

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 ЛАБОРАТОРНОЙ РАБОТЕ №11
дисциплины «Программирование на Python»

Выполнил:
Середа Кирилл Витальевич
1 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Работа с функциями в языке Python

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

Ход выполнения:

1) Пример 1 – Программа из работы 2.6 оформленная в виде функций.

```
>>> 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? Анатолий Евгений
Должность? стажер
Год поступления? 2023
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|   1 | Анатолий Евгений        | стажер              |      2023      |
+-----+-----+-----+-----+
>>> |
```

Рисунок 1 – результат выполнения программы

2) Задание 1 – составить программу с помощью функций, которая определяет положительность введенного числа

```
Введите целое число: -2
Отрицательное
```

Рисунок 2 – Результат выполнения программы

Код программы:

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

def negative():
    print("Отрицательное")

def positive():
    print("Положительное")

def test():
    num = int(input("Введите целое число: "))
    if num > 0:
```

```

        positive()
    elif num < 0:
        negative()

if __name__ == '__main__':
    test()

```

3) Задание 2 – написать программу вычисляющую площадь цилиндра с помощью функций

```

Введите радиус цилиндра: 12
Введите высоту цилиндра: 23
Хотите получить площадь боковой поверхности (s) или полную площадь (f)? f
Полная площадь цилиндра: 2638.9378290154264

```

Рисунок 3 – Результат выполнения программы

Код программы:

```

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

import math

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

def cylinder():
    radius = float(input("Введите радиус цилиндра: "))
    height = float(input("Введите высоту цилиндра: "))
    option = input("Хотите получить площадь боковой поверхности (s) или полную площадь (f)? ")

    if option == "s":
        lateral_area = 2 * math.pi * radius * height
        print(f"Площадь боковой поверхности цилиндра: {lateral_area}")
    elif option == "f":
        lateral_area = 2 * math.pi * radius * height
        total_area = lateral_area + 2 * circle(radius)
        print(f"Полная площадь цилиндра: {total_area}")
    else:
        print("Некорректный вариант. Выберите 'side' или 'full'.")

if __name__ == '__main__':
    cylinder()

```

4) Задание 3 – написать программу с использованием функций, которая перемножает введенные числа с клавиатуры пока не будет введен 0

```

Введите число (введите 0 для завершения): 12
Введите число (введите 0 для завершения): 2
Введите число (введите 0 для завершения): 3
Введите число (введите 0 для завершения): 0
Произведение введенных чисел: 72.0

```

Рисунок 4 – Результат работы программы

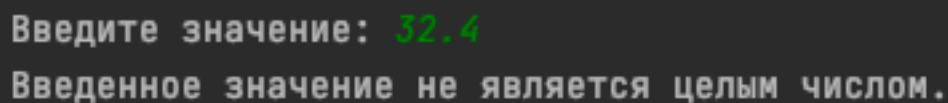
Код программы:

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

def multiply_until_zero():
    result = 1
    while True:
        number = float(input("Введите число (введите 0 для завершения): "))
        if number == 0:
            break
        result *= number
    return result

if __name__ == '__main__':
    product = multiply_until_zero()
    print(f"Произведение введенных чисел: {product}")
```

5) Задание 4 - Программа сначала вызывает функцию `get_input()` для ввода значения с клавиатуры, затем использует функцию `test_input()` для проверки, является ли введенное значение целым числом. Если значение проходит проверку, программа преобразует его в целочисленный тип с помощью функции `str_to_int()` и выводит результат с использованием функции `print_int()`. Если значение не является целым числом, программа выведет сообщение об ошибке.



Введите значение: 32.4
Введенное значение не является целым числом.

Рисунок 5 – Результат выполнения программы

Код программы:

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

def get_input():
    user_input = input("Введите значение: ")
    return user_input

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

def str_to_int(value):
    return int(value)

def print_int(value):
    print(value)

if __name__ == '__main__':
    input_str = get_input()
```

```

if test_input(input_str):
    integer_value = str_to_int(input_str)
    print_int(integer_value)
else:
    print("Введенное значение не является целым числом.")

```

б) Индивидуальное задание – переписал задание из 2.6 с помощью функций

```

>>> help
Список команд:

add - добавить студента;
list - вывести список студентов;
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Фамилия и инициалы? 12с
Номер группы? 2
Успеваемость (через пробел)? 4
>>> list
+-----+-----+-----+
|          Ф.И.О.          | Номер группы |   Успеваемость   |
+-----+-----+-----+
+-----+-----+-----+
>>> add
Фамилия и инициалы? 12с
Номер группы? 2
Успеваемость (через пробел)? 4.5
>>> list
+-----+-----+-----+
|          Ф.И.О.          | Номер группы |   Успеваемость   |
+-----+-----+-----+
| 12с                      | 2           | 4.5              |
+-----+-----+-----+
>>> exit

```

Рисунок 6 – Результат выполнения программы

Код программы:

