Задание 1

Работаю в Google Colab ♥

Задание 2

Написать программу, которая вычисляет площадь фигуры, параметры которой подаются на вход. Фигуры, которые подаются на вход: треугольник, прямоугольник, круг. Результатом работы является словарь, где ключ – это название фигуры, а значение – это площадь.

```
import math

def calculate_area(shape, params):
    if shape == 'треугольник':
        base, height = params
        area = 0.5 * base * height
    elif shape == 'прямоугольник':
        length, width = params
        area = length * width
    elif shape == 'круг':
        radius = params
        area = math.pi * radius ** 2
    else:
        raise ValueError("Неизвестная фигура")
    return area

def main():
    shapes = {
        'треугольник': (5, 10),
        'прямоугольник': (4, 6),
        'круг': (3)
    }

    areas = {}
    for shape, params in shapes.items():
        areas[shape] = calculate_area(shape, params)

    print(areas)

main()
```

```
{'треугольник': 25.0, 'прямоугольник': 24, 'круг': 28.274333882308138}
```

Задание 3

Написать программу, которая на вход получает два числа и операцию, которую к ним нужно применить. Должны быть реализованы следующие операции: +, -, /, //, abs – модуль, pow или ** – возведение в степень.

```
def calculate(num1, num2, operation):
    if operation == '+':
        return num1 + num2
    elif operation == '-':
        return num1 - num2
    elif operation == '/':
        if num2 == 0:
            return "Ошибка: деление на ноль"
        return num1 / num2
    elif operation == '//':
        if num2 == 0:
            return "Ошибка: деление на ноль"
        return num1 // num2
    elif operation == 'abs':
        return abs(num1)
    elif operation == 'pow' or operation == '**':
        return num1 ** num2
    else:
        return "Ошибка: неизвестная операция"

def main():
    num1 = float(input("Введите первое число: "))
    num2 = float(input("Введите второе число: "))
    operation = input("Введите операцию (+, -, /, //, abs, pow или
**): ")

    result = calculate(num1, num2, operation)
    print(f"Результат: {result}")

main()
```

```
Введите первое число: 1
Введите второе число: 4
Введите операцию (+, -, /, //, abs, pow или **): pow
Результат: 1.0
```

Задание 4

Напишите программу, которая считывает с консоли числа (по одному в строке) до тех пор, пока сумма введённых чисел не будет равна 0 и после этого выводит сумму квадратов всех считанных чисел.

```
def main():
    numbers = []
    total_sum = 0

    while True:
        try:
            num = float(input("Введите число: "))
```

```
            numbers.append(num)
            total_sum += num

            if total_sum == 0:
                break
        except ValueError:
            print("Ошибка: введено некорректное значение. Пожалуйста,
введите число.")

    sum_of_squares = sum(x ** 2 for x in numbers)
    print(f"Сумма квадратов всех считанных чисел: {sum_of_squares}")

main()
```

```
Введите число: 5
Введите число: 6
Введите число: -6
Введите число: -5
Сумма квадратов всех считанных чисел: 122.0
```

Задание 5

Напишите программу, которая выводит последовательность чисел, длинною N, где каждое число повторяется столько раз, чему оно равно. На вход программе передаётся неотрицательное целое число N. Например, если N = 7, то программа должна вывести 1 2 2 3 3 3 4. Вывод элементов списка через пробел – print(*list).

```
def generate_sequence(N):
    sequence = []
    current_number = 1
    while len(sequence) < N:
        for _ in range(current_number):
            if len(sequence) < N:
                sequence.append(current_number)
        current_number += 1
    return sequence

def main():
    N = int(input("Введите неотрицательное целое число N: "))
    if N < 0:
        print("Ошибка: N должно быть неотрицательным числом.")
        return

    sequence = generate_sequence(N)
    print(*sequence)

main()
```

```
Введите неотрицательное целое число N: 100
1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 6 6 6 6 6 6 7 7 7 7 7 7 7 8 8 8 8 8 8 8
```

```
8 9 9 9 9 9 9 9 9 9 10 10 10 10 10 10 10 10 10 10 11 11 11 11 11 11 11
11 11 11 11 12 12 12 12 12 12 12 12 12 12 12 12 13 13 13 13 13 13 13
13 13 13 13 13 13 14 14 14 14 14 14 14 14 14
```

