Сколько рыб и моллюсков для выдр?")
16      countFood()
17
```

A screenshot of a terminal window titled 'e1 x' showing the execution of a Python script. The script asks for the number of bananas and pineapples for monkeys, then asks for the number of beetles and worms for hedgehogs, and finally asks for the number of fish and mollusks for beavers. The user enters the numbers 20, 6, 12, 12, 10, and 9. The script calculates the total for each animal group and prints the results. The terminal output is as follows:

```
C:\Users\Evil\PycharmProjects\LB11\venv\Scripts>python script.py
Сколько бананов и ананасов для обезьян?
20
6
Всего 26 шт.
Сколько жуков и червей для ежей?
12
12
Всего 24 шт.
Сколько рыб и моллюсков для выдр?
10
9
Всего 19 шт.

Process finished with exit code 0
```

Функции придают программе структуру

```
e1.py x e2.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import math
5 import sys
6
7
8 def rectangle():
9     a = float(input("Ширина: "))
10    b = float(input("Высота: "))
11    print(f"Площадь: {a * b}")
12
13
14 def triangle():
15     a = float(input("Основание: "))
16     h = float(input("Высота: "))
17     print(f"Площадь: {0.5 * a * h}")
18
19
20 def circle():
21     r = float(input("Радиус: "))
22     print(f"Площадь: {math.pi * r**2}")
23
24
25 ▶ if __name__ == '__main__':
26     figure = input("1-прямоугольник, 2-треугольник, 3-круг: ")
27     if figure == '1':
28         rectangle()
29     elif figure == '2':
30         triangle()
31     elif figure == '3':
32         circle()
33     else:
34         print("Ошибка ввода", file=sys.stderr)
35
```

```
e2 x
C:\Users\Evil\PycharmProjects\LB11\venv\Scripts\python.exe
1-прямоугольник, 2-треугольник, 3-круг: 1
Ширина: 12
Высота: 2
Площадь: 24.0
Process finished with exit code 0
```

```
e2 x
C:\Users\Evil\PycharmProjects\LB11\venv\Script
1-прямоугольник, 2-треугольник, 3-круг: 2
Основание: 24
Высота: 10
Площадь: 120.0
Process finished with exit code 0
```

```
e2 x
C:\Users\Evil\PycharmProjects\LB11\ve
1-прямоугольник, 2-треугольник, 3-кру
Радиус: 5
Площадь: 78.53981633974483
Process finished with exit code 0
```

Локальные и глобальные переменные

```
e1.py x e2.py x e3.py x
5 import sys
6
7
8 def rectangle():
9     a = float(input("Ширина %s: " % figure))
10    b = float(input("Высота %s: " % figure))
11    print(f"Площадь: {a * b}")
12
13
14 def triangle():
15     a = float(input("Основание %s: " % figure))
16     h = float(input("Высота %s: " % figure))
17     print(f"Площадь: {0.5 * a * h}")
18
19
20 if __name__ == '__main__':
21     figure = input("1-прямоугольник, 2-треугольник: ")
22     if figure == '1':
23         rectangle()
24     elif figure == '2':
25         triangle()
26     else:
27         print("Ошибка ввода", file=sys.stderr)
28
```

```
e3 x
C:\Users\Evil\PycharmProjects\LB11\venv\Script
1-прямоугольник, 2-треугольник: 2
Основание 2: 10
Высота 2: 12
Площадь: 60.0
Process finished with exit code 0
```

Возврат значений из функции. Оператор return

```
e4.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import math
5
6
7 def cylinder():
8     r = float(input())
9     h = float(input())
10    # площадь боковой поверхности цилиндра:
11    side = 2 * math.pi * r * h
12    # площадь одного основания цилиндра:
13    circle = math.pi * r**2
14    # полная площадь цилиндра:
15    full = side + 2 * circle
16    return full
17
18
19 ▶ if __name__ == '__main__':
20     square = cylinder()
21     print(square)
22
```

```
e4 x
C:\Users\Evil\PycharmProjects\LB11\ve
3
7
188.4955592153876
Process finished with exit code 0
```

Возврат нескольких значений

```
e5.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import math
5
6
7 def cylinder():
8     r = float(input())
9     h = float(input())
10    side = 2 * math.pi * r * h
11    circle = math.pi * r**2
12    full = side + 2 * circle
13    return side, full
14
15
16 ▶ if __name__ == '__main__':
17     scyl, fcyl = cylinder()
18     print(f"Площадь боковой поверхности {scyl}")
19     print(f"Полная площадь {fcyl}")
20
```

```
e5 x
C:\Users\Evil\PycharmProjects\LB11\venv\Scripts\python.exe
5
10
Площадь боковой поверхности 314.1592653589793
Полная площадь 471.23889803846896
Process finished with exit code 0
```

Произвольное количество аргументов


```
e6.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      import math
5
6
7      def cylinder(h, r=1):
8          side = 2 * math.pi * r * h
9          circle = math.pi * r**2
10         full = side + 2 * circle
11         return full
12
13
14  ▶  if __name__ == '__main__':
15      figure1 = cylinder(4, 3)
16      figure2 = cylinder(5)
17      print(figure1)
18      print(figure2)
19      figure3 = cylinder(10, 2)
20      figure4 = cylinder(r=2, h=10)
21      print(figure3)
22      print(figure4)
23
```

