

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №5
дисциплины
«Основы кроссплатформенного программирования»

Выполнил:
Ляш Денис Александрович
2 курс, группа ИТС-б-о-23-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи», очная
форма обучения

(подпись)

Проверил:
Доцент департамента цифровых,
робототехнических систем и
электроники Воронкин Р.А.

(подпись)

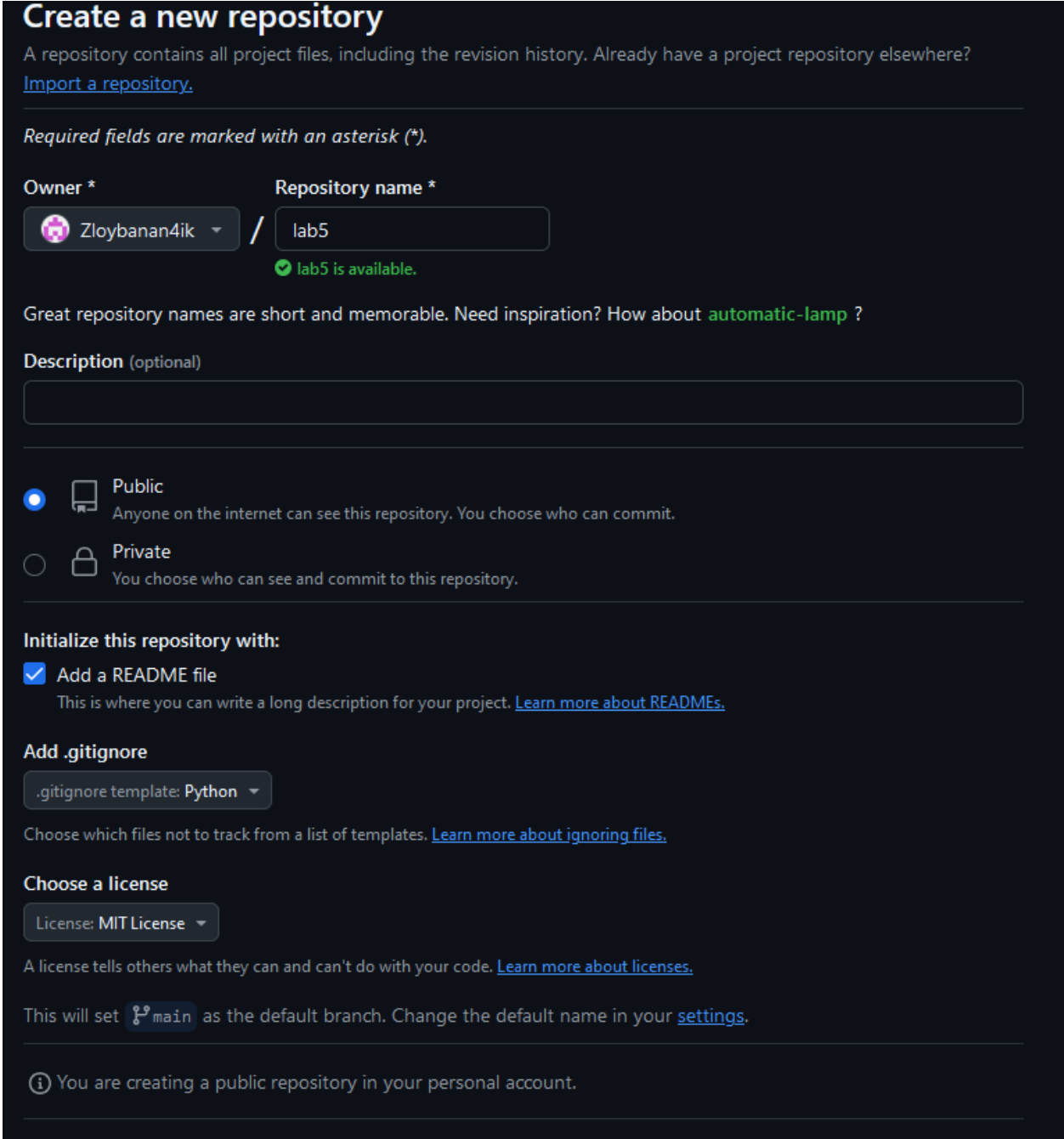
Ставрополь, 2024 г.