Задание 6

Даны два списка: A = [1, 2, 3, 4, 2, 1, 3, 4, 5, 6, 5, 4, 3, 2] B = [‘a’, ’b’, ’c’, ’c’, ’c’, ’b’, ’a’, ’c’, ’a’, ’a’, ’b’, ’c’, ’b’, ’a’] Создать словарь, в котором ключи – это содержимое списка B, а значения для ключей словаря – это сумма всех элементов списка A в соответствии с буквой, содержащийся на той же позиции в списке B. Пример результата программы: {‘a’ : 10, ‘b’ : 15, ‘c’ : 6}.

```
A = [1, 2, 3, 4, 2, 1, 3, 4, 5, 6, 5, 4, 3, 2]
B = ['a', 'b', 'c', 'c', 'c', 'b', 'a', 'c', 'a', 'a', 'b', 'c', 'b',
'a']

result = {}

for a, b in zip(A, B):
    if b in result:
        result[b] += a
    else:
        result[b] = a

print(result)
```

```
{'a': 17, 'b': 11, 'c': 17}
```

Задание 7

Скачать и загрузить данные о стоимости домов в калифорнии, используя библиотеку sklearn.

```
from sklearn.datasets import fetch_california_housing

data = fetch_california_housing(as_frame=True)
```

Задание 8

Использовать метод info().

```
import pandas

_data = pandas.DataFrame(data.data, columns=data.feature_names)
_data['Target'] = data.target

_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   MedInc      20640 non-null  float64
 1   HouseAge    20640 non-null  float64
 2   AveRooms    20640 non-null  float64
 3   AveBedrms   20640 non-null  float64
 4   Population  20640 non-null  float64
 5   AveOccup    20640 non-null  float64
 6   Latitude    20640 non-null  float64
 7   Longitude   20640 non-null  float64
 8   Target      20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
```

Задание 9

Узнать, есть ли пропущенные значения, используя isna().sum().

```
missing_values = _data.isna().sum()
print(missing_values)
```

```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
Target        0
dtype: int64
```

Задание 10

Вывести записи, где средний возраст домов в районе более 50 лет и население более 2500 человек, используя метод loc().

```
filtered_data = _data.loc[(_data['HouseAge'] > 50) &
(_data['Population'] > 2500)]
print(filtered_data)
```

```
       MedInc  HouseAge  AveRooms  AveBedrms  Population
AveOccup  \
460    1.4012      52.0  3.105714   1.060000      3337.0    9.534286

4131   3.5349      52.0  4.646119   1.047945      2589.0    5.910959
```

```
4440   2.6806      52.0  4.806283   1.057592      3062.0    4.007853

5986   1.8750      52.0  4.500000   1.206349      2688.0   21.333333

7369   3.1901      52.0  4.730942   1.017937      3731.0    4.182735

8227   2.3305      52.0  3.488860   1.170380      3018.0    3.955439

13034  6.1359      52.0  8.275862   1.517241      6675.0  230.172414

15634  1.8295      52.0  2.628169   1.053521      2957.0    4.164789

15652  0.9000      52.0  2.237474   1.053535      3260.0    2.237474

15657  2.5166      52.0  2.839075   1.184049      3436.0    1.621520

15659  1.7240      52.0  2.278566   1.082348      4518.0    1.780142

15795  2.5755      52.0  3.402576   1.058776      2619.0    2.108696

15868  2.8135      52.0  4.584329   1.041169      2987.0    3.966799
```

	Latitude	Longitude	Target
460	37.87	-122.26	1.75000
4131	34.13	-118.20	1.93600
4440	34.08	-118.21	1.53000
5986	34.10	-117.71	2.12500
7369	33.97	-118.21	1.67600
8227	33.78	-118.20	1.62500
13034	38.69	-121.15	2.25000
15634	37.80	-122.41	2.43800
15652	37.80	-122.41	5.00001
15657	37.79	-122.41	2.75000
15659	37.79	-122.41	2.25000
15795	37.77	-122.42	3.25000
15868	37.76	-122.41	2.60300

Задание 11

Узнать максимальное и минимальное значения медианной стоимости дома.

```python
max = _data['Target'].max()
min = _data['Target'].min()
print(max, min)
```

```
5.00001 0.14999
```