```
e6 x
C:\Users\Evil\PycharmProjects\LB11\venv
131.94689145077132
37.69911184307752
150.79644737231007
150.79644737231007
Process finished with exit code 0
```

lambda – функции

```
e7.py x
1 ▶ #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == '__main__':
5     foo = [2, 18, 9, 22, 17, 24, 8, 12, 27]
6     print(list(filter(lambda x: x % 3 == 0, foo)))
7     print(list(map(lambda x: x * 2 + 10, foo)))
8
```

```
e7 x
C:\Users\Evil\PycharmProjects\LB11\venv\Scripts\python.exe
[18, 9, 24, 12, 27]
[14, 46, 28, 54, 44, 58, 26, 34, 64]
Process finished with exit code 0
```

1.1 Пример 1 (рис. 1, 2, 3, 4, 5, 6).

```
ex1.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      import sys
5      from datetime import date
6
7
8      def get_worker():
9          """
10             Запросить данные о работнике.
11          """
12             name = input("Фамилия и инициалы? ")
13             post = input("Должность? ")
14             year = int(input("Год поступления? "))
15             # Создать словарь.
16             return {
17                 'name': name,
18                 'post': post,
19                 'year': year,
20             }
21
22
23     def display_workers(staff):
24         """
25             Отобразить список работников.
26         """
27         # Проверить, что список работников не пуст.
28         if staff:
29             # Заголовок таблицы.
30             line = '+-{}-+-{}-+-{}-+-{}-+'.format(
31                 '-' * 4,
32                 '-' * 30,
33                 '-' * 20,
34                 '-' * 8
35             )
36             print(line)
37             print(
38                 '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
39                     "No",
```

Рисунок 1 – Код программы

```
ex1.py x
38         | {:>4} | {:<30} | {:<20} | {:>8} |'.format(
39             "No",
40             "Ф.И.О.",
41             "Должность",
42             "Год"
43         )
44     )
45     print(line)
46     # Вывести данные о всех сотрудниках.
47     for idx, worker in enumerate(staff, 1):
48         print(
49             '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
50                 idx,
51                 worker.get('name', ''),
52                 worker.get('post', ''),
53                 worker.get('year', 0)
54             )
55         )
56         print(line)
57     else:
58         print("Список работников пуст.")
59
60
61 def select_workers(staff, period):
62     """
63     Выбрать работников с заданным стажем.
64     """
65     # Получить текущую дату.
66     today = date.today()
67     # Сформировать список работников.
68     result = []
69     for employee in staff:
70         if today.year - employee.get('year', today.year) >= period:
71             result.append(employee)
72     # Возвратить список выбранных работников.
73     return result
74
75
76 def main():
77     """
```

Рисунок 2 – Код программы, продолжение

```
ex1.py x
75
76 def main():
77     """
78     Главная функция программы.
79     """
80     # Список работников.
81     workers = []
82     # Организовать бесконечный цикл запроса команд.
83     while True:
84         # Запросить команду из терминала.
85         command = input(">>> ").lower()
86         # Выполнить действие в соответствие с командой.
87         if command == 'exit':
88             break
89         elif command == 'add':
90             # Запросить данные о работнике.
91             worker = get_worker()
92             # Добавить словарь в список.
93             workers.append(worker)
94             # Отсортировать список в случае необходимости.
95             if len(workers) > 1:
96                 workers.sort(key=lambda item: item.get('name', ''))
97         elif command == 'list':
98             # Отобразить всех работников.
99             display_workers(workers)
100         elif command.startswith('select '):
101             # Разбить команду на части для выделения стажа.
102             parts = command.split(' ', maxsplit=1)
103             # Получить требуемый стаж.
104             period = int(parts[1])
105             # Выбрать работников с заданным стажем.
106             selected = select_workers(workers, period)
107             # Отобразить выбранных работников.
108             display_workers(selected)
109         elif command == 'help':
110             # Вывести справку о работе с программой.
111             print("Список команд:\n")
112             print("add - добавить работника;")
113             print("list - вывести список работников;")
114             print("exit - завершить работу с программой.\n")
115     return
```

Рисунок 3 – Код программы, продолжение

```
112         print("add - добавить работника;")
113         print("list - вывести список работников;")
114         print("select <стаж> - запросить работников со стажем;")
115         print("help - отобразить справку;")
116         print("exit - завершить работу с программой.")
117     else:
118         print(f"Неизвестная команда {command}", file=sys.stderr)
119
120
121 if __name__ == '__main__':
122     main()
123
```

Рисунок 4 – Код программы, продолжение

