

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение
высшего образования**

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра информационных систем и технологий

Отчет по лабораторной работе №11.

Дисциплина: «Основы программной инженерии»

Выполнил:

Студент группы ПИЖ-б-о-22-1,

направление подготовки: 09.03.04

«Программная инженерия»

ФИО: Гуртовой Ярослав Дмитриевич

Проверил:

Воронкин Р. А.

Ставрополь 2022

Тема: Лабораторная работа 2.8 Работа с функциями в языке Python.

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

Выполнение работы:

1. Изучил теоретический материал работы.
2. Создал репозиторий на git.hub.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner * **Repository name ***

KingItProgger / lr-2.8

✔ lr-2.8 is available.

Great repository names are short and memorable. Need inspiration? How about **fictional-pancake** ?

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: **Python**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **MIT License**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set **main** as the default branch. Change the default name in your [settings](#).

You are creating a public repository in your personal account.

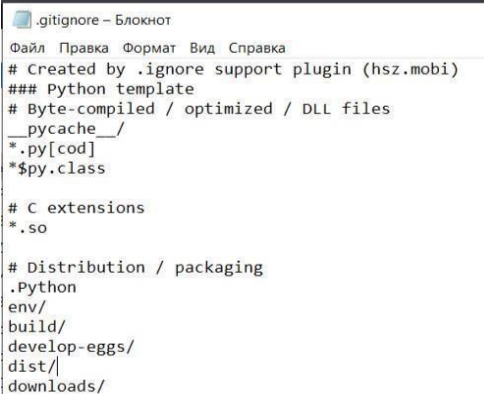
Рисунок 1 – создание репозитория

3. Клонировал репозиторий.

```
PS C:\Users\User\Desktop\учеба\Зсемак\Змії\lr_2.8> git clone http
Cloning into 'lr-2.8'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
PS C:\Users\User\Desktop\учеба\Зсемак\Змії\lr_2.8>
```

Рисунок 2 – клонирование репозитория 4.

4. Дополнить файл gitignore необходимыми правилами.



```
.gitignore – Блокнот
Файл Правка Формат Вид Справка
# Created by .ignore support plugin (hsz.mobi)
### Python template
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
env/
build/
develop-eggs/
dist/
downloads/
```

Рисунок 3 – .gitignore для IDE PyCharm

5. Проработать примеры из методички.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  usage = """
7  def get_worker():
8      """
9      Запросить данные о работнике.
10     """
11     name = input("Фамилия и инициалы? ")
12     post = input("Должность? ")
13     year = int(input("Год поступления? "))
14
15     # Создать словарь.
16     return {
17         'name': name,
18         'post': post,
19         'year': year,
20     }
21
22 2 usages = """
23  def display_workers(staff):
24      """
25      Отобразить список работников.
26      """
27      # Проверить, что список работников не пуст.
28      if staff:
29          # Заголовок таблиц.
30          line = '+--+--+--+--+--+--+'.format(
31              "name" * 4,
32              "-" * 30,
33              "-" * 20,
34              "-" * 8
35          )
36          print(line)
37          print(
38              '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
39                  "name",
40                  "Ф.И.О.",
41                  "Должность",
42                  "Год"
43              )
44          )
45          print(line)
46
47      # Вывести данные о всех сотрудниках.
48      for idx, worker in enumerate(staff, 1):
49          print(
50              '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
51                  "name",
52                  worker.get('name', ''),
53                  worker.get('post', ''),
54                  worker.get('year', 0)
55              )
56          )
57          print(line)
58          print("Список работников пуст.")
59
60 3 usages = """
61  def select_workers(staff, period):
62      """
63      Выбрать работников с заданным стажем.
64      """
65      # Получить текущую дату.
66      today = date.today()
67
68      # Сформировать список работников.
69      result = []
70      for employee in staff:
71          if today.year - employee.get('year', today.year) >= period:
72              result.append(employee)
73
74      # Возвратить список выбранных работников.
75      return result
76
77 4 usages = """
78  def main():
79      """
80      Главная функция программы.
81      """
82      # Список работников.
83      workers = []
84      # Организовать бесконечный цикл запроса команд.
85      while True:
86          # Запросить команду из терминала.
87          command = input(">>> ").lower()
88          # Выполнить действие в соответствии с командой.
89
90          if command == 'exit':
91              break
92
93          elif command == 'add':
94              # Запросить данные о работнике.
95              worker = get_worker()
96              # Добавить словарь в список.
97              workers.append(worker)
98              # Отсортировать список в случае необходимости.
99              if len(workers) > 1:
100                  workers.sort(key=lambda item: item.get('name', ''))
101
102          elif command == 'list':
103              # Отобразить всех работников.
104              display_workers(workers)
105
106          elif command.startswith('select '):
107              # Разбить команду на части для выделения стажа.
108              parts = command.split(' ', maxsplit=1)
109              # Получить требуемый стаж.
110              period = int(parts[1])
111              # Выбрать работников с заданным стажем.
112              selected = select_workers(workers, period)
113              # Отобразить выбранных работников.
114              display_workers(selected)
115
116          elif command == 'help':
117              # Вывести справку о работе с программой.
118              print("Список команд:\n")
119              print("add - добавить работника;")
120              print("list - вывести список работников;")
121              print("select стаж - запросить работников со стажем;")
122              print("help - отобразить справку;")
123              print("exit - завершить работу с программой.")
124
125          else:
126              print(f"Неизвестная команда {command}", file=sys.stderr)
127
128 5 if __name__ == '__main__':
129      main()

```

Рисунок 5 – пример 1

