МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВВГУ»)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАННЫХ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

по дисциплине
«Информатика и программирование»

Студент
гр. БИН-25-3 _____ Р.А. Сакс

Ассистент
преподавателя _____ М.В. Водяницкий

Владивосток 2025

Задание

Выполнить задания на Python по работе с функциями и пользовательским вводом, оформить отчет по стандартам ВВГУ.

Задание 1. Написать функцию, которая конвертирует время из одной величины в другую. На вход подается: число (величина времени), исходная единица измерения, единица измерения, в которую нужно перевести. Функция должна вернуть конвертированное значение.

Задание 2. Написать функцию для расчета прибыли по банковскому вкладу. Пользователь делает вклад в банке в размере ‘a‘ рублей сроком на ‘n‘ лет. Процент по вкладу зависит от суммы и срока. Используется сложный процент: каждый год процент начисляется на текущую сумму вклада.

Задание 3. Написать функцию для вывода всех простых чисел в заданном диапазоне. Нужно учитывать некорректные данные (например, начало больше конца или диапазон без простых чисел). На вход подаются два числа: начало и конец диапазона (включительно). На выходе - список всех простых чисел или сообщение об ошибке.

Задание 4. Реализовать функцию сложения двух матриц. При сложении двух матриц получается новая матрица того же размера, где каждый элемент - это сумма элементов с тем же индексом из двух исходных матриц. Складывать можно только матрицы одинакового размера, размер матрицы должен быть строго больше 2.

Задание 5. Написать функцию, которая определяет, является ли строка палиндромом. Палиндром - это строка, которая читается одинаково слева направо и справа налево (без учета пробелов, регистра и знаков препинания).

Содержание

1 Выполнение работы

1.1 Задание 1

В данном задании была создана функция для конвертации времени из одной единицы измерения в другую. Функция поддерживает конвертацию между часами (H) и минутами (M).

```
def Lab6Zad1():
    def Zad(time0, ed0, ed2):
        time0 = int(time0)
        ed0, ed2 = str(ed0).lower(), str(ed2).lower()
        if ed0 == "h" and ed2 == "m":
            return f"{time0}H == {time0 * 60}M"
        elif ed2 == "h" and ed0 == "m":
            return f"{time0}M == {time0 / 60}H"
        else:
            return f"{time0}{ed0} -            "

    #
    print("    1:          ")
    print(Zad(4, "h", "m"))
    print(Zad(30, "m", "h"))
    print(Zad(12, "s", "h"))

if __name__ == "__main__":
    Lab6Zad1()
```

Рисунок 1 – Листинг программы для задания 1

Пояснение работы программы:

1) Создана функция `Zad(time0, ed0, ed2)`, которая принимает три параметра: значение времени, исходную единицу измерения и целевую единицу измерения.

2) Функция преобразует единицы измерения к нижнему регистру для унификации обработки.

3) Реализована логика конвертации:

- Если исходная единица - часы (H), а целевая - минуты (M), выполняется умножение на 60.
- Если исходная единица - минуты (M), а целевая - часы (H), выполняется деление на 60.

4) Для других комбинаций единиц измерения функция возвращает исходные данные.

5) Использованы f-строки для форматирования вывода результата.

Результат выполнения:

```
12H == 720M
```

1.2 Задание 2

Для решения задачи была создана функция расчета прибыли по банковскому вкладу с учетом сложного процента. Процентная ставка зависит от суммы вклада и срока размещения.

```
def Lab6Zad1():
    def Zad(time0, ed0, ed2):
        time0 = int(time0)
        ed0, ed2 = str(ed0).lower(), str(ed2).lower()
        if ed0 == "h" and ed2 == "m":
            return f"{time0}H == {time0 * 60}M"
        elif ed2 == "h" and ed0 == "m":
            return f"{time0}M == {time0 / 60}H"
        else:
            return f"{time0}{ed0} -             "

    #
    print("    1:           ")
    print(Zad(4, "h", "m"))
    print(Zad(30, "m", "h"))
    print(Zad(12, "s", "h"))

if __name__ == "__main__":
    Lab6Zad1()
```

Рисунок 2 – Листинг программы для задания 2

Пояснение работы программы:

1) Создана функция `Zad(SumNak, years)` для расчета прибыли.

2) Реализована проверка минимальной суммы вклада (30 000 рублей).

3) Определена базовая процентная ставка в зависимости от срока:

- До 3 лет включительно: 3%
- От 4 до 6 лет: 5%
- Более 6 лет: 2%

4) Рассчитана дополнительная ставка в зависимости от суммы вклада:

- Каждые 10 000 рублей увеличивают ставку на 0.3%
- Максимальное увеличение ограничено 5%

5) Применен сложный процент: каждый год процент начисляется на текущую сумму вклада.

6) Прибыль рассчитана как разница между итоговой и начальной суммой.

7) Результаты выводятся с округлением до 2 знаков после запятой (копеек).

Результат выполнения:

==========

```
   : 30000   ,    : 3
          : 3.0%
    : 2781.82
---------------------------------------
   : 100000   ,    : 5
          : 5.0%
    : 27628.16
---------------------------------------
   : 200000   ,    : 8
          : 7.0%
    : 172008.68
---------------------------------------


             :
   : 50000   ,    : 4
          : 5.6%
    : 12212.85
---------------------------------------
   : 150000   ,    : 2
          : 7.5%
    : 23343.75
---------------------------------------
```

1.3 Задание 3

В данном задании была реализована функция для нахождения всех простых чисел в заданном диапазоне с обработкой некорректных входных данных.

```python
def Lab6Zad3():
    def SimpleNumber(A, B):
        Kop = []
        if A < B:
            for i in range(A, B + 1):
                if i > 1:
                    flagSimple = True
                    for j in range(2, i):
                        if i % j == 0 and j != i:
                            flagSimple = False
                            break
                    if flagSimple:
                        Kop.append(i)
        if len(Kop) != 0:
            print("              :", *Kop)
        else:
            print("Error!                ")

    #
    print("    3:           ")
    print("    1-10:")
    SimpleNumber(1, 10)
    print("\    n 15-120:")
    SimpleNumber(15, 120)
    print("\    n 0-1:")
    SimpleNumber(0, 1)

if __name__ == "__main__":
    Lab6Zad3()
```

Рисунок 3 – Листинг программы для задания 3

Пояснение работы программы:

1) Создана функция `SimpleNumber(A, B)` для поиска простых чисел в диапазоне от A до B включительно.