Тема: Условные операторы и циклы в языке Python

Цель: исследовать условные операторы и циклы в языке Python

Порядок выполнения работы:

1. Изучил теоретический материал.
2. Приступил к выполнению заданий.
3. Создал общедоступны репозиторий.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Required fields are marked with an asterisk ().*


Owner * Zloybanan4ik / Repository name * lab5

✔ lab5 is available.

Great repository names are short and memorable. Need inspiration? How about [automatic-lamp](#) ?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: Python

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).


 You are creating a public repository in your personal account.

Рисунок 1. Репозиторий

4. Выполнил клонирование репозитория

```
user@DESKTOP-VI4I94J MINGW64 ~/workspace
$ git clone https://github.com/Zloybanan4ik/lab5.git
Cloning into 'lab5'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), done.
```

Рисунок 2. Клонирование

5. Изучил рекомендации PEP-8

6. Создал проект

7. Приступил к проработке примеров

```
#!/usr/bin/env python3
```

```
# -*- coding: utf-8 -*-
```

```
import math
```

```
if __name__ == '__main__':
```

```
    x = float(input("Value of x? "))
```

```
    if x <= 0:
```

```
        y = 2 * x * x + math.cos(x)
```

```
    elif x < 5:
```

```
        y = x + 1
```

```
    else:
```

```
        y = math.sin(x) - x * x
```

```
    print(f'y = {y}')
```

```
Value of x? 5
y = -25.95892427466314

Process finished with exit code 0
|
```

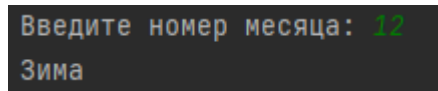
Рисунок 5. Результат первого примера

```

import sys

if __name__ == '__main__':
    n = int(input("Введите номер месяца: "))
    if n == 1 or n == 2 or n == 12:
        print("Зима")
    elif n == 3 or n == 4 or n == 5:
        print("Весна")
    elif n == 6 or n == 7 or n == 8:
        print("Лето")
    elif n == 9 or n == 10 or n == 11:
        print("Осень")
    else:
        print("Ошибка!", file=sys.stderr)
        exit(1)

```



Введите номер месяца: 12
Зима

Рисунок 6. Результат второго примера

```

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

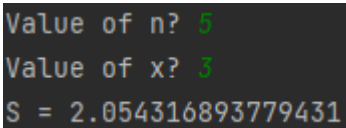
import math

if __name__ == '__main__':
    n = int(input("Value of n? "))
    x = float(input("Value of x? "))
    S = 0.0

    for k in range(1, n + 1):
        a = math.log(k * x) / (k * k)

```

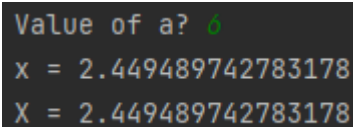
```
S += a
print(f'S = {S}')
```



```
Value of n? 5
Value of x? 3
S = 2.054316893779431
```

Рисунок 7. Результат третьего примера

```
import math
import sys
if __name__ == '__main__':
    a = float(input("Value of a? "))
    if a < 0:
        print("Illegal value of a", file=sys.stderr)
        exit(1)
    x, eps = 1, 1e-10
    while True:
        xp = x
        x = (x + a / x) / 2
        if math.fabs(x - xp) < eps:
            break
    print(f'x = {x}\nX = {math.sqrt(a)}')
```



```
Value of a? 6
x = 2.449489742783178
X = 2.449489742783178
```

