НИУ «МЭИ»

Кафедра «Релейной защиты и автоматизации энергосистем»

Вычислительные комплексы в электроэнергетике

ЛАБОРАТОРНАЯ РАБОТА №1

«*Основы языка Python»*

|  |  |
| --- | --- |
| Группа: | Э-12-20 |
| Вариант: | 7 |
| ФИО студента: | Карамянц А.И. |
| ФИО преподавателя: | Данилов С.А. |

Москва, 2024

**Цель работы:** ознакомиться с основами языка Python и выполнить на нем несколько инструкций.

**Предварительный отчет.**

**Особенности языка Python:**

* Простой
* Легкий в освоении
* Свободный и открытый
* Язык высокого уровня
* Портируемый
* Интерпретируемый
* Объектно-ориентированный
* Расширяемый
* Встраиваемый
* Обширные библиотеки

В данной лабораторной работе используются так называемые коллекции: списки, кортежи, множества и словари.

*Список*- это изменяемая последовательность элементов. Каждый ее элемент имеет место в последовательности, то есть имеет свой индекс. Список позволяет изменять элементы, добавлять и переставлять их. Список можно ввести вручную:

Name=[1, 2, 3.3, ‘Hello, world’, 5]

либо выделить под него место в памяти:

Name=[]

а затем заполнить его с помощью команды «append()»:

Name.append()

Отличительной чертой списков являются квадратные скобки, в которых записаны элементы

*Кортеж*- это неизменяемая последовательность элементов. Его элементы подобно списку имеют свои индексы, то есть можно обратиться к определенному элементу кортежа, но элементы нельзя заменить или переставить.

*Кортеж можно ввести вручную*:

*Name=(1, 2, 6 ,4, 3, 7)*

Или его можно создать из уже существующей последовательности (например списка):

Name=tuple(name\_spisok)

Отличительной чертой кортежа являются круглые скобки.

*Множества*- это совокупность уникальных чисел. То есть в нем не может быть повторяющихся элементов. Если задать в множество повторяющиеся элементы, то оно оставит лишь первый встретившийся уникальный элемент, а остальные исключит. В отличие от списка и кортежа элементы множества не имеют индексов.

Множество можно ввести вручную:

Name={1, 2, 3, 4, 5}

Или же выделить под него место в памяти

Name=set()

А затем заполнить его с помощью команды «add»:

Name.add()

Отличительной чертой множества являются фигурные скобки.

*Словари*- это неупорядоченные коллекции произвольных объектов с доступом по ключу. То есть в словарь записываются пары элементов в последовательности «ключ:значение».

Словарь можно задать вручную:

Name={ключ1:значение1, ключ2:значение2,…}

Или же выделить под него место в памяти:

Name=dict()

А затем заполнить его.

Отличительной чертой словаря являются фигурные скобки и форма записи его элементов.

**Отчет**

*Задача 1.*

Используя цикл while, сгенерировать список с диапазоном значений от 0 до 15. Вывести результат.

*Код программы:*

# 1. Генерация списка с использованием цикла while

result = [] #Создаем пустой список

i = 0

while i < 16:

    result.append(i) #Добавляем значения в список

    i += 1 #Увеличиваем переменную i на каждой итерации

print("Сгенерированный список:", result)

*Результаты работы кода:*

*Сгенерированный список: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]*

*Задача 2.*

На основе списка создать множество значений от 0 до 10, использовать при этом цикл break. Вывести результат.

*Код программы:*

# 2. Создание множества с использованием цикла while и break

set\_result = set() #Создаем пустое множество с помощью функции set

for num in result:

    if num > 10:

        break #при достижении значения больше 10 используем оператор break чтобы прервать цикл

    set\_result.add(num) #добавляем значения из списка во множество

print("Множество значений до 10:", set\_result)

*Результаты работы кода:*

Множество значений до 10: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

*Задача 3.*

Удалить из списка все значения меньше 11, используя цикл while. Вывести результат.

*Код программы:*

# 3. Удаление значений из списка с использованием цикла while

i = 0

while i < len(result): #используем len для нахождения длины списка

    if result[i] < 11:

        result.pop(i) #c помощью pop удаляем элемент

    else:

        i += 1 #увеличиваем значение на 1

print("Список после удаления значений меньше 11:", result)

*Результаты работы кода:*

Список после удаления значений меньше 11: [11, 12, 13, 14, 15]

*Задача 4.*

Очищенный список заполнить элементами кортежа, увеличив на 39 все элементы больше или равные 50 и увеличив в 3 раза остальные. Вывести результат. Элементы кортежа добавить в конец списка в обратном порядке. Вывести результат.

