Бекарев С.С. Лабораторная работа №3
Вариант 3
Основы Python

1. Программа должна быть снабжена комментариями на английском языке, в которых необходимо указать краткое предназначение программы, номер лабораторной работы и название, версию программы, Ф.И.О. разработчика и дату разработки.

```
# This program complete a different tasks in LR3
# LR_3: "Work with Python"
# version of program : 1.0
# Bekarev Stanislav Sergeevich, 03.16.2024
```

2. Программа должна быть снабжена дружелюбным и интуитивно понятным интерфейсом

```
Lab_Rab_3:  Python

Enter 0 >>> exit
Enter 1 >>> Task1 (Tailor_Func)
Enter 2 >>> Task2 (Sum cycle)
Enter 3 >>> Task3 (Is HEX)
Enter 4 >>> Task4 (Text analyze)
Enter 5 >>> Task5 (Float_list)
```

3. Выполнить документирование кода для получения справки по каждой функции

```
@func_decorator
def tailor_func(x: float, eps: float):  # ln(1 + x)
    """
    :param x: function variable
    :param eps: accuracy of calculation
    :return:
    res - result of calculation with established accuracy
    if iterations amount exceeded return result of 500 iterations;
    n - number of iterations;
    """
```

```
def cycle(use_generator: bool, amount: int) -> int:
    """
    :param use_generator: determines whether generator will be used
    :param amount: amount of generated values
    :return:
    Result of sum all values
    """
```

```
def is_hex(string: str) -> bool:
    """
    :param string: Input string
    :return: Checks is this string a HEX number
    """
```

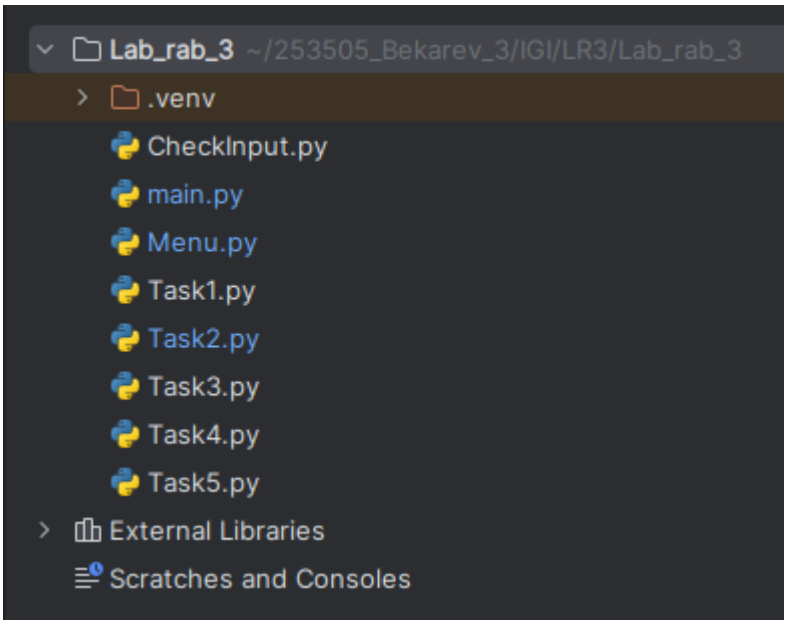
```
def text_analyze(input_string: str):
    """
    :param input_string: Input text
    :return:
    amount_str - amount of words in text;
    list_even_str - list of words with even number letters;
    min_str - minimal words than begin at 'a' if not found empty str
    repeated_str - list of words that were repeated;
    """
```

```
2 usages  Stas Bekarev
def list_analyze(float_list: list):
    """
    :param float_list: list of float values
    :return:
    max_abs_el - maximal absolute value in list;
    res - sum of values up to last positive value;
    """
```

4. Каждое задание оформить в виде отдельной бизнес-функции.

```
1 usage   Stas Bekarev
def complete_task2():
    print("*"*20)
    print("Use auto generate? [y/n]")
    str_choose = input()
    amount = -1
    use_gen = False
    if str_choose == 'y' or str_choose == 'Y':
        use_gen = True
        print("How many numbers generate?")
        amount = int_input( left: 0,  right: 0,  bound: False)
    res = cycle(use_gen, amount)
    print(f"Amount of positive number: {res}")
    print("*"*20)
    print(f"Enter to continue: ", end="")
    input()
```

5. Все функции необходимо сгруппировать в модулях, согласно их логике их работы.



