

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций
«ИССЛЕДОВАНИЕ ИНТЕРФЕЙСА МИКРОПРОЦЕССОРНОЙ
ЛАБОРАТОРИИ "МИКРОЛАБ"»**

**Отчет по лабораторной работе № 2.8
по дисциплине «Работа с функциями в языке Python»**

Выполнил студент группы ПИЖ-б-о-21-1

Логвинов Иван Васильевич « » 2022г.

Подпись студента_____

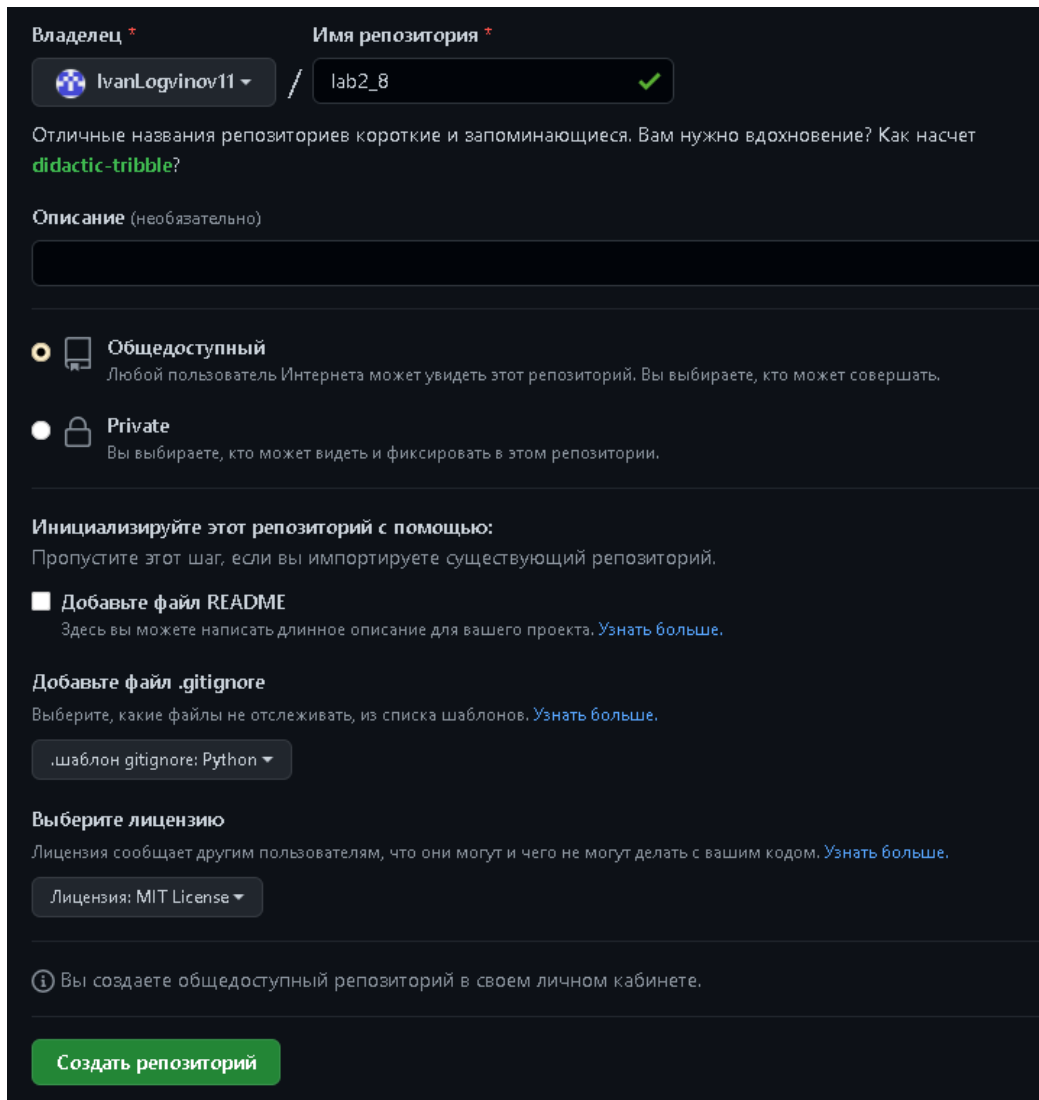
Работа защищена « »_____20__г.

