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

ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчёт о лабораторной работе №11 по дисциплине основы программной инженерии

Выполнила:
Нестеренко Тамара Антоновна,
2 курс, группа ПИЖ-б-о-20-1,
Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г.

ВЫПОЛНЕНИЕ

1. Практическая часть

```
i = 0
while i < 3:
    a = int(input())
    b = int(input())
    i += 1
```

Рисунок 1 – Пример использования циклоа

```
print("Сколько бананов и ананасов для обязьян?")
a = int(input())
b = int(input())
print("всего", a+b, "шт.")

print("Сколько жуков и червей для ежей?")
a = int(input())
b = int(input())
print("всего", a+b, "шт.")

print("Сколько рыб и моллюсков для выдр?")
a = int(input())
b = int(input())
print("всего", a+b, "шт.")
```

Рисунок 2 – Пример длинного кода без функции

```
Сколько бананов и ананасов для обязьян?
15
5
всего 20 шт.
Сколько жуков и червей для ежей?
50
12
всего 62 шт.
Сколько рыб и моллюсков для выдр?
16
8
всего 24 шт.
```

Рисунок 3 – Пример исполнения программы

```
def countFood():
    a = int(input())
    b = int(input())
    print("Всего", a+b, "шт.")
```

Рисунок 4 – Пример использования оператора def

```
def countFood():
    a = int(input())
    b = int(input())
    print("Всего", a + b, "шт.")


print("Сколько бананов и ананасов для обязьян?")
countFood()

print("Сколько жуков и червей для ежей?")
countFood()

print("Сколько рыб и моллюсков для выдр?")
countFood()
```

Рисунок 5 – Пример вызова функции

```
import math
import sys


figure = input("1-прямоугольник, 2-треугольник, 3-круг: ")

if figure == '1':
    a = float(input("Ширина: "))
    b = float(input("Высота: "))
    print(f"Площадь: {a * b}")
elif figure == '2':
    a = float(input("Основание: "))
    h = float(input("Высота: "))
    print(f"Площадь: {0.5 * a * h}")
elif figure == '3':
    r = float(input("Радиус: "))
    print(f"Площадь: {math.pi * r**2}")
else:
    print("Ошибка ввода", file=sys.stderr)
```

Рисунок 6 – Пример не структурированной программы

```python
import math
import sys

figure = input("1-прямоугольник, 2-треугольник, 3-круг: ")


def rectangle():
    a = float(input("Ширина: "))
    b = float(input("Высота: "))
    print(f"Площадь: {a * b}")


def triangle():
    a = float(input("Основание: "))
    h = float(input("Высота: "))
    print(f"Площадь: {0.5 * a * h}")


def circle():
    r = float(input("Радиус: "))
    print(f"Площадь: {math.pi * r ** 2}")


if figure == '1':
    rectangle()
elif figure == '2':
    triangle()
elif figure == '3':
    circle()
else:
    print("Ошибка ввода", file=sys.stderr)
```

Рисунок 7 – Пример структурированной программы с функциями

```
result = 0


def rectangle():
    a = float(input("Ширина: "))
    b = float(input("Высота: "))
    result = a * b


def triangle():
    a = float(input("Основание: "))
    h = float(input("Высота: "))
    result = 0.5 * a * h


figure = input("1-прямоугольник, 2-треугольник: ")
if figure == '1':
    rectangle()
elif figure == '2':
    triangle()

print("Площадь: %.2f" % result)
```

Рисунок 8 – Пример неправильного использования глобальных и локальных переменных

```python
result = 0


def rectangle():
    a = float(input("Ширина: "))
    b = float(input("Высота: "))
    global result
    result = a * b


def triangle():
    a = float(input("Основание: "))
    h = float(input("Высота: "))
    global result
    result = 0.5 * a * h


figure = input("1-прямоугольник, 2-треугольник: ")
if figure == '1':
    rectangle()
elif figure == '2':
    triangle()

print("Площадь: %.2f" % result)
```

Рисунок 9 – Пример преобразования глобальной переменной в локальную

```python
import math


def cylinder():
    r = float(input())
    h = float(input())
    side = 2 * math.pi * r * h
    circle = math.pi * r ** 2
    full = side + 2 * circle
    return full


square = cylinder()
print(square)
```

