

**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 № 2.8 Работа с функциями в языке Python**

Выполнил:

Шальнев Владимир Сергеевич,  
2 курс, группа ПИЖ-б-о-20-1,

Проверил:

Доцент кафедры  
прикладной математики и  
компьютерной безопасности,  
Воронкин Р.А.

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

Ставрополь, 2021 г.

## ВЫПОЛНЕНИЕ:

```
C:\Users\Serj\PycharmProjects>git clone https://github.com/HAXF13D/-laboratory-11
Cloning into '-laboratory-11'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 11 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (11/11), done.
Resolving deltas: 100% (2/2), done.
```

Клонирование репозитория

```
C:\Users\Serj\PycharmProjects\laboratory-11>git checkout develop
Switched to a new branch 'develop'
Branch 'develop' set up to track remote branch 'develop' from 'origin'.
```

Переход на ветку develop

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

import sys
from datetime import date

def get_worker():
    """
    Запросить данные о работнике.
    """
    name = input("Фамилия и инициалы? ")
    post = input("Должность? ")
    year = int(input("Год поступления? "))
    # Создать словарь.
    return {
        'name': name,
        'post': post,
        'year': year,
    }

def display_workers(staff):
    """
    Отобразить список работников.
    """
    # Проверить, что список работников не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "No",
                "Ф.И.О.",
                "Должность",

```

```

        "Год"
    )
)
print(line)
# Вывести данные о всех сотрудниках.
for idx, worker in enumerate(staff, 1):
    print(
        '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
            idx,
            worker.get('name', ''),
            worker.get('post', ''),
            worker.get('year', 0)
        )
    )
print(line)
else:
    print("Список работников пуст.")

def select_workers(staff, period):
    """
    Выбрать работников с заданным стажем.
    """
    # Получить текущую дату.
    today = date.today()
    # Сформировать список работников.
    result = []
    for employee in staff:
        if today.year - employee.get('year', today.year) >= period:
            result.append(employee)
    # Возвратить список выбранных работников.
    return result

def main():
    """
    Главная функция программы.
    """
    # Список работников.
    workers = []
    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()
        # Выполнить действие в соответствие с командой.
        if command == 'exit':
            break
        elif command == 'add':
            # Запросить данные о работнике.
            worker = get_worker()
            # Добавить словарь в список.
            workers.append(worker)
            # Отсортировать список в случае необходимости.
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get('name', ''))
        elif command == 'list':
            # Отобразить всех работников.
            display_workers(workers)
        elif command.startswith('select '):
            # Разбить команду на части для выделения стажа.
            parts = command.split(' ', maxsplit=1)
            # Получить требуемый стаж.
            period = int(parts[1])
            # Выбрать работников с заданным стажем.

```

```

        selected = select_workers(workers, period)
        # Отобразить выбранных работников.
        display_workers(selected)
    elif command == 'help':
        # Вывести справку о работе с программой.
        print("Список команд:\n")
        print("add - добавить работника;")
        print("list - вывести список работников;")
        print("select <стаж> - запросить работников со стажем;")
        print("help - отобразить справку;")
        print("exit - завершить работу с программой.")
    else:
        print(f"Неизвестная команда {command}", file=sys.stderr)

if __name__ == '__main__':
    main()

```

## Пример 1

```

>>> add
Фамилия и инициалы? Проеврека С.С.
Должность? Девелопер
Год поступления? 2002
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |   Год   |
+-----+-----+-----+-----+
|  1 | Проеврека С.С.          | Девелопер          |  2002  |
+-----+-----+-----+-----+
>>> select 10
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |   Год   |
+-----+-----+-----+-----+
|  1 | Проеврека С.С.          | Девелопер          |  2002  |
+-----+-----+-----+-----+
>>> select 40
Список работников пуст.
>>> exit

Process finished with exit code 0

```

## Значение №1

```

>>> er
>>> Неизвестная команда er
edit
Неизвестная команда edit
>>> add
Фамилия и инициалы? |

```

## Значение №2

```
C:\Users\Serj\PycharmProjects\-laboratory-11>git add examples/

C:\Users\Serj\PycharmProjects\-laboratory-11>git commit -m "Add examples"
[develop b0ecf8e] Add examples
 1 file changed, 122 insertions(+)
 create mode 100644 examples/ex.py
```

Коммит изменений

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

def positive():
    print("Число положительное")

def negative():
    print("Число отрицательное")

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

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

Задача 1

