

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 №5**  
дисциплины «Основы программной инженерии»

Выполнил:  
Мелтонян Одиссей  
2 курс, группа ПИЖ-б-о-22-1,  
09.03.04 «Программная инженерия»,  
направленность (профиль) «Разработка  
и сопровождение программного  
обеспечения», очная форма обучения

---

(подпись)

Проверил: Воронкин Р. А.

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г

**Тема:** Условные операторы и циклы в языке Python

**Цель работы:** приобретение навыков программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Освоить операторы языка Python версии 3.x if, while, for, break и continue , позволяющих реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.

**Ход работы:**


1. Изучил теоретический материал
2. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

## 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 \*** **Repository name \***


 **odik8** /

✔ BSE5 is available.

Great repository names are short and memorable. Need inspiration? How about **cautious-octo-invention** ?

**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**

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

**Choose a license**

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

Рисунок 1. – Создание репозитория

### 3. Клонировал репозиторий

```
MINGW64:/c/Users/varfe/OneDrive/Рабочий стол/Воронкин
varfe@DESKTOP-8SV1DU3 MINGW64 ~/OneDrive/Рабочий стол/Воронкин
$ git clone https://github.com/odik8/BSE5.git
Cloning into 'BSE5'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

varfe@DESKTOP-8SV1DU3 MINGW64 ~/OneDrive/Рабочий стол/Воронкин
$
```

Рисунок 2. – Клонирование репозитория

#### 4. Дополнил файл .gitignore

```
.gitignore - Блокнот
Файл Правка Формат Вид Справка
*.sage.py

# Environments
.env
.venv
env/
venv/
ENV/
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/

# Cython debug symbols
cython_debug/

# PyCharm
# JetBrains specific template is maintained in a separate JetBrains.gitignore that can
# be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
# and can be added to the global gitignore or merged into this file. For a more nuclear
# option (not recommended) you can uncomment the following to ignore the entire idea folder.
# .idea/

/
```

Рисунок 3. – Файл .gitignore

5. Организовал свой репозиторий в соответствии с моделью ветвления git-flow. Для этого создал ветку develop в которую будут сливаться ветки features.

```
varfe@DESKTOP-8SV1DU3 MINGW64 ~/OneDrive/Рабочий стол/Воронкин/bse5 (main)
$ git branch develop
```

Рисунок 4. – Создание ветки develop

6. Изучил рекомендации к оформлению исходного кода на языке Python PEP-8. Выполнил оформление исходного примеров лабораторной работы и индивидуальных созданий в соответствии с PEP-8.

7. Создал проект PyCharm в папке репозитория

### **Индивидуальные задания:**

Вариант 11

11\_1. Компания по снабжению электроэнергией взимает плату с клиентов по тарифу:

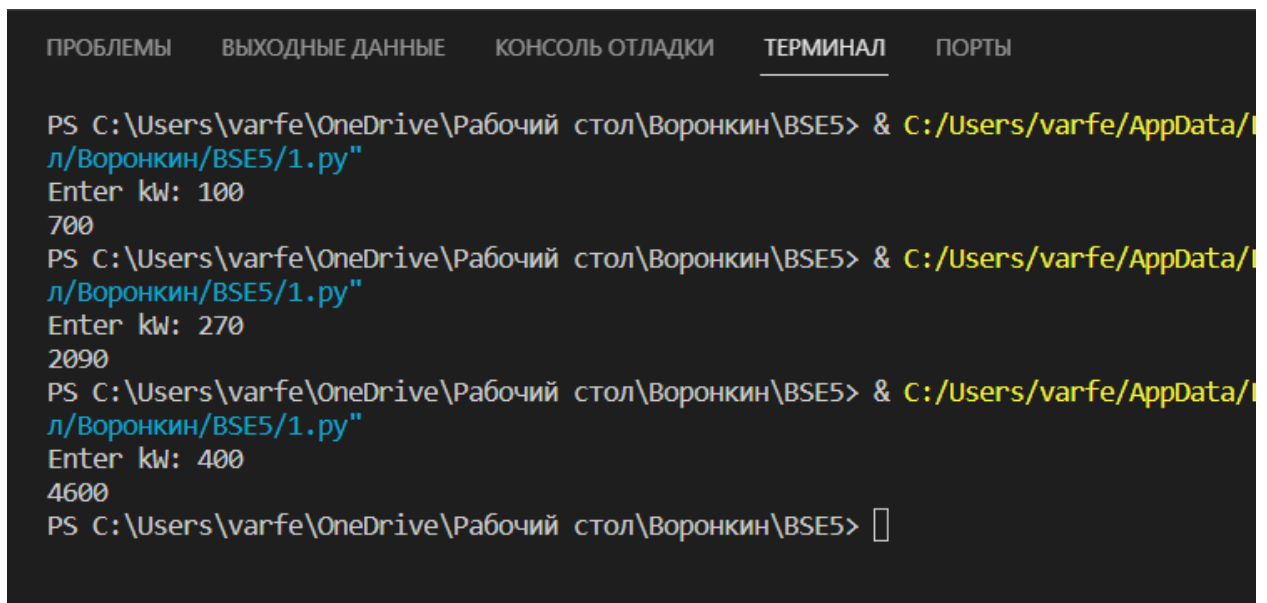
- 7 р. за 1 кВт/ч за первые 250 кВт/ч;
- 17 р. за кВт/ч, если потребление свыше 250, но не превышает 300 кВт/ч;
- 20 р. за кВт/ч, если потребление свыше 300 кВт/ч.
- Потребитель израсходовал  $n$  кВт/ч. Подсчитать плату.