Проверил Воронкин Р.А. _____

(подпись)

Ставрополь 2022

Ход работы:



The screenshot shows the GitHub 'Create new repository' page. At the top, there are two input fields: 'Владелец' (Owner) with the dropdown 'IvanLogvinov11' and 'Имя репозитория' (Repository name) with the input 'lab2_8' and a green checkmark. Below these, a tip suggests repository names should be short and memorable, with the example 'didactic-tribble?'. There is a text area for 'Описание' (Description) with the placeholder '(необязательно)'. Two radio buttons are present: 'Общедоступный' (Public) which is selected, and 'Private'. Below the radio buttons, instructions explain the visibility settings. A section titled 'Инициализируйте этот репозиторий с помощью:' (Initialize this repository with) includes a checkbox for 'Добавьте файл README' (Add a README file) and a dropdown for 'Добавьте файл .gitignore' (Add a .gitignore file) currently set to 'шаблон gitignore: Python'. Another dropdown for 'Выберите лицензию' (Choose a license) is set to 'Лицензия: MIT License'. At the bottom, a green button says 'Создать репозиторий' (Create repository).

Рисунок 1 – Создание общедоступного репозитория

1. Проработка примера из лабораторной работы:

```
#!/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()

```

Рисунок 1 – Код программы

```
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? Логвинов И.В.
Должность? студент
Год поступления? 2021
>>> add
Фамилия и инициалы? Конижев М.В.
Должность? студент
Год поступления? 2021
>>> select 1
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |   Год   |
+-----+-----+-----+-----+
|  1 | Конижев М.В.             | студент             |  2021   |
|  2 | Логвинов И.В.           | студент             |  2021   |
+-----+-----+-----+-----+
>>> add
Фамилия и инициалы? Разворотников Д.С.
Должность? попов
Год поступления? 2021
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |   Год   |
+-----+-----+-----+-----+
|  1 | Конижев М.В.             | студент             |  2021   |
|  2 | Логвинов И.В.           | студент             |  2021   |
|  3 | Разворотников Д.С.      | попов               |  2021   |
+-----+-----+-----+-----+
>>>
```

Рисунок 3 – Результат программы

Задание №1: решить следующую задачу: 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". Понятно, что вызов `test()` должен следовать после определения функций. Однако имеет ли значение порядок определения самих функций? То есть должны ли определения `positive()` и `negative()` предшествовать `test()` или могут следовать после него? Проверьте вашу гипотезу, поменяв объявления функций местами. Попробуйте объяснить результат.

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

def test():
    num = int(input("Enter number: "))
    if num > 0:
        positive()
    elif num < 0:
        negative()
    else:
        print("Your number is zero")

def positive():
    print("Your number is positive")

def negative():
    print("Your number is negative")

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

Рисунок 2 – код программы 1 задания

```
C:\Users\Иван\AppData\Local\Programs\Python\Python39\python
Enter number: 12
Your number is positive

Process finished with exit code 0
```

Рисунок 3 – результат программы 1 задания

Задание №2: Решите следующую задачу: в основной ветке программы вызывается функция `cylinder()`, которая вычисляет площадь цилиндра. В теле `cylinder()` определена функция `circle()`, вычисляющая площадь круга по формуле $S = \pi r^2$. В теле `cylinder()` у пользователя спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле $S_{\text{бок}} = 2\pi r h$, 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():
    r = float(input("Enter radius: "))
    h = float(input("Enter high: "))
    s_side = 2 * math.pi * r * h