```
Введите целое число: 10
Число положительное
```

Значение №1

```
Введите целое число: -5
Число отрицательное
```

Значение №2

```
C:\Users\Serj\PycharmProjects\-laboratory-11>git add task/task1.py

C:\Users\Serj\PycharmProjects\-laboratory-11>git commit -m "First task"
[develop ebda23a] First task
 1 file changed, 24 insertions(+)
 create mode 100644 task/task1.py
```

Коммит изменений

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

from math import pi

def cylinder():

    def circle(rad):
        return pi * rad * rad

    r = int(input("Введите радиус: "))
    h = int(input("Введите высоту: "))
    choose = input("Площадь боковой поверхности цилиндра - a\n"
                  "Полная площадь цилиндра - b\n"
                  "a/b: ")
    if choose == 'a':
        print(f"Площадь боковой поверхности цилиндра = {2 * pi * r * h}")
    else:
        print(f"Полная площадь цилиндра = {2 * pi * r * h + 2 * circle(r)}")

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

## Задача 2

```
Введите радиус: 2
Введите высоту: 4
Площадь боковой поверхности цилиндра - a
Полная площадь цилиндра - b
a/b: b
Полная площадь цилиндра = 75.39822368615503
```

## Значение №1

```
Введите радиус: 2
Введите высоту: 4
Площадь боковой поверхности цилиндра - a
Полная площадь цилиндра - b
a/b: a
Площадь боковой поверхности цилиндра = 50.26548245743669
```

## Значение №2

```
C:\Users\Serj\PycharmProjects\laboratory-11>git add task/task2.py
C:\Users\Serj\PycharmProjects\laboratory-11>git commit -m "Second task"
[develop 981d23a] Second task
1 file changed, 24 insertions(+)
create mode 100644 task/task2.py
```

## Коммит изменений

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

def multi():
    number = int(input("Введите число: "))
    result = 1
    if number == 0:
        return None
    while number != 0:
        result *= number
        number = int(input("Введите число: "))
    return result

if __name__ == '__main__':
    print(f"Вызов функции и её результата = {multi()}")
```

### Задача 3

```
Введите число: 1
Введите число: 2
Введите число: 3
Введите число: 4
Введите число: 0
Вызов функции и её результата = 24
```

### Значение №1

```
Введите число: 2
Введите число: 2
Введите число: 2
Введите число: 0
Вызов функции и её результата = 8
```

### Значение №2

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

def get_input():
    return input()

def test_input(string):
    return string.isdigit()

def str_to_int(string):
    return int(string)
```

```
def print_int(integer):
    print(integer)

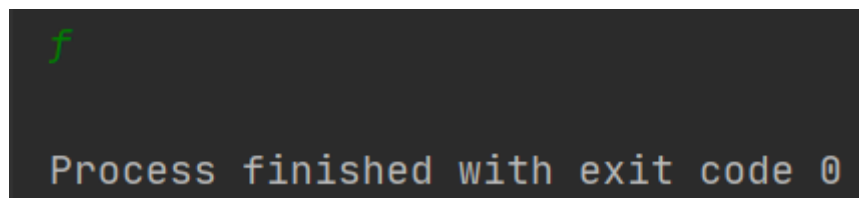
def main():
    data = get_input()
    if test_input(data):
        print_int(str_to_int(data))

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

#### Задача 4



Значение №1



Значение №2

```
C:\Users\Serj\PycharmProjects\laboratory-11>git add task/task4.py

C:\Users\Serj\PycharmProjects\laboratory-11>git commit -m "Fourth task"
[develop 0b6a6d3] Fourth task
 1 file changed, 28 insertions(+)
 create mode 100644 task/task4.py
```

Коммит изменений

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# variant 6

import sys
from datetime import datetime

def main():
    trains = []
    while True:
        command = get_command()
        if command == 'exit':
            break

        elif command == 'add':
            trains.append(add())
            if len(trains) > 1:
```



```

        trains.sort(key=lambda item: item.get('destination', ''))

    elif command == 'list':
        print_list(trains)

    elif command.startswith('select '):
        select(command, trains)

    elif command == 'help':
        print_help()
    else:
        print(f"Неизвестная команда {command}", file=sys.stderr)

def get_command():
    return input(">>> ").lower()

def add():
    destination = input("Название пункта назначения? ")
    number = int(input("Номер поезда? "))
    time = input("Время отправления ЧЧ:ММ? ")
    time = datetime.strptime(time, '%H:%M')
    train = {
        'destination': destination,
        'number': number,
        'time': time,
    }
    return train

def print_list(trains):
    line = '+-{}-+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 28,
        '-' * 14,
        '-' * 19
    )
    print(line)
    print(
        '| {:^4} | {:^28} | {:^14} | {:^19} |'.format(
            "No",
            "Название пункта назначения",
            "Номер поезда",
            "Время отправления"
        )
    )
    print(line)
    for idx, train in enumerate(trains, 1):
        print(
            '| {:>4} | {:<28} | {:<14} | {:>19} |'.format(
                idx,
                train.get('destination', ''),
                train.get('number', ''),
                train.get('time', 0).strftime("%H:%M")
            )
        )
    print(line)

def print_help():
    print("Список команд:\n")
    print("add - добавить отправление;")
    print("list - вывести список отправлений;")

```