```
ex1 x
C:\Users\Evil\PycharmProjects\LB11\venv\Scripts\python.exe C:\Users\Evil\Pychar
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Фамилия и инициалы? Hjddk E.J
Должность? NHKllj
Год поступления? 2010
>>> add
Фамилия и инициалы? Tujkhkh U.J
Должность? Qsadsfg
Год поступления? 2020
>>> add
Фамилия и инициалы? Oojhh R.U
Должность? Axnffk
Год поступления? 2000
>>> add
Фамилия и инициалы? Ahjfsjkhfs
Должность? Yuhkfs
Год поступления? 2005
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Ahjfsjkhfs              | Yuhkfs              | 2005 |
+-----+-----+-----+-----+
|  2 | Hjddk E.J              | NHKllj              | 2010 |
+-----+-----+-----+-----+
|  3 | Oojhh R.U              | Axnffk              | 2000 |
+-----+-----+-----+-----+
|  4 | Tujkhkh U.J            | Qsadsfg             | 2020 |
+-----+-----+-----+-----+
```

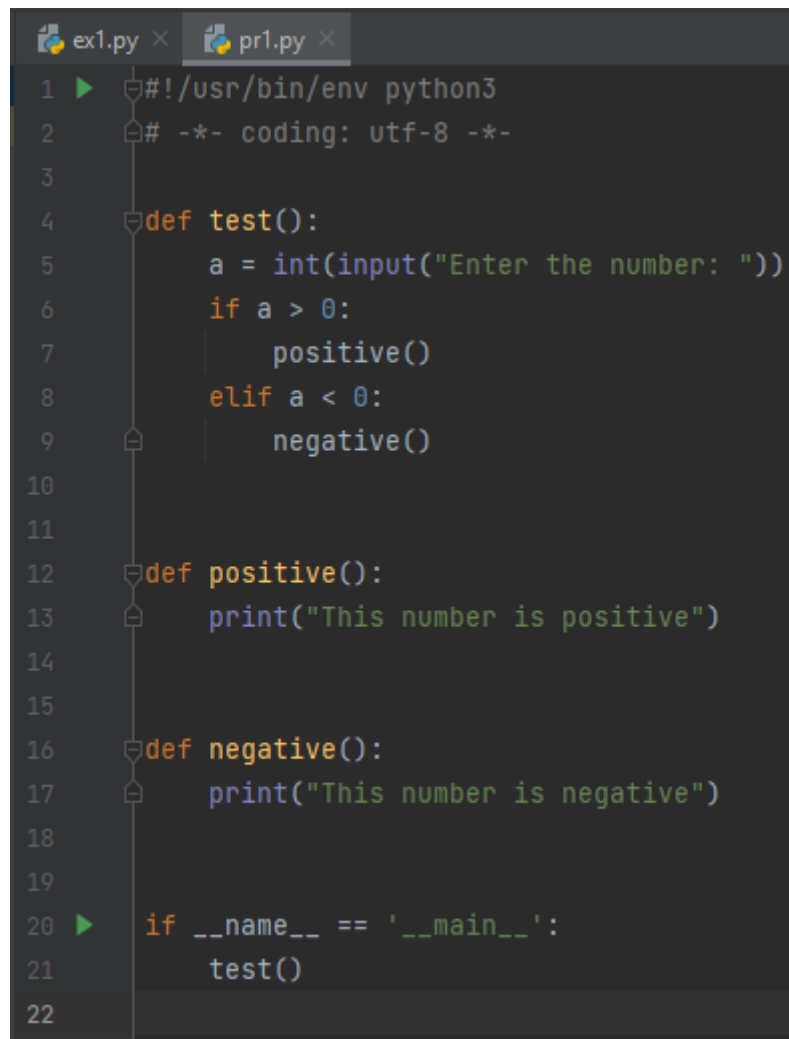
Рисунок 5 – Результат выполнения программы

```
>>> select 10
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Ahjfsjhhkfs              | Yuhkfs              |      2005      |
+-----+-----+-----+-----+
|  2 | Hjddek E.J               | NHkllj              |      2010      |
+-----+-----+-----+-----+
|  3 | Oojhh R.U                | Axnffk              |      2000      |
+-----+-----+-----+-----+
>>> hhjsf
>>> Неизвестная команда hhjsf
exit

Process finished with exit code 0
```

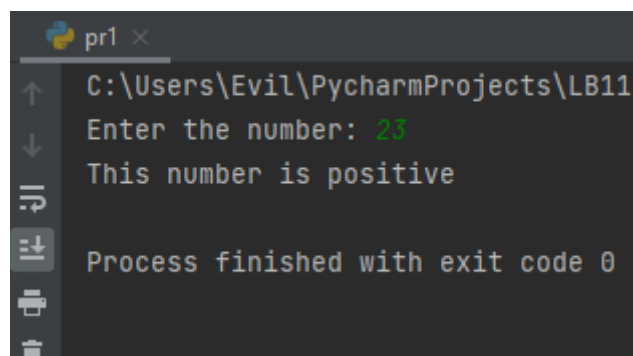
Рисунок 6 – Результат выполнения программы, продолжение

1.2 Задача 1 (рис. 7, 8, 9).



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def test():
5      a = int(input("Enter the number: "))
6      if a > 0:
7          positive()
8      elif a < 0:
9          negative()
10
11
12  def positive():
13      print("This number is positive")
14
15
16  def negative():
17      print("This number is negative")
18
19
20  if __name__ == '__main__':
21      test()
22
```