Рисунок 10 – Пример возврата значений из функции

```python
import math


def cylinder():
    try:
        r = float(input())
        h = float(input())
    except valueError:
        return

    side = 2 * math.pi * r * h
    circle = math.pi * r ** 2
    full = side + 2 * circle
    return full


print(cylinder())
```

Рисунок 11 – Пример работы except обработчика исключений

```python
import math


def cylinder():
    r = float(input())
    h = float(input())
    side = 2 * math.pi * r * h
    circle = math.pi * r ** 2
    full = side + 2 * circle
    return side, full


scyl, fcyl = cylinder()
print(f"Площадь боковой поверхности {scyl}")
print(f"Полная площадь {fcyl}")
```

Рисунок 12 – Пример возврата нескольких значений

```python
import math


def cylinder(h, r=1):
    side = 2 * math.pi * r * h
    circle = math.pi * r ** 2
    full = side + 2 * circle
    return full


figure1 = cylinder(4, 3)
figure2 = cylinder(5)
print(figure1)
print(figure2)
```

Рисунок 13 – Пример работы с произвольным количеством аргументов

```
def one_or_many(*a):
    print(a)


one_or_many(1)
one_or_many('1', 1, 2, 'abc')
one_or_many()
```

Рисунок 14 – Пример передачи аргумента в функцию

```
def func(x, y):
    return x ** 2 + y ** 2


func = lambda x, y: x ** 2 + y ** 2
```

Рисунок 15 – Пример работы конструкции lambda

```
foo = [2, 18, 9, 22, 17, 24, 8, 12, 27]

print(list(filter(lambda x: x % 3 == 0, foo)))
print((list(map(lambda x: x * 2 + 10, foo))))
```

Рисунок 16 – Пример хорошего применения lambda со встроенными функциями – map, filter

```
def kos_root():
    """Return the pathname of the KOS root directory."""
    global _kos_root
    if _kos_root: return _kos_root
```

Рисунок 17 – Пример одиночной строки документации

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

import ...


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(
```

```
    line = '+-{}-+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 30,
        '-' * 20,
        '-' * 8
    )
    print(line)
    print(
        '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
            "№",
            "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', '')
            )
        )
    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 emloyee in staff:
        if today.year - emloyee.get('year', today.year) >= period:
            result.append(emloyee)

    # 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.
```

```
    # 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("")


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

Рисунок 18 – Пример № 1

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

import sys


def test():
    a = int(input("Введите число: "))
    if 0 < a:
        positive(a)
    elif a == 0:
        print("Это не положительное и не отрицательное число")
    else:
        negative(a)


def positive(a):
    print("Это положительное число")


def negative(a):
    print("Это отрицательное число")


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

Рисунок 19 – Пример решения задания №1

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

import ...


def cylinder():
    r = float(input("Введите радиус: "))
    h = float(input("Введите высоту: "))

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

    numbers = input("1 - площадь боковой поверхности, 2 - полная площадь: ")
    if numbers == '1':
        print(2 * math.pi * r * h)
    elif numbers == '2':
        print(circle()*2)
    else:
        print("Ошибка")