```

print("select <ЧЧ:ММ> - вывод на экран информации о "
      "поездах, отправляющихся после этого времени;")
print("help - отобразить справку;")
print("exit - завершить работу с программой.")

def select(command, trains):
    count = 0
    parts = command.split(' ', maxsplit=1)
    time = datetime.strptime(parts[1], '%H:%M')
    for train in trains:
        if train.get("time") > time:
            count += 1
            print(
                '{:>4}: {} {}'.format(
                    count,
                    train.get('destination', ''),
                    train.get("number")
                )
            )
    if count == 0:
        print("Отправлений позже этого времени нет.")

if __name__ == '__main__':
    main()

```

## Индивидуальное задание 1

```

>>> add
Название пункта назначения? Butovo
Номер поезда? 127
Время отправления ЧЧ:ММ? 12:30
>>> list

```

No	Название пункта назначения	Номер поезда	Время отправления
1	Butovo	127	12:30

## Значение №1

```

add
Название пункта назначения? Moscow
Номер поезда? 126
Время отправления ЧЧ:ММ? 14:20
>>> help
Список команд:

add - добавить отправление;
list - вывести список отправлений;
select <ЧЧ:ММ> - вывод на экран информации о поездах, отправляющихся после этого времени;
help - отобразить справку;
exit - завершить работу с программой.
>>> select 13:20
1: Moscow 126
>>> select 12:00
1: Butovo 127
2: Moscow 126
>>> |

```

## Значение №2

### ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ:

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

Функция представляет собой обособленный участок кода, который можно вызывать, обратившись к нему по имени, которым он был назван. При вызове происходит выполнение команд тела функции.

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

В языке программирования Python функции определяются с помощью оператора def.

Выход из функции и передача данных в то место, откуда она была вызвана, выполняется оператором return.

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

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.

#### 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?

С помощью так называемых параметров, которые указываются в скобках в заголовке функции. Количество параметров может быть любым.

Однако в Python у функций бывают параметры, которым уже присвоено значение по умолчанию. В таком случае, при вызове можно не передавать соответствующие этим параметрам аргументы. Хотя можно и передать.

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

```
def do_smth(a, b=2) # Значение по умолчанию b = 2
```

**7. Каково назначение lambda-выражений в языке Python?**

интересный синтаксис, позволяющий 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.

**8. Как осуществляется документирование кода согласно PEP257?**

- Тройные кавычки используются даже если строка помещается на одной линии. Это облегчает последующее расширение документации.
- Закрывающие кавычки находятся на той же строке, что и открывающие. Для однострочных `docstring` это выглядит лучше.
- Ни до, ни после документации не пропускаются строки. Код пишется сразу же на следующей линии
- Документационная строка — это «фраза», заканчивающаяся точкой. Она описывает эффект функции или метода в командном тоне
- Однострочная документация НЕ должна быть простой «подписью», повторяющей параметры функции/метода

Многострочные:

- Многострочные документации состоят из сводной строки (`summary line`) имеющей такую же структуру, как и однострочный `docstring`, после которой следует пустая линия, а затем более сложное описание.
- Оставляйте пустую строку после всех документаций (однострочных или многострочных), которые используются в классе;
- Документация скрипта (автономной программы) представляет из себя сообщение «о правильном использовании» и возможно будет напечатано, когда скрипт вызовется с неверными или отсутствующими аргументами
- Документация модуля должна обычно содержать список классов, исключений и функций (и любых других важных объектов), которые экспортируются при помощи библиотеки, а также однострочное пояснение для каждого из них.
- Документация функции или метода должна описывать их поведение, аргументы, возвращаемые значения, побочные

эффекты, возникающие исключения и ограничения на то, когда они могут быть вызваны.

- Документация класса должна обобщать его поведение и перечислять открытые методы, а также переменные экземпляра.
- Если класс является потомком и его поведение в основном наследуется от основного класса, в его документации необходимо упомянуть об этом и описать возможные различия.

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

Одиночные строки документации предназначены для действительно очевидных случаев. Они должны уместиться на одной строке.

Многострочные строки документации состоят из однострочной строки документации с последующей пустой строкой, а затем более подробным описанием.