Рисунок 7 – Код программы



```
pr1 x
C:\Users\Evil\PycharmProjects\LB11\
Enter the number: 23
This number is positive
Process finished with exit code 0
```

Рисунок 8 – Результат выполнения программы при вводе
положительного числа

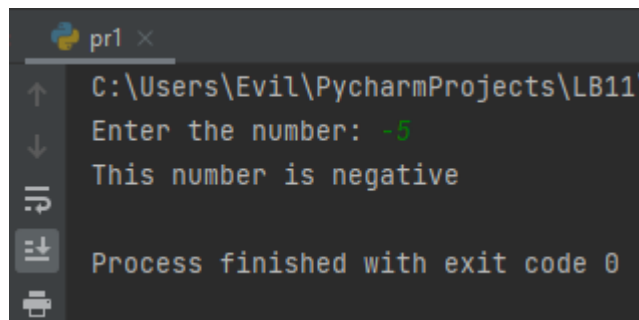
A screenshot of a terminal window titled 'pr1'. The window shows the following text: 'C:\Users\Evil\PycharmProjects\LB11\', 'Enter the number: -5', 'This number is negative', and 'Process finished with exit code 0'. The input '-5' is highlighted in green. On the left side of the terminal, there is a vertical toolbar with icons for navigation (up, down), search, and other standard terminal functions.

Рисунок 9 – Результат выполнения программы при вводе отрицательного числа

Порядок определения функций не имеет значения, так как на момент вызова соответствующих функций из основной все имена функций уже определены. Программа выдала бы ошибку, если попытаться вызвать функцию до ее объявления.

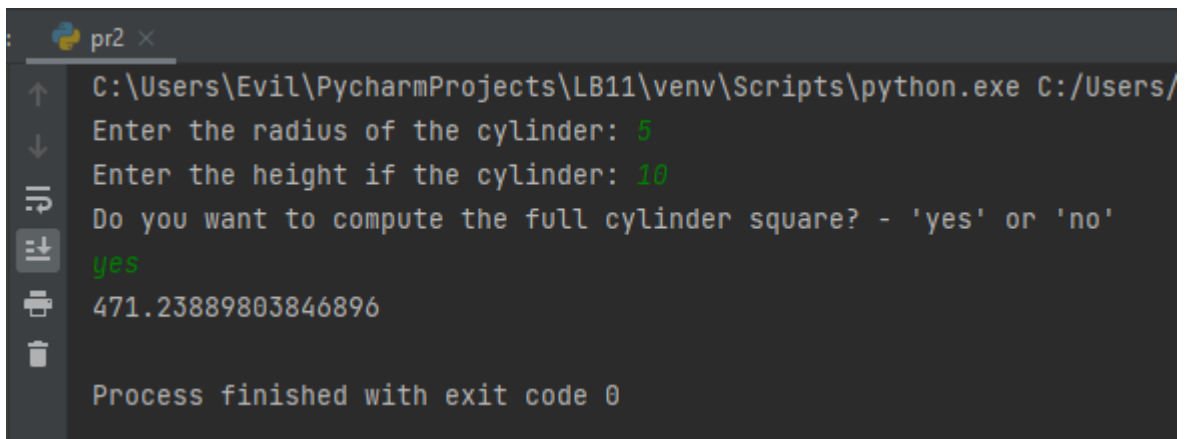
1.3 Задача 2 (рис. 10, 11, 12).

```
ex1.py × pr1.py × pr2.py × pr3.py × pr4.py × ind1.py ×
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import math
5
6
7 def cylinder():
8
9     def circle(r):
10         circle_square = math.pi * r ** 2
11         return circle_square
12
13     radius = float(input("Enter the radius of the cylinder: "))
14     height = float(input("Enter the height if the cylinder: "))
15     print("Do you want to compute the full cylinder square? - 'yes' or 'no'")
16     command = input().lower()
17     if command == 'yes':
18         print(2 * math.pi * radius * height + 2 * circle(radius))
19     if command == 'no':
20         print(2 * math.pi * radius * height)
21
22
23 ▶ if __name__ == '__main__':
24     cylinder()
25
```

Рисунок 10 – Код программы

```
pr2 ×
C:\Users\Evil\PycharmProjects\LB11\venv\Scripts\python.exe C:/Use
Enter the radius of the cylinder: 5
Enter the height if the cylinder: 10
Do you want to compute the full cylinder square? - 'yes' or 'no'
no
314.1592653589793
Process finished with exit code 0
```