Задание 12

Используя метод apply(), вывести на экран название признака и его среднее значение.

```
mean_values = _data.apply(lambda x: x.mean())
for feature, mean_value in mean_values.items():
    print(f"Признак: {feature}, Среднее значение: {mean_value}")
```

```
Признак: MedInc, Среднее значение: 3.8706710029069766
Признак: HouseAge, Среднее значение: 28.639486434108527
Признак: AveRooms, Среднее значение: 5.428999742190376
Признак: AveBedrms, Среднее значение: 1.096675149606208
Признак: Population, Среднее значение: 1425.4767441860465
Признак: AveOccup, Среднее значение: 3.0706551594363742
Признак: Latitude, Среднее значение: 35.63186143410853
Признак: Longitude, Среднее значение: -119.56970445736432
Признак: Target, Среднее значение: 2.068558169089147
```

Задание 1.*

Дан текст на английском языке. Необходимо закодировать его с помощью азбуки Морзе, где каждой букве соответствует последовательность точек и тире. Например, буква «g» превратится в строку «--.». В переменной morze для удобства хранится словарь соответствия латинских букв коду Морзе. morze = {'a': '.-', 'b': '-…', 'c': '-.-.', 'd': '-..', 'e': '.', 'f': '..-.', 'g': '--.', 'h': '….', 'i': '..', 'j': '.---', 'k': '-.-', 'l': '.-..', 'm': '--', 'n': '-.', 'o': '---', 'p': '.--.', 'q': '--.-', 'r': '.-.', 's': '…', 't': '-', 'u': '..-', 'v': '…-', 'w': '.--', 'x': '-..-', 'y': '-.--', 'z': '--..'} На входе: В одной строке вам дан текст, который состоит из латинских букв и пробелов. На выходе: Выведите каждое слово исходного текста, закодированное азбукой Морзе. Количество строк в ответе должно совпадать с количеством слов в исходном тексте. Между закодированными буквами ставится ровно один пробел. Например, слово «Help» превратится в «.... . .-.. .--.». Строчные и заглавные буквы кодируются одинаково. Например: Ignition sequence start Перевод .. --. -. .. - .. --- -. … . --.- ..- . -. -.-. . … - .- .-. -

```
# Словарь соответствия латинских букв коду Морзе
morze = {
    'a': '.-', 'b': '-...', 'c': '-.-.', 'd': '-..',
    'e': '.', 'f': '..-.', 'g': '--.', 'h': '....',
    'i': '..', 'j': '.---', 'k': '-.-', 'l': '.-..',
    'm': '--', 'n': '-.', 'o': '---', 'p': '.--.',
    'q': '--.-', 'r': '.-.', 's': '...', 't': '-',
    'u': '..-', 'v': '...-', 'w': '.--', 'x': '-..-',
    'y': '-.--', 'z': '--..'
}

def encode_to_morse(text):
    encoded_text = []
    for char in text.lower():
        if char in morze:
            encoded_text.append(morze[char])
```

```
        elif char == ' ':
            encoded_text.append('\n')
    return ''.join(encoded_text)

input_text = str(input("Введите текст: "))
encoded_text = encode_to_morse(input_text)
print(f"Закодированный текст:\n{encoded_text}")
```

```
Введите текст: Ignition sequence start
Закодированный текст:
..--.-...-.----.
.....-...-.-.-..
...-.-.-.-
```

Задание 2.*

В некотором городе открывается новая служба по доставке электронных писем. Необходимо наладить систему регистрации новых пользователей. Регистрация должна работать следующим образом: если новый пользователь хочет зарегистрироваться на сайте, то он должен послать системе запрос name со своим именем. Система должна определить, существует ли уже такое имя в базе данных. Если такого имени не существует, то оно заносится в базу данных системы и пользователю возвращается ответ "OK", подтверждающий успешную регистрацию. А если пользователь с таким именем уже существует, то система должна сформировать новое имя и выдать его пользователю в качестве подсказки, при этом сама подсказка также добавляется в базу данных. Новое имя формируется следующим образом: к name последовательно приписываются числа, начиная с 1 (name1, name2 и так далее), и среди них находят такое наименьшее i, что namei еще не содержится в системе. Входные данные В первой строке входных данных задано число n (1 ≤ n ≤ 100000). Следующие n строк содержат запросы к системе. Каждый запрос представляет собой непустую строку длиной не более 32 символов, состоящую только из строчных букв латинского алфавита. Выходные данные В выходных данных должно содержаться n строк – ответы системы на запросы: "OK" в случае успешной регистрации, или подсказка с новым именем, если запрашиваемое уже занято.

