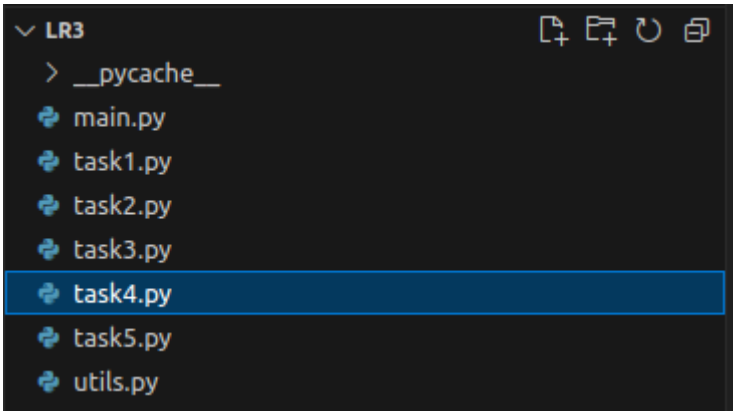
Лабораторная работа №3

Тема: Стандартные типы данных, коллекции, функции, модули.

Выполнил: студент группы 253502 Шишко Виктор

- Общая структура лабораторной работы



1. main.py - главный файл, в котором собраны все задания из отдельных модулей.

```python
"""Laboratory 3. In built data types, collections, functions, modules
Elaborated by Shyshko Victor 253502 with python 3.10.12 version
"""
import math
from task1 import custom_asin, EPS
from task2 import find_max
from task3 import count_non_whitespace
from task4 import STRING, find_letter_z, upper_amount, exclude_a_words
from task5 import generate_splice, task
from utils import validate_input, sequence_generator, sequence_generator_input
# autopep8 --in-place --aggressive --aggressive <filename.py> for PEP

def main():
    while True:
        print("Choose any number from 1 to 5 choose task number\n")
        print("1. Custom asin function")
        print("2. Find max in list")
        print("3. Count non whitespace")
        print("4. Analyze string")
        print("5. Task with list")

        task_number = input()
        match task_number:
            case "1":
                while True:
                    try:
                        x = float(input("Input any number: "))
                        if x > 1 or x < -1:
                            raise Exception
                    except Exception:
                        print(
                            "x should be in range [-1, 1] or no correct type")
                        continue

                    result = custom_asin(x, EPS)
                    print(
                        f"n = {result[0]}\nx = {x}\nF(x) = {result[1]}\nMath(F(x)) = {math.asin(x)}\neps = {EPS}")
                    break
```

```
            break
    case "2":
        while True:
            print(
                f"Choose the way of generation of list\n(1) User input\n(other) Custom generator")
            try:
                choice = int(input())
            except ValueError:
                print("Invalid input! Try again")

            if choice == 1:
                lst = validate_input(int)
            else:
                lst = list(sequence_generator_input(int))
            print(find_max(lst))
            break

    case "3":
        string = input("Input any string: ")
        print(count_non_whitespace(string))
    case "4":
        print(
            f"Upper letters amount: {upper_amount(STRING)}\nfirst letter z in occurence: {find_letter_z(STRING)}\
                \nexclude words begin with letter 'a': {exclude_a_words(STRING)}")
    case "5":
        while True:
            print(
                f"Choose the way of generation of list\n1. User input\n2. Custom generator")
            try:
                choice = int(input())
            except ValueError:
                print("Invalid input! Try again")

            if choice == 1:
                lst = validate_input(float)
            else:
                lst = list(sequence_generator_input(float))
```

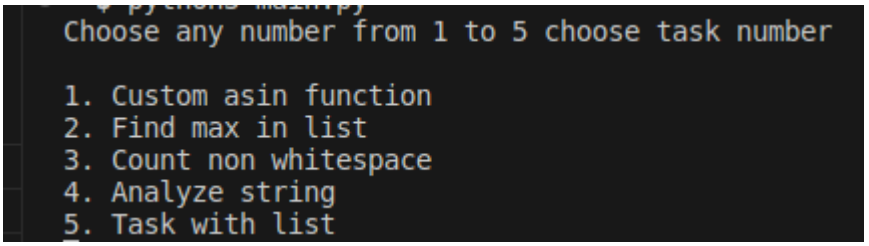
```
                        f"Choose the way of generation of list\n1. User input\n2. Custom generator")
                    try:
                        choice = int(input())
                    except ValueError:
                        print("Invalid input! Try again")

                    if choice == 1:
                        lst = validate_input(float)
                    else:
                        lst = list(sequence_generator_input(float))

                    splice = generate_splice(lst)
                    print(
                        f"List output:{lst}\nSolving task:{task(lst, splice)}")
                    break
                case "0":
                    print("Goodbye!")
                    exit(1)
                case _:
                    print("It's wrong. Just try again")
                    continue

            continue_or_exit = input("Do you want to continue? (yes/no): ")
            if continue_or_exit.lower() != "yes":
                break

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

- файл utils.py

