

Федеральное государственное автономное образовательное учреждение
высшего образования «Северо-Кавказский федеральный университет»
Институт цифрового развития

ОТЧЕТ ПО ДИСЦИПЛИНЕ
ОСНОВЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ
Лабораторная работа №11

Выполнила:

Мирзаева Камилла Мирзаевна

2 курс, группа ПИЖ-б-о-20-1

Принял:

Воронкин Роман Александрович

Ставрополь, 2021 г.

ВЫПОЛНЕНИЕ

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

def positive():
    print("Positive")

def negative():
    print("Negative")

def test():
    a = int(input("Put int num: "))
    if a > 0:
        positive()
    else:
        negative()

# def positive():
#     print("Positive")
#
#
# def negative():
#     print("Negative")

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

Рисунок 1 - Задача negative, positive

```
Put int num: -6
Negative
```

Рисунок 2 – Вывод

```

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

PI = 3.14

def circle(r):
    return PI * (r * r)

def cylinder():
    h = float(input("Put high of cylinder: "))
    r = float(input("Put radius of cylinder: "))
    x = int(input("1 - Calc only side area\n2 - Calc full area\n"))

    if x == 1:
        area = 2 * PI * r * h
    elif x == 2:
        area = 2 * PI * r * h
        area = 2 * (circle(r))
    print(area)

if __name__ == '__main__':
    cylinder()

```

Рисунок 3 - Задача про площадь фигур

```

Put high of cylinder: 5
Put radius of cylinder: 8
1 - Calc only side area
2 - Calc full area
1
251.20000000000002
Process finished with exit code 0

```

Рисунок 4 - Вывод

```

Put high of cylinder: 5
Put radius of cylinder: 8
1 - Calc only side area
2 - Calc full area
2
401.92
Process finished with exit code 0

```

Рисунок 5 - Вывод

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

def mult():
    x = 1
    while True:
        a = int(input("Put numbers you want mult(0 - stop): "))
        if a == 0:
            break
        else:
            x = x * a
    if x == 1:
        exit()
    else:
        print(x)

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

Рисунок 6 - Возвращение полученного произведения

```
Put numbers you want mult(0 - stop): 0

Process finished with exit code 0
```

Рисунок 7 - Вывод

```
Put numbers you want mult(0 - stop): 3
Put numbers you want mult(0 - stop): 5
Put numbers you want mult(0 - stop): 10
Put numbers you want mult(0 - stop): 0
300

Process finished with exit code 0
```

Рисунок 8 - Вывод

Пример №1.

```
#!/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(
        "№",
        "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)

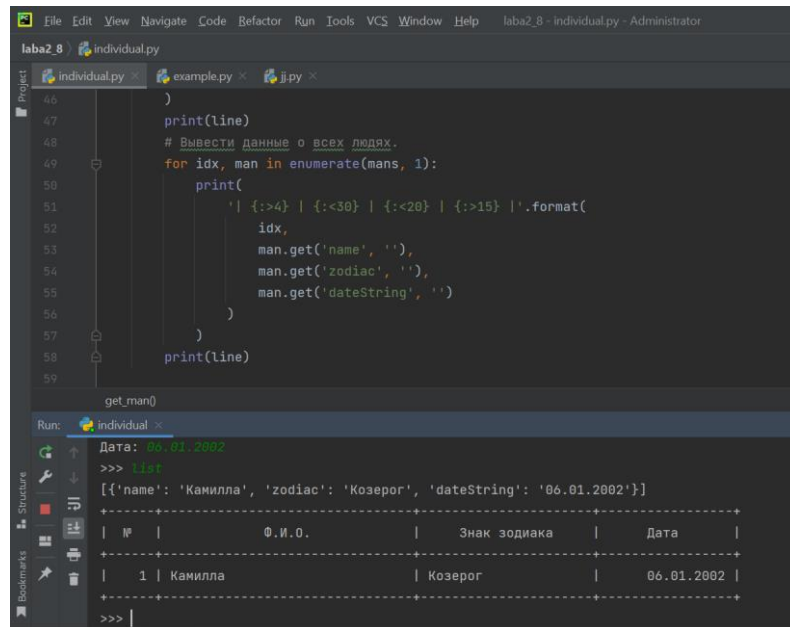
        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить работника;")
            print("list - вывести список работников;")
            print("select <стаж> - запросить работников со стажем;")
            print("help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

if __name__ == '__main__':
    main()

```

Индивидуальное задание

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



```
46 )
47 print(line)
48 # Вывести данные о всех людях.
49 for idx, man in enumerate(mans, 1):
50     print(
51         '| {:>4} | {:<30} | {:<20} | {:>15} |'.format(
52             idx,
53             man.get('name', ''),
54             man.get('zodiac', ''),
55             man.get('dateString', '')
56         )
57     )
58 print(line)
59
get_man()
```

Run: individual

Дата: 06.01.2002

>>> list

[{'name': 'Камилла', 'zodiac': 'Козерог', 'dateString': '06.01.2002'}]

№	Ф.И.О.	Знак зодиака	Дата
1	Камилла	Козерог	06.01.2002

>>> |

Ответы на контрольные вопросы

1. Функции решают проблему дублирования кода в разных местах программы. Благодаря им, есть возможность один и тот же участок кода не сразу, а когда понадобится.
2. Функции определяются оператором `def`. `Return` возвращает значение, вычисленное функцией в основное тело программы для его дальнейшей обработки.
3. Локальные и глобальные переменные призваны разграничить доступ к переменным между частями кода. Так глобальные переменные доступны в любом месте кода, когда локальные могут быть использованы, например, только в функции, если они были объявлены внутри неё.
4. Вернуть несколько значений сразу можно перечислив их через запятую после оператора `return`.

5. Передать значение в функцию можно передав имя переменных в качестве параметра, либо передав сами значения переменных.

6. Чтобы задать значение аргументов функции по умолчанию следует после объявлений всех параметров указать те, которые принимают значения по умолчанию в случае, если их значения не будут указаны (`def funct(a, b, c = 2)`).

7. Лямбды – те же функции, но с упрощенным синтаксисом, и по сути являются выражениями. Они могут быть использованы там, где не могут функции, внутри литералов или в вызовах функций.

8. Документирование кода по PEP257 предусматривает использование тройных двойных кавычек. Также существует две формы строк документации: однострочная и многострочная.

9. Однострочная строка документации не должна быть "подписью" параметров функции / метода. Этот тип строк документации подходит только для C функций, где интроспекция не представляется возможной. 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.