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

    check = int(input("Enter 1 if you need side area or 2 for full area: "))
    if check == 1:
        print(f"Side area of cylinder is: {s_side}")
    else:
        full_area = s_side + circle() * 2
        print(f"Full area of cylinder is: {full_area}")

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

Рисунок 4 – Код программы 2 задания

```
C:\Users\Иван\AppData\Local\Programs\Python\Python39\pyt
Enter radius: 3
Enter high: 12
Enter 1 if you need side area or 2 for full area: 2
Full area of cylinder is: 282.7433388230814

Process finished with exit code 0
```

Рисунок 5 – Результат программы 2 задания

Задание №3: решите следующую задачу: 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 -*-

def multiply():
    f_num = 1
    while True:
        n_num = int(input("Enter multiplied number: "))
        if n_num != 0:
            f_num *= n_num
        else:
            break
    return f_num

if __name__ == "__main__":
    print(f"The product of all the numbers entered before '0': {multiply()}")
```

Рисунок 6 – Код программы 3 задания

```
Enter multiplied number: 3
Enter multiplied number: 4
Enter multiplied number: 5
Enter multiplied number: 6
Enter multiplied number: 7
Enter multiplied number: 0
The product of all the numbers entered before '0': 2520

Process finished with exit code 0
```

Рисунок 7 – Результат программы 3 задания

Задание №4: решите следующую задачу: напишите программу, в которой определены следующие четыре функции:

1. Функция `get_input()` не имеет параметров, запрашивает ввод с клавиатуры и возвращает в основную программу полученную строку.
2. Функция `test_input()` имеет один параметр. В теле она проверяет, можно ли переданное ей значение преобразовать к целому числу. Если можно, возвращает логическое `True`. Если нельзя – `False`.
3. Функция `str_to_int()` имеет один параметр. В теле преобразовывает переданное значение к целочисленному типу. Возвращает полученное число.
4. Функция `print_int()` имеет один параметр. Она выводит переданное значение на экран и ничего не возвращает.

В основной ветке программы вызовите первую функцию. То, что она вернула, передайте во вторую функцию. Если вторая функция вернула `True`, то те же данные (из первой функции) передайте в третью функцию, а возвращенное третьей функцией значение – в четвертую.

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

def get_input():
    return input("Enter num: ")

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

def str_to_int(num):
    return int(num)

def print_int(num):
    print(num, type(num))

if __name__ == "__main__":
    ent_num = get_input()
    print(ent_num, type(ent_num))
    if test_input(ent_num):
        str_to_int(ent_num)
        print_int(str_to_int(ent_num))
    else:
        print(f"Cant use 'str_to_int' for {ent_num}")
```

Рисунок 8 – Код программы 4 задания

```
C:\Users\Иван\AppData\Local\Programs\Python\Pyt
Enter num: 9
9 <class 'str'>
9 <class 'int'>

Process finished with exit code 0
```

Рисунок 9 – Результат программы 4 задания

Индивидуальное задание: решить индивидуальное задание лабораторной работы 2.6, оформив каждую команду в виде отдельной функции.

Код программы:

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

import sys

def get_bank_acc():
    """
    Request for bank account details with verification
    """
    while True:
        s_b_a = input("Enter the sender's bank account: ")
        if len(s_b_a) != 20 or not s_b_a.isdigit():
            print("Incorrect bank account!")
        else:
            break

    while True:
        b_a = input("Enter the beneficiary's account: ")
        if len(b_a) != 20 or not b_a.isdigit():
            print("Incorrect bank account!")
        else:
            break

    t_a = input("Enter transfer amount in P: ")

    return {
        "s_b_a": s_b_a,
        "b_a": b_a,
        "t_a": t_a,
    }

def display_acc(accounts):
    """
    Display of entered bank accounts
    """
    if accounts:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
```