```

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

def add_student(students):
    full_name = input("Фамилия и инициалы? ")
    group_number = input("Номер группы? ")
    grades_str = input("Успеваемость (через пробел)? ")

    grades = [float(grade) for grade in grades_str.split()]

    student = {
        'full_name': full_name,
        'group_number': group_number,
        'grades': grades,
    }

```

```

students.append(student)
students.sort(key=lambda item: item.get('group_number', ''))

def list_students(students):
    line = '+-{}-+-{}-+-{}-+'.format(
        '-' * 30,
        '-' * 15,
        '-' * 20
    )
    print(line)
    print(
        '| {:^30} | {:^15} | {:^20} |'.format(
            "Ф.И.О.",
            "Номер группы",
            "Успеваемость"
        )
    )
    print(line)

    for student in students:
        average_grade = sum(student.get('grades', 0)) /
len(student.get('grades', 1))
        if average_grade > 4.0:
            print(
                '| {:<30} | {:<15} | {:<20} |'.format(
                    student.get('full_name', ''),
                    student.get('group_number', ''),
                    ', '.join(map(str, student.get('grades', [])))
                )
            )
    print(line)

def help_command():
    print("Список команд:\n")
    print("add - добавить студента;")
    print("list - вывести список студентов;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")

def main():
    students = []

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

        if command == 'exit':
            break
        elif command == 'add':
            add_student(students)
        elif command == 'list':
            list_students(students)
        elif command == 'help':
            help_command()
        else:
            print(f"Неизвестная команда {command}")

if __name__ == '__main__':
    main()

```

Ответы на 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. 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.

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

Ключевое слово `def` сообщает интерпретатору, что перед ним определение функции. За `def` следует имя функции. Оно может быть любым, также как и всякий идентификатор, например, переменная. В программировании весьма желательно давать всему осмысленные имена. 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, встречает `return`, то он "забирает" значение, указанное после этой команды, и "уходит" из функции.

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

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.

4. Как вернуть несколько значений из функции Python?

В Питоне позволительно возвращать из функции несколько объектов, перечислив их через запятую после команды `return`. Фокус здесь в том, что перечисление значений через запятую (например, 10, 15, 19) создает объект типа `tuple`.

5. Какие существуют способы передачи значений в функцию?

В программировании функции могут не только возвращать данные, но также принимать их, что реализуется с помощью так называемых параметров, которые указываются в скобках в заголовке функции. Количество параметров может быть любым. Параметры представляют собой локальные переменные, которым присваиваются значения в момент вызова функции. Конкретные значения, которые передаются в функцию при ее вызове, будем называть аргументами. Следует иметь в виду, что встречается иная терминология. Например, формальные параметры и фактические параметры. В Python же обычно все называют аргументами. Когда интерпретатор переходит к функции, чтобы начать ее исполнение, он присваивает переменным-параметрам переданные в функцию значения-аргументы. Изменение значений переменных в теле функции никак не скажется на значениях, переданных в нее. Они останутся прежними. В Python такое поведение характерно для неизменяемых типов данных, к которым относятся, например, числа и строки. Говорят, что в функцию данные передаются по значению. Существуют изменяемые типы данных. Для Питона, это, например, списки и словари. В этом случае данные передаются по ссылке. В функцию передается ссылка на них, а не сами данные. И эта

ссылка связывается с локальной переменной. Изменения таких данных через локальную переменную обнаруживаются при обращении к ним через глобальную. Это есть следствие того, что несколько переменных ссылаются на одни и те же данные, на одну и ту же область памяти.

6. Как задать значение аргументов функции по умолчанию?

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. Тогда значение по умолчанию заменится на переданное. Например, в следующем определении функции параметр со значением по умолчанию будет `r`: `def cylinder(h, r=1)`.

7. Каково назначение lambda-выражений в языке Python?

Python поддерживает интересный синтаксис, позволяющий определять небольшие однострочные функции на лету. Позаимствованные из Lisp, так называемые lambda-функции могут быть использованы везде, где требуется функция. `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?

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.

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 нужно использовать `"""triple double quotes"""` для строк документации. Можно использовать `r"""raw triple double quotes"""`, если будет присутствовать обратная косая черта в строке документации. Существует две формы строк документации: однострочная и многострочная

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

Одиночные строки документации предназначены для действительно очевидных случаев. Они должны уместиться на одной строке. Однострочная строка документации не должна быть "подписью" параметров функции / метода (которые могут быть получены с помощью интроспекции). Этот тип строк документации подходит только для C функций (таких, как встроенные модули), где интроспекция не представляется возможной. Тем не менее, возвращаемое значение не может быть определено путем 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.

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