```

C:\Users\User\AppData\Local\Programs\Python\Python311\python.exe "C:/Users/User
>>> add
Фамилия и инициалы? ярослав г
Должность? директор
Год поступления? 2000
>>> add
Фамилия и инициалы? иванов и
Должность? секретарь
Год поступления? 2020
>>> list
+-----+-----+-----+-----+
| № |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | иванов и                  | секретарь           |      2020     |
|  2 | ярослав г                 | директор            |      2000     |
+-----+-----+-----+-----+
Список работников пуст.
>>>

```

Рисунок 6 – пример выполнения примера 1

6. 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".

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

def test():
    number = int(input("Введите целое число: "))
    if number > 0:
        positive()
    elif number < 0:
        negative()

def positive():
    print("Положительное")

def negative():
    print("Отрицательное")

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

Рисунок 7 – выполнения 9 задания

```
C:\Users\User\AppData\Local\Programs\Python\Python311\python.exe "C:/Users/User/Desktop/ОПИ/лр 2.8/task1.py"
Введите целое число: 3
Положительное

Process finished with exit code 0
```

Рисунок 8 – пример выполнения 9 задания

7. В основной ветке программы вызывается функция `cylinder()`, которая вычисляет площадь цилиндра. В теле `cylinder()` определена функция `circle()`, вычисляющая площадь круга по формуле πr^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().

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import math

def cylinder():

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

    r = float(input("Введите радиус цилиндра: "))
    h = float(input("Введите высоту цилиндра: "))
    side_area = 2 * math.pi * r * h
    full_area = side_area + 2 * circle(r)

    choice = input("Хотите получить только площадь боковой поверхности цилиндра? (да/нет): ")
    if choice.lower() == "да":
        print(f"Площадь боковой поверхности цилиндра: {side_area}")
    else:
        print(f"Полная площадь цилиндра: {full_area}")

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

Рисунок 9 – выполнение задания 10

```
C:\Users\User\AppData\Local\Programs\Python\Python311\python.exe "C:/Users/User
Введите радиус цилиндра: 50
Введите высоту цилиндра: 20
Хотите получить только площадь боковой поверхности цилиндра? (да/нет): нет
Полная площадь цилиндра: 21991.14857512855

Process finished with exit code 0
|
```

Рисунок 10 – результат выполнения задания 10

8. 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.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import math

def check():
    n = int(input())
    s = n

    while n != 0:
        n = int(input())
        s += n

    print(s)

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

Рисунок 11 – выполнение 12 задания

```
C:\Users\User\AppData\Local\Programs
1
2
3
4
5
0
15

Process finished with exit code 0
```

Рисунок 12 – результат выполнения 12 задания

9. Напишите программу, в которой определены четыре Функции.


```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def get_input():
    n = input("Введите значение: ")

    return n

def test_input(value):
    try:
        int(value)
        return True
    except ValueError:
        return False

def str_to_int(n):
    s = int(n)
    return s

def print_int(s):
    print(s)

if __name__ == '__main__':
    n = get_input()