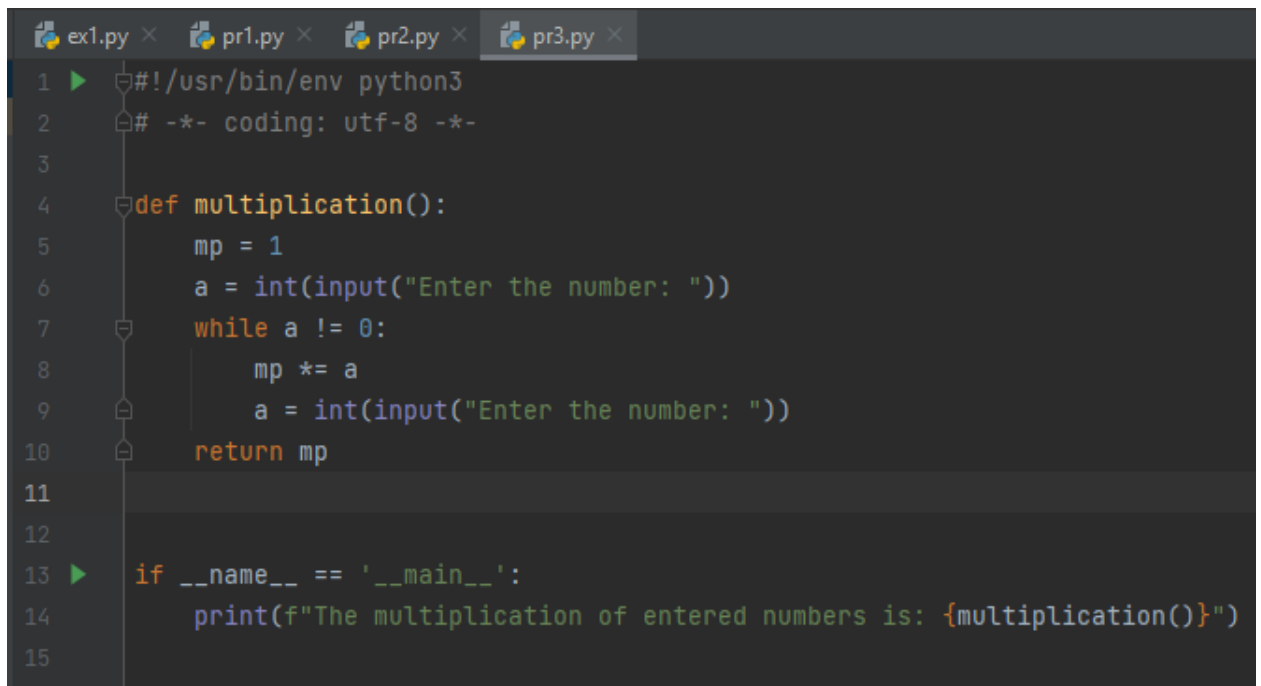
Рисунок 11 – Результат вычисления площади боковой поверхности



```
pr2 x
C:\Users\Evil\PycharmProjects\LB11\venv\Scripts\python.exe C:/Users/
Enter the radius of the cylinder: 5
Enter the height if the cylinder: 10
Do you want to compute the full cylinder square? - 'yes' or 'no'
yes
471.23889803846896
Process finished with exit code 0
```

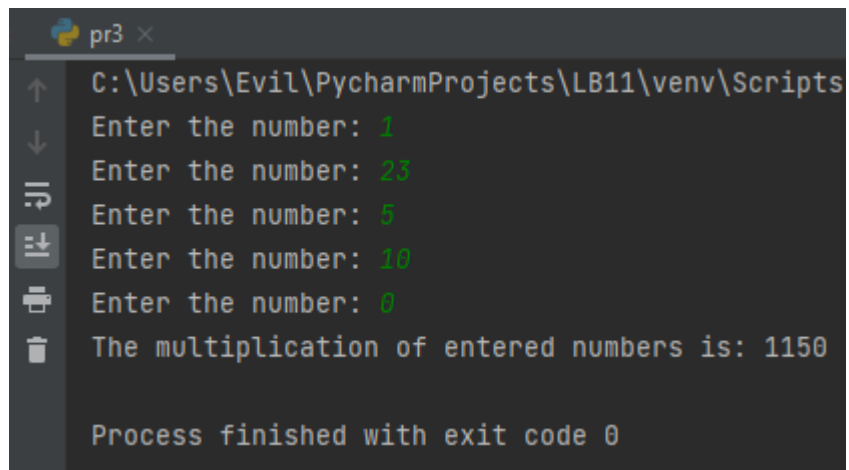
Рисунок 12 – Результат вычисления полной площади цилиндра

1.4 Задача 3 (рис. 13, 14).



```
ex1.py x pr1.py x pr2.py x pr3.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      def multiplication():
5          mp = 1
6          a = int(input("Enter the number: "))
7          while a != 0:
8              mp *= a
9              a = int(input("Enter the number: "))
10         return mp
11
12
13  ▶  if __name__ == '__main__':
14         print(f"The multiplication of entered numbers is: {multiplication()}")
15
```

Рисунок 13 – Код программы



```
pr3 x
C:\Users\Evil\PycharmProjects\LB11\venv\Scripts
Enter the number: 1
Enter the number: 23
Enter the number: 5
Enter the number: 10
Enter the number: 0
The multiplication of entered numbers is: 1150

Process finished with exit code 0
```