2) Реализована проверка корректности диапазона (A должно быть меньше B).

3) Для каждого числа в диапазоне проверяется, является ли оно простым:

- Число должно быть больше 1.
- Проверяется делимость на все числа от 2 до B-1.
- Если число делится без остатка на какое-либо число кроме себя, оно не является простым.

4) Найденные простые числа сохраняются в списке `Kop`.

5) Если список пустой (простых чисел не найдено), выводится сообщение ”Error”.

6) Иначе простые числа выводятся через пробел с использованием оператора распаковки `*`.

Результат выполнения:

```
2 3 5 7 11 13
2 3 5 7
17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79
```

```
83 89 97 101 103 107 109 113
Error
```

1.4 Задание 4

Задача решена путем создания функции сложения двух квадратных матриц одинакового размера с обработкой ошибок ввода и проверкой корректности данных.

```python
def Lab6Zad4():
    def SumMatrix():
        #
        n_str = input("             n: ")

        #
        try:
            n = int(n_str)
        except:
            print("Error!")
            return

        #
        if n <= 2:
            print("Error!                2")
            return

        print(f"             {n}x{n}  (    ):")
        #
        matrix1 = []
        i = 0
        while i < n:
            try:
                row_str = input(f"    {i+1}: ")
            except:
                print("Error!")
                return

            parts = row_str.split()

            #
            if len(parts) != n:
                print("Error!                ")
                return

            #
            ch2 = []
            j = 0
            while j < n:
                try:
                    num = int(parts[j])
                except:
                    print("Error!             ")
                    return
                ch2.append(num)
                j += 1

            matrix1.append(ch2)
            i += 1

        print(f"             {n}x{n}  (    ):")
        #
        matrix2 = []
        i = 0
        while i < n:
            try:
                row_str = input(f"    {i+1}: ")
            except:
                print("Error!")
                return

            parts = row_str.split()

            #
            if len(parts) != n:
                print("Error!                ")
                return

            #
            ch2 = []
            j = 0
            while j < n:
                try:
                    num = int(parts[j])
                except:
                    print("Error!             ")
                    return
```

Пояснение работы программы:

1) Создана функция `SumMatrix()` для сложения двух матриц.

2) Реализован пошаговый ввод данных:

- Сначала вводится размер матрицы n.
- Затем построчно вводятся элементы первой матрицы.
- После этого построчно вводятся элементы второй матрицы.

3) Выполнены проверки корректности данных:

- Размер матрицы должен быть целым числом больше 2.
- Количество элементов в каждой строке должно соответствовать размеру матрицы.
- Все элементы должны быть целыми числами.

4) При обнаружении ошибки выводится сообщение ”Error!”.

5) Матрицы складываются поэлементно: каждый элемент результирующей матрицы равен сумме соответствующих элементов исходных матриц.

6) Результат выводится в том же формате, что и ввод - построчно, элементы через пробел.

Результат выполнения (пример ввода):

```
3
2 4 5
5 6 7
3 0 1
1 1 1
1 1 1
1 1 1
```

Вывод:

```
3 5 6
6 7 8
4 1 2
```

1.5 Задание 5

В данном задании была реализована функция проверки строки на палиндром с учетом игнорирования пробелов, регистра и знаков препинания.

```python
def Lab6Zad5():
    def palindrome():
        #
        text = input("               : ")

        #
        text = text.lower()

        #           -     (,          )
        letters = "
                  abcdefghijklmnopqrstuvwxyz0123456789"
        clean_text = ""

        for char in text:
            if char in letters:
                clean_text += char

        print(f"         : {clean_text}")

        #
        n = len(clean_text)
        is_pal = True

        for i in range(n // 2):
            if clean_text[i] != clean_text[n - i - 1]:
                is_pal = False
                break

        if is_pal:
            print("     :  ")
        else:
            print("     :   ")

    #
    print("    5:             ")

    #
    test_cases = [
        "                ",
        "Borrow or rob",
        "          "
    ]

    for test in test_cases:
        print(f"\  n: '{test}'")
        #
        import io
        import sys

        old_stdin = sys.stdin
        sys.stdin = io.StringIO(test)
        try:
            palindrome()
        finally:
            sys.stdin = old_stdin

if __name__ == "__main__":
    Lab6Zad5()
```

Рисунок 5 – Листинг программы для задания 5

Пояснение работы программы:

1) Создана функция `palindrome()` для проверки строк на палиндром.

2) Строка приводится к нижнему регистру для регистронезависимой проверки.

3) Выполняется очистка строки от не-буквенных символов:

- Сохраняются только русские и английские буквы, а также цифры.

- Удаляются пробелы, знаки препинания и другие символы.

4) Проверка палиндрома выполняется путем сравнения символов с начала и конца очищенной строки:

- Сравнивается первый символ с последним.
- Затем второй с предпоследним и т.д.
- Если все пары символов совпадают, строка является палиндромом.

5) Результат выводится в виде ”Да”(палиндром) или ”Нет”(не палиндром).

Результат выполнения:

```
Borrow or rob
```