```

        '-' * 2,
        '-' * 25,
        '-' * 25,
        '-' * 10
    )
    print(line)
    print(
        '| {:^2} | {:^25} | {:^25} | {:^10} |'.format(
            "№",
            "Sender bank account",
            "beneficiary account",
            "Amount",
        )
    )
    print(line)

    for ind, requisite in enumerate(accounts, 1):
        print(
            '| {:^2} | {:^25} | {:^25} | {:^10} |'.format(
                ind,
                requisite.get('s_b_a'),
                requisite.get('b_a'),
                requisite.get('t_a'),
            )
        )
        print(line)
    else:
        print("You have no bank accounts for now!")

def sum_check(requisites, account):
    """
    Amount of all money withdrawn
    """
    full_summa = 0
    for sender_req in requisites:

        if int(sender_req.get("s_b_a")) == int(account):
            full_summa += float(sender_req.get("t_a"))

    if full_summa == 0:
        print("This bank account does not exist")
    else:
        print(full_summa)

def help_me():
    print("Command List:\n")
    print("add - Add bank account;")
    print("list - Display a list of bank accounts;")
    print("select <bank account> -", end=" ")
    print("The withdrawn amount from account;")
    print("help - Display Help;")
    print("exit - End the program.")
    print("\n")

def invalid_com():
    print('\n')
    print(f"Invalid command use help", file=sys.stderr)

def main():
    """

```

```

Main function
"""
requisites = []
while True:

    command = input("Enter Command: ").lower()
    if command == "exit":
        break

    elif command == "add":
        requisite = get_bank_acc()
        requisites.append(requisite)

        if len(requisites) > 1:
            requisites.sort(key=lambda item: item.get("s_b_a", ""))

    elif command == "list":
        display_acc(requisites)

    elif command.startswith("select "):
        parts = command.split(" ", maxsplit=1)
        bank_acc = parts[1]
        sum_check(requisites, bank_acc)

    elif command == 'help':
        help_me()

    else:
        invalid_com()

if __name__ == '__main__':
    main()

```

Рисунок 10 – Код программы индивидуального задания

```
C:\Users\Иван\AppData\Local\Programs\Python\Python39\pythonw.exe C:/Users/Иван
Enter Command: help
Command List:

add - Add bank account;
list - Display a list of bank accounts;
select <bank account> - The withdrawn amount from account;
help - Display Help;
exit - End the program.

Enter Command: add
Enter the sender's bank account: 12345678900987654321
Enter the beneficiary's account: 12121213141435356575
Enter transfer amount in ₺: 123123123
Enter Command: add
Enter the sender's bank account: 12312343123454657686
Enter the beneficiary's account: 12123123123132123133
Enter transfer amount in ₺: 123123321
Enter Command: list
+---+-----+-----+-----+-----+
| № | Sender bank account | beneficiary account | Amount |
+---+-----+-----+-----+-----+
| 1 | 12312343123454657686 | 12123123123132123133 | 123123321 |
+---+-----+-----+-----+-----+
| 2 | 12345678900987654321 | 12121213141435356575 | 123123123 |
+---+-----+-----+-----+-----+
Enter Command:
```

Рисунок 11 – Результат работы программы индивидуального задания

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.

2. Каково назначение операторов def и return ?

В языке программирования Python функции определяются с помощью оператора def

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`.

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.

4. Как вернуть несколько значений из функции Python?

5. Какие существуют способы передачи значений в функцию?

Через параметры, и через ввод, запрашиваемый самой функцией

6. Как задать значение аргументов функции по умолчанию?

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

8. Как осуществляется документирование кода согласно PEP257?

Документирование кода в python - достаточно важный аспект, ведь от нее 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?

Для согласованности, всегда используйте `"""triple double quotes"""` для строк документации. Используйте `r"""raw triple double quotes"""`, если вы будете использовать обратную косую черту в строке документации. Существует две формы строк документации: однострочная и многострочная.