Рисунок 14 – Пример работы программы

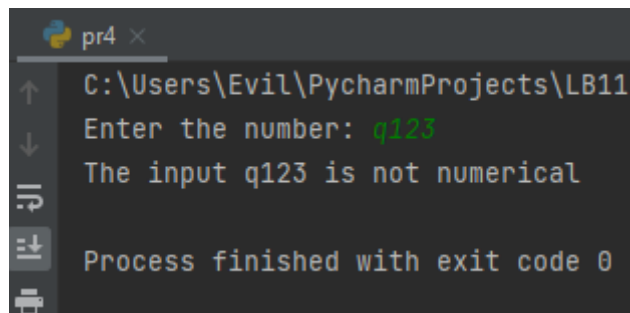
1.5 Задача 4 (рис. 15

```
ex1.py × pr1.py × pr2.py × pr3.py × pr4.py ×
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 def get_input():
5     a = input("Enter the number: ")
6     return a
7
8
9 def test_input(a):
10     return a.isdigit()
11
12
13 def str_to_int(a):
14     a = int(a)
15     return a
16
17
18 def print_int(a):
19     print(a)
20
21
22 ▶ if __name__ == '__main__':
23     num_str = get_input()
24     if test_input(num_str):
25         num = str_to_int(num_str)
26         print_int(num)
27     else:
28         print(f"The input {num_str} is not numerical")
29
```

Рисунок 15 – Код программы

```
pr4 ×
C:\Users\Evil\PycharmProjects\LB11\
Enter the number: 123
123
Process finished with exit code 0
```

Рисунок 16 – Вывод программы при test_input() равном «true»



```
pr4 x
C:\Users\Evil\PycharmProjects\LB11
Enter the number: q123
The input q123 is not numerical
Process finished with exit code 0
```

Рисунок 17 – Вывод программы при `test_input()` равном «false»

1.6 Индивидуальное задание вариант №5. (рис. 18-25).

```
ex1.py x pr1.py x pr2.py x pr3.py x pr4.py x ind1.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6
7 def get_flight():
8     """
9     Запросить данные о полёте
10    """
11    flight_destination = input("Введите название пункта назначения ")
12    flight_number = input("Введите номер рейса ")
13    airplane_type = input("Введите тип самолета ")
14    return {
15        'flight_destination': flight_destination,
16        'flight_number': flight_number,
17        'airplane_type': airplane_type,
18    }
19
20
21 def display_flights(flights):
22     """
23     Отобразить список рейсов
24    """
25    if flights:
26        line = '+-{}--{}--{}--{}--{}--+'.format(
27            '-' * 4,
28            '-' * 30,
29            '-' * 20,
30            '-' * 15
31        )
32        print(line)
33        print(
34            '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
35                "No",
36                "Пункт назначения",
37                "Номер рейса",
38                "Тип самолета"
39            )

```

Рисунок 18 – Код программы


```
ex1.py × pr1.py × pr2.py × pr3.py × pr4.py × ind1.py ×
37         "Номер рейса",
38         "Тип самолета"
39     )
40 )
41 print(line)
42
43     for idx, flight in enumerate(flights, 1):
44         print(
45             '| {:>4} | {:<30} | {:<20} | {:<15} |'.format(
46                 idx,
47                 flight.get('flight_destination', ''),
48                 flight.get('flight_number', ''),
49                 flight.get('airplane_type', 0)
50             )
51         )
52     print(line)
53
54     else:
55         print("Список рейсов пуст")
56
57
58 def select_flights(flights, airplane_type):
59     """
60     Выбрать рейсы самолётов заданного типа
61     """
62     count = 0
63     res = []
64     for flight in flights:
65         if flight.get('airplane_type') == airplane_type:
66             count += 1
67             res.append(flight)
68     if count == 0:
69         print("рейсы не найдены")
70
71     return res
72
73
74 def main():
75     """
```

Рисунок 19 – Код программы, продолжение

```
ex1.py × pr1.py × pr2.py × pr3.py × pr4.py × ind1.py ×
74 def main():
75     """
76     Главная функция программы
77     """
78     flights = []
79     while True:
80         command = input(">>> ").lower()
81         if command == 'exit':
82             break
83
84         elif command == 'add':
85             flight = get_flight()
86             flights.append(flight)
87             if len(flights) > 1:
88                 flights.sort(
89                     key=lambda item:
90                     item.get('flight_destination', ''))
91
92         elif command == 'list':
93             display_flights(flights)
94
95         elif command.startswith('select '):
96             parts = command.split(' ', maxsplit=1)
97             airplane_type = (parts[1].capitalize())
98             print(f"Для типа самолета {airplane_type}:")
99             selected = select_flights(flights, airplane_type)
100             display_flights(selected)
101
102         elif command == 'help':
103             # Вывести справку о работе с программой.
104             print("Список команд:\n")
105             print("add - добавить рейс;")
106             print("list - вывести список всех рейсов;")
107             print("select <тип самолета> - запросить рейсы указанного типа "
108                   "самолета;")
109             print("help - отобразить справку;")
110             print("exit - завершить работу с программой.")
111         else:
112             print(f"Неизвестная команда {command}", file=sys.stderr)
```