6. Разработанные основные функции, размещенные в отдельных модулях, нужно подключить в другом модуле, где будет происходить тестирование данных функций.

```python
import math
import random
from CheckInput import *
from Task1 import tailor_func
from Task2 import cycle
from Task3 import is_hex
from Task4 import text_analyze
from Task5 import list_analyze


2 usages  Stas Bekarev *
def menu():
    print("\nLab_Rab_3:  Python")
    while True:
```

7. Размерность списка задается пользователем.

```
Enter 5 >>> Task5 (Float_list)
5
********************
Enter a size of array:
12
Use auto generation? [y/n]
y
Resulted list:
[10.0, -3.0, 4.0, 4.0, 4.0, 5.0, 1.0, 1.0, -1.0, 2.0, -1.0, -1.0]
Maximal absolute value: 10.0
Sum ending last positive number: 27.0
********************
Enter to continue:
```

8. Предусмотреть способы инициализации последовательности: с помощью **функции генератора** и пользовательского ввода. Оформить способы инициализации в виде отдельных функций, которые на вход принимают последовательность для инициализации, и сгруппировать эти функции в

отдельный модуль от основной функции программы.

```
1 usage  Stas Bekarev
def num_generator(amount):
    while amount > 0:
        yield random.randint(-9,  b: 9)
        amount -= 1
```

9. Продемонстрировать использование **декоратора** в любом из заданий

```
1 usage  Stas Bekarev
def func_decorator(input_func):
    Stas Bekarev
    def output_func(*args):
        print("Counting...")
        res, n = input_func(*args)
        if n == 501:
            print("Number of iterations exceeded")
        else:
            print("Success")
        return res, n
    return output_func
```

10. Обеспечить обработку конкретных классов исключений

```
    try:
        res, n = tailor_func(x, eps)
        print(f"Enter x: {x}")
        print(f"Enter eps: {eps}")
        print(f"MathFunc result = {math.log(1 + x)}"
        print(f"TailorFunc result = {res}")
        print(f"Number of iterations: {n}")
    except ValueError:
        print("Warning!!! Exception: ValueError")
```

1.

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. Результат получить в виде:

x	*n*	*F(x)*	*Math F(x)*	*eps*

Здесь x – значение аргумента, F(x) – значение функции, n – количество просуммированных членов ряда, Math F(x) – значение функции, вычисленное с помощью модуля math.

3.	$\ln(1+x)=\sum_{n=0}^{\infty}(-1)^{n-1}\frac{x^n}{n}=x-\frac{x^2}{2}+\frac{x^3}{3}+\ldots,\ \lvert x\rvert<1$

```
**********
MathFunc: ln(1 + x), |x| < 1
Enter x: 0.5
Enter eps: 0.001
Counting...
Success
Enter x: 0.5
Enter eps: 0.001
MathFunc result = 0.4054651081081644
TailorFunc result = 0.40531529017857143
Number of iterations: 8
**********
```

Корректная обработка ошибок ввода пользователя

```
********************
MathFunc: ln(1 + x), |x| < 1
Enter x: 1234
Input number is bounding between -1 and 1
Try again: qwer
This is not a float!
Try again: 0.789
Enter eps: -1
Input number is bounding between 0 and 1
Try again: 0
Counting...
Number of iterations exceeded
```

2.

	количество отрицательных чисел. Окончание цикла – ввод числа, большего 100
3.	Организовать цикл, принимающий целые числа и подсчитывающий количество положительных. Окончание – ввод 10
4	Организовать цикл, который принимает целые числа с клавиатуры и

```
********************
Use auto generate? [y/n]
n
Enter a sequence of numbers ended by 10 :
1
2
-4
-5
4
-4
10
Amount of positive number: 3
********************
```

```
********************
Use auto generate? [y/n]
y
How many numbers generate?
10
-9 0 7 0 0 -9 2 -3 2 -2
Amount of positive number: 3
********************
```

3.

3.	Определить, является ли введенная с клавиатуры строка шестнадцатеричным числом

```
**********
Enter your string: 123abDF03
Your string is HEX number
**********
```

```
**********
Enter your string: 123rw3420f
Your string is NOT HEX number
**********
```

4.

3.	а) определить количество слов в строке и вывести на экран все слова, количество букв у которых четное; б) найти самое короткое слово, которое начинается на 'а'; в) вывести повторяющиеся слова

```
**********
Use default string? [y/n]
y
Words amount: 55
Words with odd count of letters:
in mind as well as made feel very sleepy stupid pleasure of making be of up when suddenly Rabbit with pink eyes by
The shortest word begin at 'a': a
Repeated words:
she her as the and of a
**********
```

5.

3.	Найти максимальный по модулю элемент списка и сумму элементов списка расположенных до последнего положительного элемента

```
**********
Enter a size of array:
12
Use auto generation? [y/n]
y
Resulted list:
[4.0, -10.0, 5.0, 8.0, 1.0, -4.0, 1.0, 0.0, 1.0, 3.0, -2.0, -1.0]
Maximal absolute value: -10.0
Sum ending last positive number: 9.0
**********
```