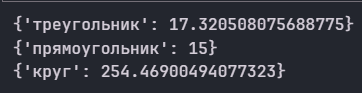
Практическая работа №1

1. 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.



def calculate\_area(shape, \*\*params):

areas = {}

if shape == 'треугольник':

a, b, c = params['a'], params['b'], params['c']

s = (a + b + c) / 2

area = (s \* (s - a) \* (s - b) \* (s - c)) \*\* 0.5

areas['треугольник'] = area

elif shape == 'прямоугольник':

area = params['width'] \* params['height']

areas['прямоугольник'] = area

elif shape == 'круг':

import math

area = math.pi \* params['radius'] \*\* 2

areas['круг'] = area

return areas

# Примеры использования:

print(calculate\_area('треугольник', a=5, b=7, c=8))

print(calculate\_area('прямоугольник', width=3, height=5))

print(calculate\_area('круг', radius=9))

1. 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(a, b, operation):

if operation == '+':

return a + b

elif operation == '-':

return a - b

elif operation == '/':

return a / b

elif operation == '//':

return a // b

elif operation == 'abs':

return abs(a), abs(b)

elif operation in ('pow', '\*\*'):

return a \*\* b

else:

return "Некорректная операция"

a = float(input("Введите первое число: "))

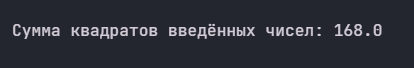
b = float(input("Введите второе число: "))

operation = input("Введите операцию (+, -, /, //, abs, pow, \*\*): ")

result = calculate(a, b, operation)

print("Результат:", result)

1. 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



numbers = []

total = 0

while True:

    num = int(input("Введите число: "))

    numbers.append(num)

    total += num

    if total == 0:

        break

sum\_of\_squares = sum(x\*\*2 for x in numbers)

print("Сумма квадратов:", sum\_of\_squares)

1. 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).



N = int(input("Введите неотрицательное целое число N: "))

sequence = []

num = 1

while len(sequence) < N:

sequence.extend([num] \* num)

num += 1

print(\*sequence[:N])

1. Даны два списка: А = [1, 2, 3, 4, 2, 1, 3, 4, 5, 6, 5, 4, 3, 2] В = [‘a’, ’b’, ’c’, ’c’, ’c’, ’b’, ’a’, ’c’, ’a’, ’a’, ’b’, ’c’, ’b’, ’a’] Создать словарь, в котором ключи – это содержимое списка В, а значения для ключей словаря – это сумма всех элементов списка А в соответствии с буквой, содержащийся на той же позиции в списке В. 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']

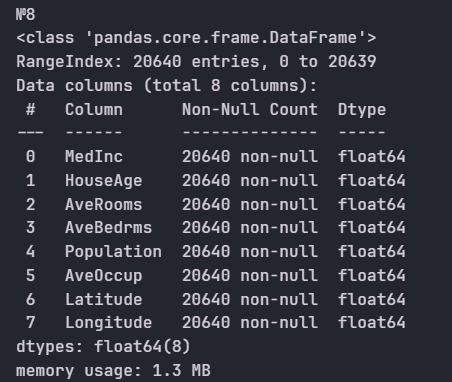
result\_dict = {}

for value, key in zip(A, B):

result\_dict[key] = result\_dict.get(key, 0) + value

print(result\_dict)

1. Использовать метод info().

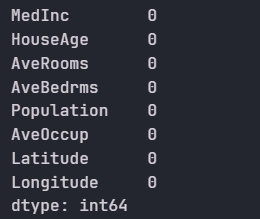


from sklearn.datasets import fetch\_california\_housing

data = fetch\_california\_housing(as\_frame=True)

data.data.info()