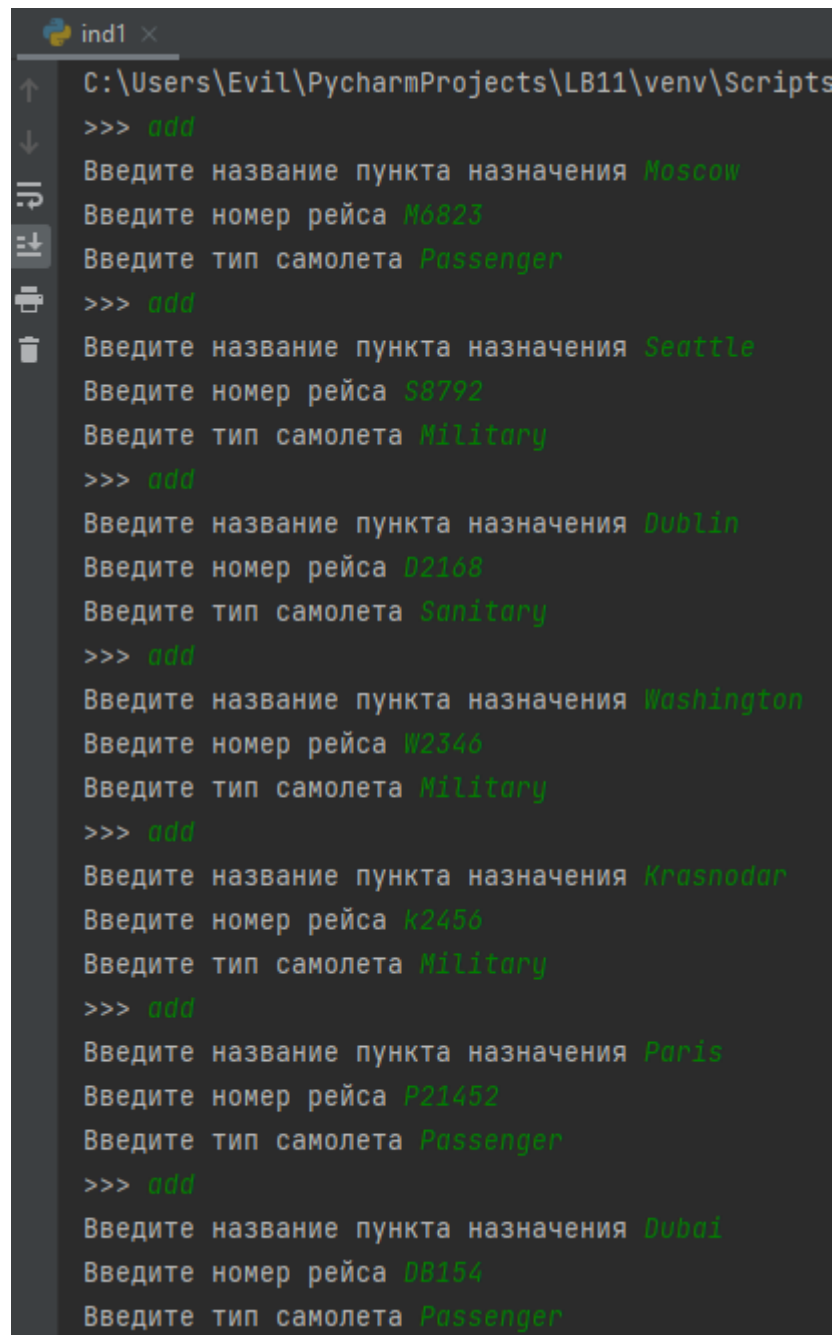
Рисунок 20 – Код программы, продолжение

```

112     print(f"Неизвестная команда {command}", file=sys.stderr)
113
114
115 ► if __name__ == '__main__':
116     main()
117

```

Рисунок 21 – Код программы, продолжение



The screenshot shows a Python REPL session with the following commands and user input:

```

C:\Users\Evil\PycharmProjects\LB11\venv\Scripts
>>> add
Введите название пункта назначения Moscow
Введите номер рейса M0823
Введите тип самолета Passenger
>>> add
Введите название пункта назначения Seattle
Введите номер рейса S8792
Введите тип самолета Military
>>> add
Введите название пункта назначения Dublin
Введите номер рейса D2168
Введите тип самолета Sanitary
>>> add
Введите название пункта назначения Washington
Введите номер рейса W2346
Введите тип самолета Military
>>> add
Введите название пункта назначения Krasnodar
Введите номер рейса K2456
Введите тип самолета Military
>>> add
Введите название пункта назначения Paris
Введите номер рейса P21452
Введите тип самолета Passenger
>>> add
Введите название пункта назначения Dubai
Введите номер рейса DB154
Введите тип самолета Passenger

```

Рисунок 22 – Заполнение словаря

```
>>> list
```

No	Пункт назначения	Номер рейса	Тип самолета
1	Dubai	DB154	Passenger
2	Dublin	D2168	Sanitary
3	Krasnodar	k2456	Military
4	Moscow	M6823	Passenger
5	Paris	P21452	Passenger
6	Seattle	S8792	Military
7	Washington	W2346	Military