```python
import time
import typing


def timing_average(func):
    """Custom timer decorator

    Args:
        func (_type_): any function we're going to wrap with this decorator
    """
    def wrapper(x, y):
        t0 = time.time_ns()
        res = func(x, y)
        t1 = time.time_ns()
        print("It took {} nanoseconds to calculate the average".format(t1 - t0))
        return res

    return wrapper


def validate_input(num_type: type) -> list:
    """Check for user input

    Args:
        num_type (type): type of numbers you're goig to input

    Returns:
        list: just a list
    """
    while True:
        try:
            lst = list(map(num_type, input("Input array elements: ").split()))
            return lst
        except ValueError:
            print("Invalid input! Try again")
```

```python
def sequence_generator(start, end):
    """Generates a sequence of numbers

    Args:
        start (_type_): starting number
        end (_type_): ending number

    Yields:
        _type_: keeps yielding numbers from start to end
    """
    current = start
    while current <= end:
        yield current
        current += 1


def sequence_generator_input(num_type: int | float):
    """Generates a sequence of numbers

    Yields:
        int: keeps yielding numbers from start to end
    """
    size = int(input("Enter size of sequence: "))
    it = 0
    while it < size:
        inp = num_type(input("Enter any number: "))
        yield inp
        it += 1


def custom_factorial(x: int) -> int:
    """Factorial function calculation

    Args:
        x (int): integer number

    Returns:
        int: factorial result
    """
    if x == 1 or x == 0:
        return 1
    else:
        return x * custom_factorial(x - 1)
```

При запуске имеем возможность выполнить любую задачу с возможностью повторения выполнения кода



Задание 1. В соответствии с заданием своего варианта составить программу для вычисления значения функции с помощью разложения функции в степенной ряд. Задать точность вычислений eps.

Предусмотреть максимальное количество итераций, равное 500.

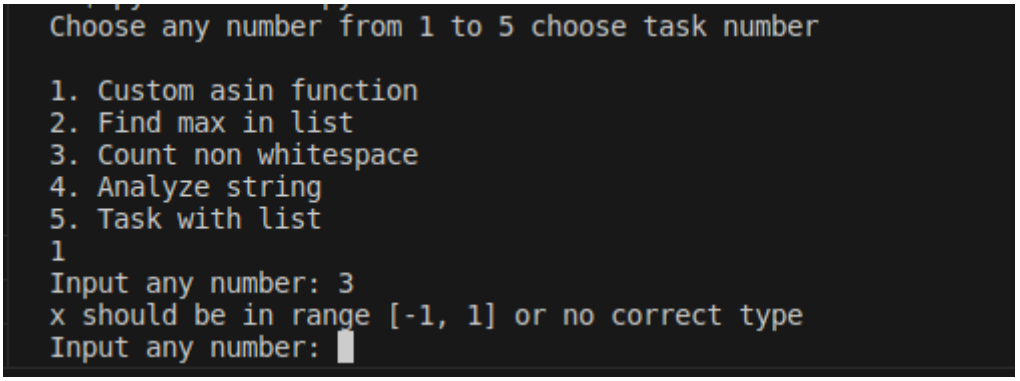
Вывести количество членов ряда, необходимых для достижения указанной точности вычислений.

- Код программы

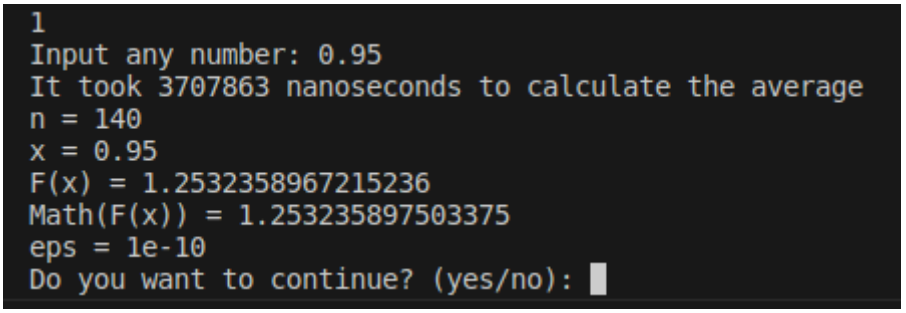
```python
from utils import custom_factorial, timing_average

