Избранные главы информатики

Лабораторная работа №3

подготовил:
Макаров Алексей
студент группы 253503

Задание 1

Программа считает функцию sin(x) с помощью ряда Тейлора. Аргументы для синуса должны быть в пределе [-pi/2; pi/2]. Для форматирования входных данных была разработана функция сериализации.

```
def serialize_argument(argument: float) -> (float, bool):
    """Convert sin(x) argument to suitable format
    """
    argument = fmod(argument, 2 * pi)
    is_neg = False
    if argument < 0:
        argument += 2 * pi

    if 3 * pi / 2 > argument > pi / 2:
        argument = pi - argument
    elif 2 * pi > argument > 3 * pi / 2:
        argument = argument - 2 * pi

    if argument < 0:
        argument *= -1
        is_neg = True
    return argument, is_neg
```

Функция находящая сумму ряда

```
def my_sin(argument: float, eps: float):
    """
    Calculate sin(x) by Taylor sires.
    sin(x) = x - x^3/3! + ... x^(2n+1)/(2n+1)!
    :rtype: (int, float)
    """
    n = 0
    factorial = 1

    argument, is_neg = serialize_argument(argument)
    member = argument
    sum_ = 0
    while n <= 500 and fabs(member) > eps:
        sum_ += member
        member *= argument * argument
        factorial *= 2 * (n + 1) * (2 * n + 3)
        member /= -factorial
        n += 1
    if is_neg:
        sum_ *= -1
    return n, sum_
```

Сама программа выполняется в функции main()

```
    @infinity_program
    def main():
        """Perform Task1"""
        x, eps = input_arg_eps()
        print("| x | n | F(x) | math F(x) | eps ")
        print(x, *my_sin(x, eps), sin(x), eps)
```

В ней реализована функция считывания аргументов с клавиатуры, которая требует от пользователя правильный ввод. Все основные рабочие программы обернуты декоратором, который обеспечивает бесконечное выполнение программы, пока пользователь это хочет.

```
def infinity_program(func):
    def wrapper():
        while True:
            func()
            text = input("Do you want restart program? (y/n)\n")
            if text not in ('y', 'Y'):
                break

    return wrapper
```

Пример работы программа

```
/home/bass-n-cripp/university/253503_MAKAROV_16/IGI/LR3/.v
Input x: 5
Input eps: .00001
| x | n | F(x) | math F(x) | eps
5.0 3 -0.9358759318412018 -0.9589242746631385 1e-05
Do you want restart program? (y/n)
```

Задание 2

Организовать цикл, который принимает целые числа и вычисляет количество четных натуральных чисел. Окончание – ввод 0

```
@infinity_program
def main():
    """Perform Task2"""
    n = 0
    count = 0
    print("Start input integers")
    while (digit := input_type(int, )) != 0:
        if digit > 0 and digit % 2 == 0:
            count += 1
        n += 1
    print(f"Total count of numbers: {n}\n"
          f"Count of positive even numbers: {count}")
```

В задаче появляется функция input_type, которая обрабатывает ошибки неправильного формата.

```
def input_type(func, string=''):
    """"""
    while True:
        try:
            res = func(input(string))
        except ValueError as e:
            print(e.args[0])
            continue
        break
    return res
```

```
Start input integers.
0 is triggering stop input.
54
87
13
0
Total count of numbers: 3
Count of positive even numbers: 1
```

Задание 3

В строке, введенной с клавиатуры, подсчитать количество знаков пунктуации.

```
@infinity_program
def main():
    """Perform Task3"""
    text = input_or_generate_seq(generate_random_string)
    count = 0
    for i in text:
        if i in punctuation:
            count += 1
    print(''.join(text))
    print(f"Count of punctuation: {count}")
```