Рисунок 8. Результат четвертого примера

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

```
import math
import sys
```

```

# Постоянная Эйлера.
EULER = 0.5772156649015328606

# Точность вычислений.
EPS = 1e-10

if __name__ == '__main__':
    x = float(input("Value of x? "))
    if x == 0:
        print("Illegal value of x", file=sys.stderr)
        exit(1)

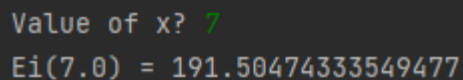
    a = x
    S, k = a, 1

# Найти сумму членов ряда.

while math.fabs(a) > EPS:
    a *= x * k / (k + 1) ** 2
    S += a
    k += 1

# Вывести значение функции.
print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")

```



```

Value of x? 7
Ei(7.0) = 191.50474333549477

```

Рисунок 9. Результат пятого примера

8. Зафиксировал изменения в репозитории

```

user@DESKTOP-VI4I94J MINGW64 ~/workspace/lab5 (main)
$ git add .gitignore

user@DESKTOP-VI4I94J MINGW64 ~/workspace/lab5 (main)
$ git commit -m "updated .gitignore"
[main f0fa401] updated .gitignore
1 file changed, 1 insertion(+)

user@DESKTOP-VI4I94J MINGW64 ~/workspace/lab5 (main)
$ git add .

user@DESKTOP-VI4I94J MINGW64 ~/workspace/lab5 (main)
$ git commit -m "Examples"
[main 881c690] Examples
5 files changed, 83 insertions(+)
create mode 100644 ex1.py
create mode 100644 ex2.py
create mode 100644 ex3.py
create mode 100644 ex4.py
create mode 100644 ex5.py

```

Рисунок 10. Коммит

9. Приступил к выполнению индивидуального задания

```

x1 = float(input("Enter the price of one roll of wallpaper (x1): "))
x2 = float(input("Enter the price of one can of paint (x2): "))
total_cost = 8 * x1 + 2 * x2

```

```

if 200 <= total_cost <= 500:
    discount = 0.03 # 3%
elif 500 < total_cost <= 800:
    discount = 0.05 # 5%
elif 800 < total_cost <= 1000:
    discount = 0.07 # 7%
elif total_cost > 1000:
    discount = 0.09 # 9%
else:
    discount = 0.0 # Нет скидки

```

```

final_cost = total_cost * (1 - discount)

```

```

print(f"The total cost is: {final_cost:.2f} rubles")

```

```

Enter the price of one roll of wallpaper (x1): 5
Enter the price of one can of paint (x2): 3
The total cost is: 46.00 rubles

```

Рисунок 16. Результат задания 1

```

import math

x1, y1, r1 = map(float, input("Enter the coordinates of the center (x1, y1) and
radius (r1) of the first circle: ").split())
x2, y2, r2 = map(float, input("Enter the coordinates of the center (x2, y2) and
radius (r2) of the second circle: ").split())

distance = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)

if distance > r1 + r2:
    result = "No intersection points (the circles are separate).\"
elif distance < abs(r1 - r2):
    result = "No intersection points (one circle is inside the other).\"
elif distance == 0 and r1 == r2:
    result = "Infinite points of intersection (the circles coincide).\"
elif distance == r1 + r2 or distance == abs(r1 - r2):
    result = "One intersection point (the circles are tangent).\"
else:
    result = "Two intersection points (the circles intersect).\"

print(result)

```

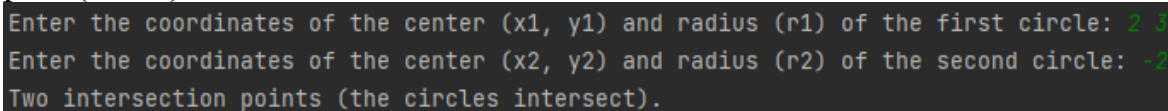


Рисунок 18. Результат задания 2

```

denominations = [500, 100, 50, 10, 5, 2, 1]

S = int(input("Enter the amount to pay (S): "))

bills_count = {}

for denomination in denominations:
    count = S // denomination # Сколько купюр данного номинала нужно
    if count > 0:
        bills_count[denomination] = count
    S %= denomination # Остаток суммы

print("The customer will give the following bills:")
for denomination, count in bills_count.items():
    print(f'{denomination} RUB: {count}')