"""Task 1. Variant 27. Calculate arcsin(x) function, using function expansion.
Compare with math alternative, using eps
"""
EPS = 1e-10


@timing_average
def custom_asin(x: float | int, eps: float) -> float | int:
    """Custom asin function

    Args:
        x (float | int): number we use for calculating asin
        eps (float): accuraccy calculation

    Returns:
        float | int: our result
    """
    result = 0
    prev_result = 0

    for n in range(500):
        result += custom_factorial(2 * n) / (custom_factorial(n)
                                             ** 2 * 4**n * (2 * n + 1)) * x**(2 * n + 1)
        if abs(result - prev_result) < eps:
            break
        prev_result = result

    return (n, result,)
```

Проверка выполнения кода программы:

- Некорректный пользовательский ввод



- Корректный ввод и результат выполнения программы



Задание 2. Организовать цикл, который принимает целые числа и вычисляет наибольшее из них. Окончание цикла – ввод числа 1

- Код в main.py

```
case "2":
    while True:
        print(
            f"Choose the way of generation of list\n(1) User input\n(other) Custom generator")
        try:
            choice = int(input())
        except ValueError:
            print("Invalid input! Try again")

        if choice == 1:
            lst = validate_input(int)
        else:
            lst = list(sequence_generator_input(int))
        print(find_max(lst))
        num = int(
            input("Do you want to continue?(1 to exit, or continue): "))
        if num == 0:
            continue
        else:
            break
case "3":
```

- Код программы task2.py

```python
"""Task 2. Variant 27. Find the max element in the list
"""

def find_max(lst: list) -> int:
    """Find the max element in the list

    Args:
        lst (list): the list of numbers

    Returns:
        int: returns the max element in the list
    """
    return f"Max list element: {max(lst)}"
```

В данной задаче есть два способа инициализировать последовательность(пользовательский ввод и функция генератор с вводом)

```
Choose any number from 1 to 5 choose task number

1. Custom asin function
2. Find max in list
3. Count non whitespace
4. Analyze string
5. Task with list
2
Choose the way of generation of list
(1) User input
(other) Custom generator
1
Input array elements: 1 2 -10 2 3 4 20 99 23
Max list element: 99
Do you want to continue?(1 to exit, or continue): 0
Choose the way of generation of list
(1) User input
(other) Custom generator
2
Enter size of sequence: 5
Enter any number: 1
Enter any number: 2
Enter any number: -10
Enter any number: 4
Enter any number: 5
Max list element: 5
Do you want to continue?(1 to exit, or continue):
```

Задание 3. В строке, введенной с клавиатуры, подсчитать количество символов, отличных от пробельных

- Код из main.py

```
case "3":
    string = input("Input any string: ")
    print(count_non_whitespace(string))
```

- Код программы task3.py

```
"""Task 3. Variant 27. Count non-whitespace characters in the string
"""

def count_non_whitespace(string: str) -> int:
    """Count non-whitespace characters in the string

    Args:
        string (str): any string

    Returns:
        int: amount of non-whitespace characters
    """
    return len(''.join(string.split()))
```

- Результат выполнения программы

```
Choose any number from 1 to 5 choose task number

1. Custom asin function
2. Find max in list
3. Count non whitespace
4. Analyze string
5. Task with list
3
Input any string: qw[wqr q[rqr q[krq[ qkr[qkr qkqr
28
Do you want to continue? (yes/no):
```

Задание 4. Не использовать регулярные выражения. Дана строка текста, в которой слова разделены пробелами и запятыми.

а) определить количество заглавных строчных букв;

б) найти первое слово, содержащее букву 'z' и его номер;

в) вывести строку, исключив из нее слова, начинающиеся с 'a'

«So she was considering in her own mind, as well as she could, for the hot day made her feel very sleepy and stupid, whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.»

- Код из main.py

```python
case "4":
    print(
        f"Upper letters amount: {upper_amount(STRING)}\nfirst letter z in occurence: {find_letter_z(STRING)}\
            \nexclude words begin with letter 'a': {exclude_a_words(STRING)}")
```

- Код из task4.py

```python
STRING = "wS So she was considering in her own mind, as well as she could, \
    for the hot day made her feel very sleepy and stupid, whether the pleasure of making a \
    daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a \
    White Rabbit with pink eyes ran close by her. zw"


def upper_amount(string: str) -> int:
    """Calculate the amount of upper case letters in the string