Рисунок 23 – Вывод отсортированного словаря

```
>>> select Military
Для типа самолета Military:
```

No	Пункт назначения	Номер рейса	Тип самолета
1	Krasnodar	k2456	Military
2	Seattle	S8792	Military
3	Washington	W2346	Military

```
>>> select Sanitary
Для типа самолета Sanitary:
```

No	Пункт назначения	Номер рейса	Тип самолета
1	Dublin	D2168	Sanitary

```
>>> select Training
Для типа самолета Training:
рейсы не найдены
Список рейсов пуст
```

Рисунок 24 – Вывод пунктов назначения и номеров рейса для указанных типов самолёта

```
>>> gwkr
>>> Неизвестная команда gwkr
help
Список команд:

add - добавить рейс;
list - вывести список всех рейсов;
select <тип самолета> - запросить рейсы указанного типа самолета;
help - отобразить справку;
exit - завершить работу с программой.
>>> exit

Process finished with exit code 0
```

Рисунок 25 – Вывод программы при вводе неправильной команде, а также команда help

2. Ответы на контрольные вопросы

1. Функция в программировании представляет собой обособленный участок кода, который можно вызывать, обратившись к нему по имени, которым он был назван. При вызове происходит выполнение команд тела функции. Функции можно сравнить с небольшими программками, которые сами по себе, т. е. автономно, не исполняются, а встраиваются в обычную программу. Нередко их так и называют – подпрограммы. Других ключевых отличий функций от программ нет. Функции также при необходимости могут получать и возвращать данные. Только обычно они их получают не с ввода (клавиатуры, файла и др.), а из вызывающей программы. Сюда же они возвращают результат своей работы. Внедрение функций позволяет решить проблему дублирования кода в разных местах программы. Благодаря им можно исполнять один и тот же участок кода не сразу, а только тогда, когда он понадобится.

2. В языке программирования Python функции определяются с помощью оператора `def`. Ключевое слово `def` сообщает интерпретатору, что

перед ним определение функции. За `def` следует имя функции. Оно может быть любым, так же, как и всякий идентификатор, например, переменная.

3. Функции могут передавать какие-либо данные из своих тел в основную ветку программы.

Говорят, что функция возвращает значение. В большинстве языков программирования, в том числе Python, выход из функции и передача данных в то место, откуда она была вызвана, выполняется оператором `return`. Если интерпретатор Питона, выполняя тело функции, встречается `return`, то он "забирает" значение, указанное после этой команды, и "уходит" из функции.

4. В программировании особое внимание уделяется концепции о локальных и глобальных переменных, а также связанное с ними представление об областях видимости. Соответственно, локальные переменные видны только в локальной области видимости, которой может выступать отдельно взятая функция. Глобальные переменные видны во всей программе. "Видны" – значит, известны, доступны. К ним можно обратиться по имени и получить связанное с ними значение. К глобальной переменной можно обратиться из локальной области видимости. К локальной переменной нельзя обратиться из глобальной области видимости, потому что локальная переменная существует только в момент выполнения тела функции. При выходе из нее, локальные переменные исчезают. Компьютерная память, которая под них отводилась, освобождается. Когда функция будет снова вызвана, локальные переменные будут созданы заново.

5. В Питоне позволительно возвращать из функции несколько объектов, перечислив их через запятую после команды `return`.

6. В строке объявления функции указать в скобках значение параметра по умолчанию.

7. `lambda` функции позволяют определять небольшие однострочные функции на лету. `lambda` – это выражение, а не инструкция. По этой причине

ключевое слово `lambda` может появляться там, где синтаксис языка Python не позволяет использовать инструкцию `def`, – внутри литералов или в вызовах функций, например. Есть и еще одно интересное применение - хранение списка обработчиков данных в списке/словаре.

8. Строки документации - строковые литералы, которые являются первым оператором в модуле, функции, классе или определении метода. Такая строка документации становится специальным атрибутом `__doc__` этого объекта. Все модули должны, как правило, иметь строки документации, и все функции и классы, экспортируемые модулем также должны иметь строки документации. Публичные методы (в том числе `__init__`) также должны иметь строки документации. Пакет модулей может быть документирован в `__init__.py`. Для согласованности, всегда используйте `"""triple double quotes"""` для строк документации. Используйте `r"""raw triple double quotes"""`, если вы будете использовать обратную косую черту в строке документации. Существует две формы строк документации: однострочная и многострочная.

9. Одиночные строки документации предназначены для действительно очевидных случаев. Они должны уместиться на одной строке. Используйте тройные кавычки, даже если документация уместается на одной строке. Потом будет проще её дополнить. Однострочная строка документации не должна быть "подписью" параметров функции / метода (которые могут быть получены с помощью интроспекции). 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. Вставляйте

пустую строку до и после всех строк документации (однострочных или многострочных), которые документируют класс - вообще говоря, методы класса разделены друг от друга одной пустой строкой, а строка документации должна быть смещена от первого метода пустой строкой; для симметрии, поставьте пустую строку между заголовком класса и строкой документации. Строки документации функций и методов, как правило, не имеют этого требования.