```



```
Enter the amount to pay ($): 12
The customer will give the following bills:
10 RUB: 1
2 RUB: 1
```

Рисунок 20. Результат задания 3

10. Зафиксировал изменения

```
user@DESKTOP-VI4I94J MINGW64 ~/workspace/lab5 (main)
$ git add .

user@DESKTOP-VI4I94J MINGW64 ~/workspace/lab5 (main)
$ git commit -m "individual"
[main 8e02948] individual
3 files changed, 52 insertions(+)
create mode 100644 i1.py
create mode 100644 i2.py
create mode 100644 i3.py
```

Рисунок 22. Изменения

Вывод: в ходе работы исследовал условные операторы и циклы в языке Python

1. Для чего нужны диаграммы деятельности UML? Диаграммы деятельности UML используются для моделирования бизнес-процессов и алгоритмов.

Они помогают визуализировать последовательность действий, условия и потоки управления в системе.

2. Что такое состояние действия и состояние деятельности?

Состояние действия — это конкретное действие или операция, выполняемая в процессе. Состояние деятельности — это более общее состояние, которое может включать в себя несколько действий.

3. Какие нотации существуют для обозначения переходов и ветвлений в диаграммах деятельности?

Переходы обозначаются стрелками. Ветвления обозначаются ромбами, где условия перехода указываются на стрелках.

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Алгоритм разветвляющейся структуры — это алгоритм, который принимает решение на основе условий и выполняет разные действия в зависимости от результата.

5. Чем отличается разветвляющийся алгоритм от линейного?

Линейный алгоритм выполняет действия последовательно, без условий, а разветвляющийся алгоритм включает условия, которые определяют, какие действия выполнять.

6. Что такое условный оператор? Какие существуют его формы? Условный оператор позволяет выполнять разные действия в зависимости от истинности условия. Основные формы: if if-else if-elif-else

7. Какие операторы сравнения используются в Python?

= равно, != не равно, >, <, >=, <=.

8. Что называется простым условием?

Простое условие — это условие, состоящее из одного логического выражения.

9. Что такое составное условие?

Составное условие — это условие, состоящее из нескольких логических выражений, объединенных логическими операторами.

10. Какие логические операторы допускаются при составлении сложных условий?

And, or, not

11. Может ли оператор ветвления содержать внутри себя другие ветвления?

Да, оператор ветвления может содержать другие ветвления, создавая вложенные условия.

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм циклической структуры выполняет одно и то же действие несколько раз, пока выполняется определенное условие.

13. Типы циклов в языке Python.

for — цикл, который перебирает элементы последовательности. while — цикл, который выполняется, пока истинно заданное условие.

14. Назовите назначение и способы применения функции range.

Функция range используется для генерации последовательности чисел. Она часто применяется в циклах для итерации по числовым диапазонам.

15. Как с помощью функции range организовать перебор значений от 15 до 0 с шагом 2?

сору for i in range(15, -1, -2)

16. Могут ли быть циклы вложенными?

Да, циклы могут быть вложенными, что позволяет создавать более сложные структуры итерации.

17. Как образуется бесконечный цикл и как выйти из него?

Бесконечный цикл образуется, когда условие цикла всегда истинно. Выйти из него можно с помощью оператора break или прерывания программы.

18. Для чего нужен оператор break?

Оператор break используется для немедленного выхода из цикла, прекращая его выполнение.

19. Где употребляется оператор `continue` и для чего он используется?

Оператор `continue` используется для пропуска текущей итерации цикла и перехода к следующей.

20. Для чего нужны стандартные потоки `stdout` и `stderr`?

`stdout` используется для вывода информации, `stderr` используется для вывода сообщений об ошибках.

21. Как в Python организовать вывод в стандартный поток `stderr`?

через модуль `sys`

22. Каково назначение функции `exit`?

Функция `exit` используется для завершения программы. Она может принимать код завершения, который указывает на успешное или неуспешное завершение.