    Args:
        string (str): any sting

    Returns:
        int: amount of upper case letters
    """
    return sum(1 for char in string if char.isupper())


def find_letter_z(string: str) -> str:
    """Search for the first word with letter 'z' and it's number

    Args:
        string (str): any sting

    Returns:
        str: number and word with letter 'z'
    """
    for index, word in enumerate(start=1, iterable=string.split()):
        if 'z' in word:
            return f"{index} {word}"


def exclude_a_words(string: str) -> str:
    """Exclude words begin with letter 'a'

    Args:
        string (str): any sting

    Returns:
        str: source string, excluding words begin with letter 'a'
    """
    return ' '.join(word for word in string.split()
                    if not word.startswith('a'))
```

- Результат выполнения программы

```
Choose any number from 1 to 5 choose task number

1. Custom asin function
2. Find max in list
3. Count non whitespace
4. Analyze string
5. Task with list
4
Upper letters amount: 4
first letter z in occurence: 57 zw
exclude words begin with letter 'a': wS So she was considering in her own mind, well she could, for the hot day made her feel very sleepy stupid, whether the pleasure of making daisy-c
hain would be worth the trouble of getting up picking the daisies, when suddenly White Rabbit with pink eyes ran close by her. zw
Do you want to continue? (yes/no):
```

Задание 5. В соответствии с заданием своего варианта составить программу для обработки вещественных списков. Программа должна содержать следующие базовые функции:

1) ввод элементов списка пользователем;

2) Найти произведение элементов с четными номерами и сумму элементов, расположенных между первым и последним нулевыми элементами

3) реализация основного задания с выводом результатов;

4) вывод списка на экран.

- Код из main.py

```
    case "5":
        while True:
            print(
                f"Choose the way of generation of list\n1. User input\n2. Custom generator")
            try:
                choice = int(input())
            except ValueError:
                print("Invalid input! Try again")

            if choice == 1:
                lst = validate_input(float)
            else:
                lst = list(sequence_generator_input(float))

            splice = generate_splice(lst)
            print(
                f"List output:{lst}\nSolving task:{task(lst, splice)}")
            break
    case "0":
        print("Goodbye!")
        exit(1)
    case _:
        print("It's wrong. Just try again")
        continue

continue_or_exit = input("Do you want to continue? (yes/no): ")
if continue_or_exit.lower() != "yes":
    break
```

- Код из task5.py

```python
def task(lst: list, splice: list) -> tuple:
    """Task, that finds the mul of even list numbers and sum of elems that stay

    Args:
        lst (list): the list of numbers
        splice (list): the list of numbers

    Returns:
        tuple: returns a tuple with mul and sum results
    """
    mul = 1
    for i in lst[1::2]:
        mul *= i
    return f"Mul result: {mul}\nSum result: {sum(splice)}"


def generate_splice(splice: list) -> list:
    """Generates a splice of numbers between first and last zeros

    Args:
        splice (list): generated list

    Returns:
        list: returns a list of numbers between first and last zeros
    """
    try:
        first_zero = splice.index(0)
    except ValueError:
        first_zero = None

    try:
        last_zero = len(splice) - 1 - splice[::-1].index(0)
    except ValueError:
        last_zero = None

    if first_zero is not None and last_zero is not None:
        splice = splice[first_zero + 1:last_zero]
    else:
        splice = []
    return splice
```

- Результат выполнения программы

```
Choose any number from 1 to 5 choose task number

1. Custom asin function
2. Find max in list
3. Count non whitespace
4. Analyze string
5. Task with list
5
Choose the way of generation of list
1. User input
2. Custom generator
1
Input array elements: qrqr qr 23 13 q rq rqwr
Invalid input! Try again
Input array elements: 1 2.3 -40.2 0 221 4 0 2 3
List output:[1.0, 2.3, -40.2, 0.0, 221.0, 4.0, 0.0, 2.0, 3.0]
Solving task:Mul result: 0.0
Sum result: 225.0
Do you want to continue? (yes/no): yes
Choose any number from 1 to 5 choose task number

1. Custom asin function
2. Find max in list
3. Count non whitespace
4. Analyze string
5. Task with list
5
Choose the way of generation of list
1. User input
2. Custom generator
2
Enter size of sequence: 6
Enter any number: 1
Enter any number: 0.4
Enter any number: 2
Enter any number: 0
Enter any number: 3
Enter any number: 0
List output:[1.0, 0.4, 2.0, 0.0, 3.0, 0.0]
Solving task:Mul result: 0.0
Sum result: 3.0
Do you want to continue? (yes/no):
```