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

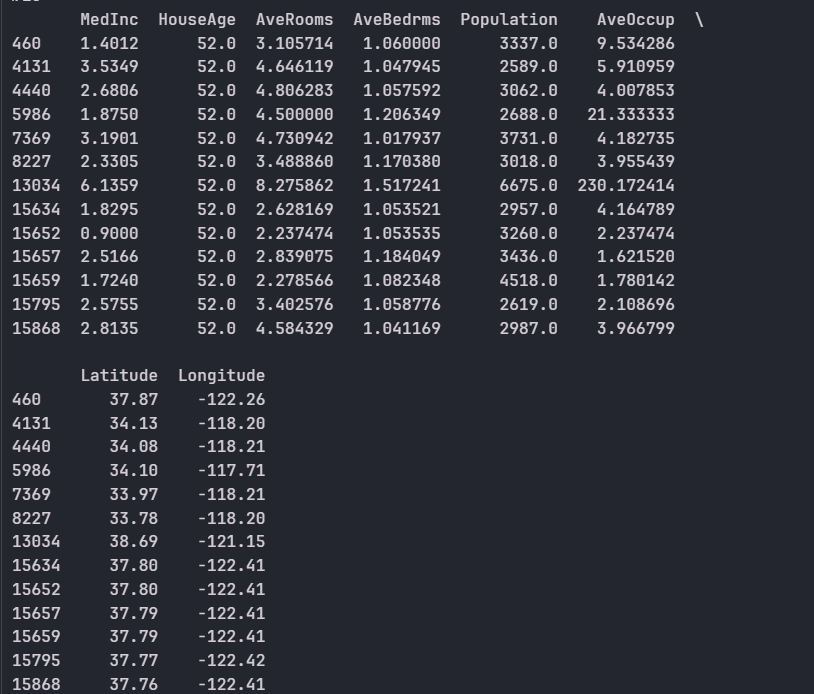


from sklearn.datasets import fetch\_california\_housing

data = fetch\_california\_housing(as\_frame=True)

print(data.data.isna().sum())

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



from sklearn.datasets import fetch\_california\_housing

filtered = data.data.loc[(data.data['HouseAge'] > 50) & (data.data['Population'] > 2500)]

print(filtered)

1. Узнать максимальное и минимальное значения медианной стоимости дома.



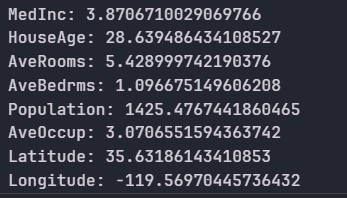
from sklearn.datasets import fetch\_california\_housing

data = fetch\_california\_housing(as\_frame=True)

print("Максимальная стоимость:", data.target.max())

print("Минимальная стоимость:", data.target.min())

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



from sklearn.datasets import fetch\_california\_housing

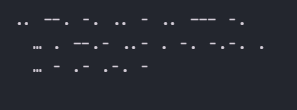
data = fetch\_california\_housing(as\_frame=True)

means = data.data.apply('mean')

for feature, mean\_value in means.items():

print(f"{feature}: {mean\_value}")

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 .. --. -. .. - .. --- -. … . --.- ..- . -. -.-. . … - .- .-. –



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\_morse(text):

text = text.lower()

encoded = []

for char in text:

if char in morze:

encoded.append(morze[char])

elif char == ' ':

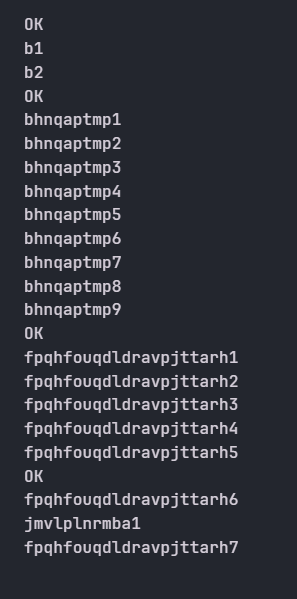
encoded.append(' \n ')

return ' '.join(encoded)

e\_text = str(input("Введите текст на английском: "))

print(encode\_morse(e\_text))

1. 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 "ОК", 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.



def run\_block(n: int):

db = {} # имя -> следующий суффикс

for i in range(n):

name = input(f"({i+1}/{n}) Логин: ").strip()

if name in db:

k = db[name]

new\_name = f"{name}{k}"

print(new\_name)

db[name] = k + 1

db[new\_name] = 1

else:

print("OK")

db[name] = 1

def main():

while True:

line = input("Сколько логинов? (Enter — выход): ").strip()

if not line:

break

try:

n = int(line)

except ValueError:

print("Введите целое число.")

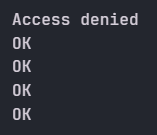
continue

run\_block(n)

if \_\_name\_\_ == "\_\_main\_\_":

main()

1. 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.



n = int(input())

permissions = {}

for \_ in range(n):

parts = input().split()

filename = parts[0]

perms = set(parts[1:])

permissions[filename] = perms

op\_map = {'write': 'w', 'read': 'r', 'execute': 'x'}

m = int(input())

for \_ in range(m):

op, filename = input().split()

if op\_map[op] in permissions.get(filename, set()):

print("OK")

else:

print("Access denied")