*Код программы:*

# 4. Модификация списка и добавление элементов кортежа в обратном порядке

for i in range(len(result)): #используем len для нахождения длины списка

    if result[i] >= 50: #если элемент больше или равен 50

        result[i] = (result[i] + 39,) #заменяем его на кортеж увеличенный на 39

    else: #если условие не выполняется

        result[i] = (result[i] \* 3,) #заменяем его на кортеж увеличенный в 3 раза

result.extend(tuple(result[::-1])) #c помощью extend добавляем кортеж в обратном порядке

print("Список с добавленными элементами кортежа:", result)

*Результаты работы кода:*

Список с добавленными элементами кортежа: [(33,), (36,), (39,), (42,), (45,), (45,), (42,), (39,), (36,), (33,)]

*Задача 5.*

Написать функцию, на вход которой подается два списка в одном значение длины, а другом ширины. Найти площади круга, описывающего каждый из прямоугольников и записать в кортеж. Найти среднее площадей и удалить все значения из списка больше неё на 10%. Из полученных значений найти произведение и поделить его на длину входного списка и вывести ответ. Итоговый ответ вернуть и вывести на экран. При выполнении задания использовать функцию filter().

*Код программы:*

# 5. Функция для расчета площадей, фильтрации и вычисления результата

import math #импортируем модуль math чтобы использовать математические функции

def calculate\_circle\_area(length\_list, width\_list): #функция которая принимает два списка

    areas = [] #создаем пустой список куда будем записывать площади

    for length, width in zip(length\_list, width\_list): #перебираем значения списков с помощью цикла for

        radius = math.sqrt((length/2)\*\*2 + (width/2)\*\*2) #находим радиус круга из сторон прямоугольника

        area = math.pi \* radius\*\*2 #находим площадь круга

        areas.append(area) #добавляем значения площадей в список

    average\_area = sum(areas) / len(areas) #считаем среднюю площадь кругов как сумму всех площадей, деленную на количество площадей

    filtered\_areas = list(filter(lambda x: x <= 0.9 \* average\_area, areas)) #используем функцию filter для фильтрации значений в списке areas

    product = 1

    for num in filtered\_areas:

        product \*= num #перемножаем все значения в отфильтрованном списке

    result = product / len(length\_list) #делим это произведение на длину входного списка

    return result #выходим из функции

length\_list = [4, 6, 3, 5]

width\_list = [3, 5, 2, 4]

answer = calculate\_circle\_area(length\_list, width\_list) #передаем в функцию списки

print(answer) #выводим результат

*Результаты работы кода:*

*50.11908484928189*

*Задача 6.*

Написать функцию, на вход которой подается наименьшее, наибольшее значение и шаг. По этим параметрам сгенерировать заполненный список. Посчитать количество чисел кратных 7 и отнять от него медиану всех чисел списка. Если полученный результат отрицательный необходимо зеркально отобразить список, если положительный, то создать копию старого списка и добавить в его начало полученное число. Итоговый список вернуть и вывести на экран. При выполнении задания использовать функцию filter().

*Код программы:*

# 6. Функция для генерации списка, фильтрации и обработки

def generate\_and\_process(min\_val, max\_val, step): #создаем функцию которая принимает два значения и шаг

    numbers = list(range(min\_val, max\_val, step)) #создаем список чисел с заданным диапозоном и шагом

    divisible\_by\_7 = len(list(filter(lambda x: x % 7 == 0, numbers))) #находим количество чисел кратных 7 c помощью filter

    median = sorted(numbers)[len(numbers) // 2] #находим их медиану

    result = divisible\_by\_7 - median #вычисляем разность

    if result < 0: #если разность отрицательна

        return numbers[::-1] #возвращаем список, отраженный в обратном порядке

    else: #если уловие не выполнилось

        new\_list = numbers.copy() #создаем копию

        new\_list.insert(0, result) #с помощью insert помещаем число в начало списка

        return new\_list #возвращаем скопированный список

final\_result = generate\_and\_process(5, 35, 3) #передаем в функцию значения

print("Итоговый список:", final\_result) #выводим результат

*Результаты работы кода:*

Итоговый список: [32, 29, 26, 23, 20, 17, 14, 11, 8, 5]

*Задача 7.*

Написать функцию поиска и замены. На вход функции подается исходная строка, часть строки, которую нужно заменить, то, на что нужно заменить.

*Код программы:*

# 7. Функция поиска и замены

def find\_and\_replace(input\_string, search\_str, replace\_str): #создаем функцию которая принимает исходную строку, часть строки для замены и то, на что нужно заменить

    return input\_string.replace(search\_str, replace\_str) #пользуем метод replace для замены указанной части строки на указанное значение

original\_string = "Роналду лучший футболист мира" #вводим строку которую будем изменять

new\_string = find\_and\_replace(original\_string, "Роналду", "Месси") #передаем в функцию значения

print("Измененная строка:", new\_string) #выводим результат

*Результаты работы кода:*

Измененная строка: Месси лучший футболист мира

**Вывод:** в результате работы с помощью языка Python было решено 7 задач. В результате каждого решения были получены ожидаемые результаты Пояснения к работе кода представлены в виде построчного комментирования.