Код:

```
1.py > ...
1  def main():
2      n = int(input("Enter kW: "))
3      payment = 0
4      for i in range(1, n+1):
5          if i <= 250: payment += 7
6          elif 250 < i <= 300: payment += 17
7          else: payment += 20
8
9      return payment
10
11
12  if __name__ == "__main__":
13      print(main())
```

Рисунок 5. – Код решения

Результаты при различных исходных данных:

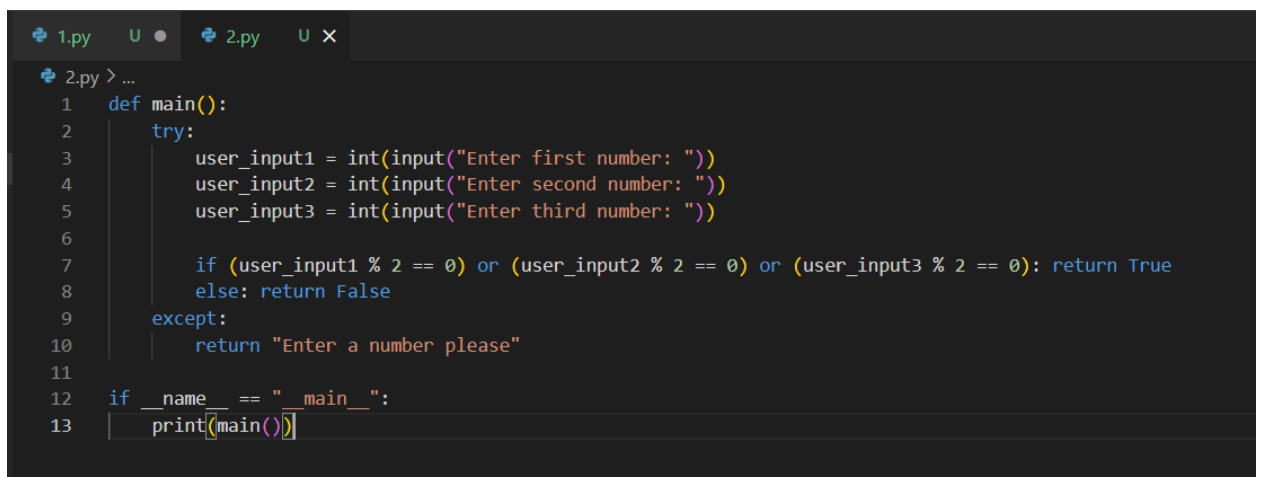


```
PS C:\Users\varfe\OneDrive\Рабочий стол\Воронкин\BSE5> & C:/Users/varfe/AppData/Local/Воронкин/BSE5/1.py
Enter kw: 100
700
PS C:\Users\varfe\OneDrive\Рабочий стол\Воронкин\BSE5> & C:/Users/varfe/AppData/Local/Воронкин/BSE5/1.py
Enter kw: 270
2090
PS C:\Users\varfe\OneDrive\Рабочий стол\Воронкин\BSE5> & C:/Users/varfe/AppData/Local/Воронкин/BSE5/1.py
Enter kw: 400
4600
PS C:\Users\varfe\OneDrive\Рабочий стол\Воронкин\BSE5> 
```

Рисунок 6. – Результат выполнения команды

11\_2. Определить, есть ли среди трёх заданных чисел чётные.

Код решения:



```
1 def main():
2     try:
3         user_input1 = int(input("Enter first number: "))
4         user_input2 = int(input("Enter second number: "))
5         user_input3 = int(input("Enter third number: "))
6
7         if (user_input1 % 2 == 0) or (user_input2 % 2 == 0) or (user_input3 % 2 == 0): return True
8         else: return False
9     except:
10        return "Enter a number please"
11
12 if __name__ == "__main__":
13     print(main())
```

Рисунок 7. – Код решения второй задачи



1		1	2	3	4	5	6	7	8	9
2		2	4	6	8	10	12	14	16	18
3		3	6	9	12	15	18	21	24	27
4		4	8	12	16	20	24	28	32	36
5		5	10	15	20	25	30	35	40	45
6		6	12	18	24	30	36	42	48	54
7		7	14	21	28	35	42	49	56	63
8		8	16	24	32	40	48	56	64	72
9		9	18	27	36	45	54	63	72	81

Рисунок 10. – Результат выполнения кода

Задание повышенной сложности:

Составить UML-диаграмму деятельности, программу и произвести вычисления вычисление значения специальной функции по ее разложению в ряд с точностью  $\varepsilon = 10^{-10}$ , аргумент функции вводится с клавиатуры.