Здесь используется функция input_or_generate_seq, которая дает пользователю выбор: ввести аргументы самому или сгенерировать случайно. На вход она требует функцию генератор и тип вводимых аргументов. Ниже описана сама функция и генераторы.

```
def generate_random_float_list(count: int, start=-100, end=100):
    """Generate list  with fandom float in given range."""
    return [random.uniform(start, end) for i in range(count)]
```

```python
def generate_random_string(count: int):
    """Generate random string. Contains printable ASCII symbols."""
    chars = string.printable
    return ''.join([random.choice(chars) for i in range(count)])


def input_or_generate_seq(generator_func, type_: type = str):
    """Generate random sequence or  user input sequence by yourself.
    Required generator function and type of
     """
    n = input_type(int, "Input sequence length: ")
    flag = input("Do you want input sequence? (y/n)  ")
    sequence = []
    if flag in ('y', "Y"):
        for i in range(n):
            sequence.append(input_type(type_))
    else:
        sequence.extend(generator_func(n))
    return sequence
```

Результат работы программы:

```
Input sequence length: 10
Do you want input sequence? (y/n)  n
BhVXge!kAY
Count of punctuation: 1
```

Задание 4

а) определить число слов, заканчивающихся на согласную;
б) найти среднюю длину слов в строке, округлив результат до
целого числа, и вывести все слова, которые имеют такую длину,
или сообщение «Слов длиной n символов в строке нет»;
в) вывести каждое седьмое слово

Код задачи выполнен следующим образом. За каждую подзадачу отвечают отдельные функции.

```python
@infinity_program
def main():
    """Perform Task4"""
    global base_text
    word_list = ''.join([i for i in base_text if i not in
punctuation]).split(' ')
    sub_task_a(word_list)
    sub_task_b(word_list)
    sub_task_c(word_list)
```

```python
def sub_task_a(words: list[str]):
    """
    Find count of words, that start from constraints
    """
    print("\nSub task A")
    the_vowels = ["a", "e", "u", "i", "o"]
    N = 0
    for i in words:
        if i[0].lower() not in the_vowels:
            N += 1
    print(f"Count of words, that start from constraints: {N}")


def sub_task_b(words: list[str]):
    """Find middle length of words. Then print words, that length equal
middle length"""
    print("\nSub task B")
    N = 0
    ans = []
    for i in words:
        N += len(i)
    mid_len = N // len(words)
    print(f"Middle len of the words: {mid_len}")
    for i in words:
        if len(i) == mid_len:
            ans.append(i)

    if len(ans) == 0:
        print(f"Words with len {mid_len} don't exist in a text")
    else:
        for i in ans:
            print(i)


def sub_task_c(words):
    """Print each 7th word in text"""
    print("\nSub task C")
    for i in range(len(words), 7):
        print(words[i])
```

Результат программы

```
Sub task A
Count of words, that start from constraints: 41

Sub task B
Middle len of the words: 4
mind
well
made
when
with
pink
eyes

Sub task C
So
mind
the
and
a
getting
suddenly
ran
```

Задание 5)

Найти сумму неотрицательных элементов и произведение элементов списка, расположенных между максимальным и минимальным по модулю элементами

Функция решения задачи:

```python
def solve(number_seq):
    """Find indexes first maximum and minimum element.
    Program find sum all positive numbers and mul all numbers  between max
and min."""
    max_ = min_ = 0
    for i in range(1, len(number_seq)):
        if abs(number_seq[i]) > abs(number_seq[max_]):
            max_ = i
        if abs(number_seq[i]) < abs(number_seq[min_]):
            min_ = i
    left_border = min(min_, max_)
    right_border = max(min_, max_)
    mul_res = 1
    sum_res = 0
    for i in number_seq[left_border:right_border + 1]:
```

```
        if i > 0:
            sum_res += i
        mul_res *= i
    print(*number_seq)
    print([sum_res, mul_res])
```

Результат работы программы

```
Input sequence length: 5
Do you want input sequence? (y/n)  т
26.485059679438066 18.651530732302064 67.77477872086169 38.85569783542297 14.79715980500849
[121.42763636129315, 38967.37810538417]
```