    if test_input(n):
        s = str_to_int(n)
        print_int(s)
    else:
        print("Введенное значение не является числом!!!")
```

Рисунок 13 – выполнение задания 14

Рисунок 14 – результат выполнения задания 14

10. Решить индивидуальное задание лабораторной работы 2.6, оформив каждую команду в виде отдельной функции.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

def help1():
    """
    Функция для вывода списка команд
    """
    # Вывести справку о работе с программой.
    print("Список команд:\n")
    print("add - добавить рейс;")
    print("list - вывести список рейсов;")
    print("select <тип> - вывод на экран пунктов назначения и номеров рейсов для данного типа самолёта")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")

def add1():
    """
    Функция для добавления информации о новых рейсах
    """
    # Запросить данные о работнике.
    name = input("Название пункта назначения рейса? ")
    number = int(input("Номер рейса? "))
    tip = input("Тип самолета? ")
    # Создать словарь.
    i = {
        'name': name,
        'number': number,
        'tip': tip,
    }

    return i

def error1():
    """
    Функция для неопознанных команд
    """
    print(f"Неизвестная команда {command}")

def list1(aircrafts):
    """
    Функция для вывода списка добавленных рейсов
    """
    # Заголовок таблицы.
    line = '+-{}-+-{}-+-{}-+-+'.format(
        '-' * 4,
        '-' * 30,
        '-' * 20,
        '-' * 8
    )
    print(line)
```

```

print(line)

# Вывести данные о всех сотрудниках.
for idx, i in enumerate(aircrafts, 1):
    print(
        '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
            idx,
            i.get('name', ''),
            i.get('number', ''),
            i.get('tip', '')
        )
    )
print(line)

def select(command, aircrafts):
    """
    Функция для получения номера рейса и пункта назначения по заданному типу самолёта.
    """
    # Разбить команду на части для выделения номера года.
    parts = input("Введите значение: ")
    # Проверить сведения работников из списка.
    count = 0

    for i in aircrafts:
        for k, v in i.items():
            if v == parts:
                print("Пункт назначения - ", i["name"])
                print("Номер рейса - ", i["number"])
                count += 1

    # Если счетчик равен 0, то работники не найдены.

    if count == 0:
        print("Рейс с заданным типом самолёта не найден.")

def main():
    """
    Главная функция программы.
    """
    print("Список команд:\n")
    print("add - добавить рейс;")
    print("list - вывести список рейсов;")
    print("select <тип> - вывод на экран пунктов назначения и номеров рейсов для данного типа самолёта")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")
    # Список работников.
    aircrafts = []
    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()
        # Выполнить действие в соответствии с командой.

```

```
match command:
    case 'exit':
        break

    case 'add':
        # Добавить словарь в список.
        i = add1()
        aircrafts.append(i)
        # Отсортировать список в случае необходимости.
        if len(aircrafts) > 1:
            aircrafts.sort(key=lambda item: item.get('name', ''))

    case 'list':
        list(aircrafts)

    case 'select':
        select(command, aircrafts)

    case 'help':
        help1()

    case _:
        error1()

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

Рисунок 15 – выполнение индивидуального задания

```

>>> add
Название пункта назначения рейса? dubai
Номер рейса? 2
Тип самолета? 777
>>> add
Название пункта назначения рейса? berlin
Номер рейса? 20
Тип самолета? 777
>>> list
+-----+-----+-----+-----+
| 1 | berlin | 20 | 777 |
| 2 | dubai | 2 | 777 |
+-----+-----+-----+-----+
>>> |

```

Рисунок 16 – результат выполнения индивидуального задания
Вывод: приобрел 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?

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. 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. Каково назначение операторов def и return?

Оператор def в Python используется для определения функции. Он начинает заголовок функции и может принимать ноль или более аргументов, которые могут использоваться в теле функции. Оператор return используется для возврата результата выполнения функции. Он может быть необязательным, так как функция может ничего не возвращать. Оператор return не только возвращает значение, но и производит выход из функции. Поэтому он должен определяться после остальных инструкций. Если функция не возвращает никакого значения, после оператора return не ставится никакого возвращаемого значения.

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

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.

4. Как вернуть несколько значений из функции Python?

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

5. Какие существуют способы передачи значений в функцию?

```
figure4 = cylinder(r=2, h=10)
```

```
def cylinder(h, r=1):
```

```
figure4 = cylinder(i, h)(передаются значения глобальных переменных)
```

6. Как задать значение аргументов функции по умолчанию?

```
def cylinder(h, r=1):
```

7. Каково назначение lambda-выражений в языке Python?

Lambda-выражения в Python, также известные как “анонимные функции”, используются для создания небольших функций без необходимости использования ключевого слова def. Они представляют собой компактный способ определения функции.

8. Как осуществляется документирование кода согласно PEP257?

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. 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

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

Однострочные строки документации используются для краткого описания функции, метода, класса или модуля, что делает и какие аргументы принимает. Они заключаются в тройные кавычки и пишутся в императивной форме. Многострочные строки документации используются для более подробного

описания функции, метода, класса или модуля, включая их параметры, типы, возвращаемые значения, исключения, примеры и другие детали. Они также заключаются в тройные кавычки, но имеют определенный формат и стиль, в зависимости от выбранной конвенции.