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

Рисунок 20 – Пример решения задания №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()
```

Рисунок 21 – Пример решения задания №3

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

import sys
from datetime import datetime


def get_worker():
    surname = input("Фамилия: ")
    name = input("Имя: ")
    zodiac = input("Знак зодиака: ")
    date = input("Дата: ")

    return {
        'surname': surname,
        'name': name,
        'zodiac': zodiac,
        'date': datetime.strptime(date, "%Y-%m-%d")
    }


def display_workers(staff):
    if staff:

        line = '+-{}-+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 15,
            '-' * 15
```

```
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^15} | {:^15} |'.format(
                "№",
                "Фамилия",
                "Имя",
                "Знак зодиака",
                "Дата рождения"
            )
        )
        print(line)

        # Вывести данные о всех сотрудниках.
        for idx, worker in enumerate(staff, 1):
            print(
                '| {:^4} | {:^30} | {:^20} | {:^15} | {:^15} |'.format(
                    idx,
                    worker.get('surname', ''),
                    worker.get('name', ''),
                    worker.get('zodiac', ''),
                    str(worker.get('date', '').date())
                )
            )
        print(line)

    else:
        print("Список работников пуст.")
```

```python
def select_workers(staff):
    month1 = int(input("Введите месяц: "))
    result = []
    for worker in staff:
        if worker.get('date', '').month == month1:
            result.append(worker)
    return result


def main():
    workers = []

    while True:

        command = input(">>> ").lower()

        if command == 'exit':
            break

        elif command == 'add':
            worker = get_worker()

            workers.append(worker)
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get('name', ''))

        elif command == 'list':
            display_workers(workers)

        elif command.startswith('select'):
```

```
            selected = select_workers(workers)
            display_workers(selected)

        elif command == 'help':
            print("Список команд:\n")
            print("add - добавить запись;")
            print("list - вывести список;")
            print("select - список родившихся в один месяц;")
            print("help - отобразить справку;")
            print("exit - завершить работу с программой.")

        else:
            print("")


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

Рисунок 22 – Пример решения индивидуального задания. Вариант №16

2. Вопросы для защиты работы

1. **Каково назначение функций в языке программирования Python?**
   Функция – это средство (способ) группирования фрагментов программного кода таким образом, что этот программный код может вызваться многократно с помощью использования имени функции.
   Использование функций в программах на Python даёт следующие взаимосвязанные преимущества:

- избежание повторения одинаковых фрагментов кода в разных частях программы;
- уменьшение избыточности исходного кода программы. Как следствие, уменьшение логических ошибок программирования;
- улучшенное восприятие исходного кода программы в случаях, где вместо блоков многочисленных инструкций (операторов) вызываются имена готовых протестированных функций. Это, в свою очередь, также уменьшает количество ошибок;

- упрощение внесения изменений в повторяемых блоках кода, организованных в виде функций. Достаточно внести изменения только в тело функции, тогда во всех вызовах данной функции эти изменения будут учтены;
- с помощью функций удобно разбивать сложную систему на более простые части. Значит, функции – удобный способ структурирования программы;
- уменьшение трудозатрат на программирование, а, значит, повышение производительности работы программиста.

2. **Каково назначение операторов def и return?**
   Оператор def, выполняемый внутри определения функции, определяет локальную функцию, которая может быть возвращена или передана. Свободные переменные, используемые во вложенной функции, могут обращаться к локальным переменным функции, содержащей def.
   Оператор return [выражение] возвращает результат из функции. Оператор return без аргументов аналогичен return None
3. **Каково назначение локальных и глобальных переменных при написании функций в Python?**
   Все variabes Python, которые доступны в какой - то момент в коде либо в локальной области видимости или в глобальном масштабе.
   Объяснение состоит в том, что локальная область действия включает в себя все переменные, определённые в текущей функции, а глобальная область действия включает переменную, определённую вне текущей функции.
4. **Как вернуть несколько значений из функции Python?**
   С помощью оператора return из функции можно вернуть одно или несколько значений. Возвращаемым объектом может быть: число, строка, None. Чтобы вернуть несколько значений, нужно написать их через запятую.
5. **Какие существуют способы передачи значений в функцию?**
   Существует два способа передачи параметров в функцию: по значению и по адресу. При передаче по значению на месте формальных параметров записываются имена фактических параметров. При вычислении функции в стек заносятся копии значений фактических параметров, и операторы функции работают с этими копиями.
6. **Как задать значение аргументов функции по умолчанию?**
   В Python аргументам функции можно присваивать значения по умолчанию. Мы можем предоставить аргументу значение по умолчанию, используя оператор присваивания =. Вот пример: def greet(name, msg="Доброе утро!"): "" " Эта функция выводит для человека с именем name сообщение msg.
7. **Каково назначение lambda-выражений в языке Python**
   Лямбда-выражения на Python - конструкторы простых безымянных однострочных функций. Могут быть использованы везде, где требуется.
8. **Как осуществляется документирование кода согласно PEP257?**

Документирование кода в python - достаточно важный аспект, ведь от неё порой зависит читаемость и быстрота понимания вашего кода, как другими людьми, так и вами через полгода. 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.

9. **В чем особенность однострочных и многострочных форм строк документации?**
Одиночные строки документации предназначены для действительно очевидных случаев.

```
def kos_root():
    """Return the pathname of the KOS root directory."""
    global _kos_root
    if _kos_root: return _kos_root
```

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.

```
def complex(real=0.0, imag=0.0):
    """Form a complex number.

    Keyword arguments:
    real -- the real part (default 0.0)
    imag -- the imaginary part (default 0.0)

    """
    if imag == 0.0 and real == 0.0: return complex_zero
    ...
```

Ссылка на репозиторий: https://github.com/tamaranesterenko/Python.LR_11