Данные для проверки:

Вход:

3

b

b

b

Выход:

OK

b1

b2

Вход:

10

bhnqaptmp

bhnqaptmp

bhnqaptmp

bhnqaptmp

bhnqaptmp

bhnqaptmp

bhnqaptmp

bhnqaptmp

bhnqaptmp

bhnqaptmp

Выход:

OK

bhnqaptmp1

bhnqaptmp2

bhnqaptmp3

bhnqaptmp4

bhnqaptmp5

bhnqaptmp6

bhnqaptmp7

bhnqaptmp8

bhnqaptmp9

Вход:

10

fpqhfouqdldravpjttarh

fpqhfouqdldravpjttarh

fpqhfouqdldravpjttarh

fpqhfouqdldravpjttarh

fpqhfouqdldravpjttarh

fpqhfouqdldravpjttarh

jmvlplnrmba

fpqhfouqdldravpjttarh

jmvlplnrmba

fpqhfouqdldravpjttarh

Выход:

OK

fpqhfouqdldravpjttarh1

fpqhfouqdldravpjttarh2

fpqhfouqdldravpjttarh3

fpqhfouqdldravpjttarh4

fpqhfouqdldravpjttarh5

OK

fpqhfouqdldravpjttarh6

jmvlplnrmba1

fpqhfouqdldravpjttarh7

```
def register_user(user_database, request):
    if request not in user_database:
        user_database[request] = 1
        return "OK"
    else:
        count = user_database[request]
        new_name = f"{request}{count}"
        while new_name in user_database:
            count += 1
            new_name = f"{request}{count}"
        user_database[new_name] = 1
        user_database[request] = count + 1
        return new_name

def main():
    user_database = {}
    n = int(input("Введите количество запросов: "))
```

```
    for _ in range(n):
        request = input("Введите имя для регистрации: ")
        result = register_user(user_database, request)
        print(result)

main()
```

```
Введите количество запросов: 3
Введите имя для регистрации: b
OK
Введите имя для регистрации: b
b1
Введите имя для регистрации: b
b2
```

Задание 3.*

Необходимо создать программу обработки запросов пользователей к файловой системе компьютера. Над каждым файлом можно производить следующие действия: запись – w ("write"), чтение – r ("read"), запуск – x ("execute"). Входные данные На вход программе подаются следующие параметры: число n – количество файлов в файловой системе. В следующих n строках содержится информация с именами файлов и допустимыми действиями (w, x, r), разделенных пробелами. Далее идет число m – количество запросов к файлам вида «операция файл» (обозначение операции: "write", "read", "execute"). Выходные данные Для каждого допустимого запроса программа должна возвращать OK, для недопустимого – Access denied.

Данные для проверки:

Вход:

python.exe x

book.txt r w

notebook.exe r w x

5

read python.exe

read book.txt

write notebook.exe

execute notebook.exe

write book.txt

Вывод:

Access denied

OK

OK

OK

OK

Вход:

root.html r w x

main.py x

login.txt w r

4

read root.html

write main.py

execute main.py

execute login.txt

Вывод:

OK

Access denied

OK

Access denied

Вход:

2

1.txt

2.txt

2

write 1.txt

execute 2.txt

Выход:

Access denied

Access denied

```
def process_request(file_info, request, operation_map):
    operation, file = request.split()
    if file in file_info and operation_map[operation] in
file_info[file]:
        return "OK"
    else:
        return "Access denied"

def main():
    operation_map = {
        "execute": "x",
        "read": "r",
        "write": "w"
    }

    n = int(input("Введите количество файлов: "))
    file_info = {}

    for _ in range(n):
        file_data = input("Введите имя файла и допустимые действия:
").split()
        file_name = file_data[0]
        actions = set(file_data[1:])
        file_info[file_name] = actions

    m = int(input("Введите количество запросов: "))

    for _ in range(m):
        request = input("Введите запрос (операция файл): ")
        result = process_request(file_info, request, operation_map)
        print(result)

main()
```

```
Введите количество файлов: 3
Введите имя файла и допустимые действия: python.exe x
Введите имя файла и допустимые действия: book.txt r w
Введите имя файла и допустимые действия: notebook.exe r w x
Введите количество запросов: 5
Введите запрос (операция файл): read python.exe
Access denied
Введите запрос (операция файл): read book.txt
OK
Введите запрос (операция файл): write notebook.exe
OK
Введите запрос (операция файл): execute notebook.exe
OK
Введите запрос (операция файл): write book.txt
OK
```