11. Интегральный синус:

$$\text{Si}(x) = \int_0^x \frac{\sin t}{t} dt = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)(2n+1)!}.$$

Код:



```
hard.py x 3.py
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5
6  1 usage new *
7  def integral_sine(x, epsilon=1e-10):
8      result = 0
9      term = x
10     n = 1
11
12     # Вычисление ряда с учетом заданной точности
13     while abs(term) > epsilon:
14         result += term
15         n += 2
16         term *= -1 * (x**2) / (n * (n - 1))
17
18     return result
19
20 if __name__ == "__main__":
21     x = float(input("Введите значение x для вычисления Si(x): "))
22
23     print(f"Значение Si({x}) = {integral_sine(x)}")
24
```

Рисунок 11. – Код решения задачи повышенной сложности

```
Run hard x
C:\Users\varfe\AppData\Local\Programs\Python\Py
Введите значение x для вычисления Si(x): 5
Значение Si(5.0) = -0.9589242746836518
Process finished with exit code 0
```

Рисунок 12. – Результат выполнения кода

1. **Для чего нужны диаграммы деятельности UML?** – Диаграммы деятельности UML используются для визуализации и моделирования бизнес-процессов, системных алгоритмов, их взаимодействия и потоков данных.

2. **Что такое состояние действия и состояние деятельности?** – Состояние действия в диаграммах деятельности обозначает выполнение конкретной операции, а состояние деятельности представляет собой более общий термин, описывающий какую-либо активность или процесс.

3. **Какие нотации существуют для обозначения переходов и ветвлений в диаграммах деятельности?** – Стрелки для переходов, ромбы для условных ветвлений.

4. **Какой алгоритм является алгоритмом разветвляющейся структуры?** – Алгоритм с точкой разветвления, где выполнение может перейти по разным путям в зависимости от условия.

5. **Чем отличается разветвляющийся алгоритм от линейного?** – Разветвляющийся алгоритм предоставляет несколько путей выполнения, в то время как линейный выполняется последовательно без разветвлений.

6. **Что такое условный оператор? Какие существуют его формы?** – Условный оператор позволяет выполнение блока кода при определенном условии. Формы: **if** (если), **if-else** (если-иначе), **if-elif-else** (если-иначе если-иначе).

7. **Какие операторы сравнения используются в Python?** – **==** (равно), **!=** (не равно), **<** (меньше), **>** (больше), **<=** (меньше или равно), **>=** (больше или равно).

8. **Что называется простым условием? Приведите примеры.** – Простое условие содержит одно выражение. Пример: **if x > 0: print("Положительное число")**.

9. **Что такое составное условие? Приведите примеры.** – Составное условие включает несколько выражений с логическими операторами. Пример:  
`if x > 0 and x % 2 == 0: print("Положительное четное число").`

10. **Какие логические операторы допускаются при составлении сложных условий?** – **and** (логическое И), **or** (логическое ИЛИ), **not** (логическое НЕ).

11. **Может ли оператор ветвления содержать внутри себя другие ветвления?** – Да, оператор ветвления может содержать вложенные ветвления.

12. **Какой алгоритм является алгоритмом циклической структуры?** – Алгоритм циклической структуры выполняет один блок кода несколько раз.

13. **Типы циклов в языке Python.** – Цикл **for** для итерации по последовательности, цикл **while** для выполнения, пока условие истинно.

14. **Назовите назначение и способы применения функции range.** – Функция **range** создает последовательность чисел. Может принимать начало, конец и шаг.

15. **Как с помощью функции range организовать перебор значений от 15 до 0 с шагом 2?** – `for i in range(15, -1, -2): print(i).`

16. **Могут ли быть циклы вложенными?** – Да, циклы могут быть вложенными.

17. **Как образуется бесконечный цикл и как выйти из него?** – Бесконечный цикл образуется, когда условие всегда истинно. Выход – прерывание, например, **Ctrl+C**.

18. **Для чего нужен оператор break?** – Оператор **break** используется для прерывания выполнения цикла досрочно.

19. Где употребляется оператор **continue** и для чего он используется? – Оператор **continue** пропускает текущую итерацию цикла, переходя к следующей.

20. Для чего нужны стандартные потоки **stdout** и **stderr**? – **stdout** для вывода информации, **stderr** для вывода сообщений об ошибках.

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

—

```
import sys;
```

```
sys.stderr.write("Сообщение об ошибке\n").
```

22. Каково назначение функции **exit**? – Функция **exit** используется для завершения программы, принимая код возврата (0 – успешное завершение).