# Отчет о проделанной работе

## Отчет по файлу: task1.py

## Исходный код

1 from math import pi

2

3

4 def task() -> dict:

5 res = {}

6

7 num = input("""

8 Введите номер фигуры:

9 1. Треугольник

10 2. Прямоугольник

11 3. Круг

12 """)

13 shapes = {"1": "Треугольник", "2": "Прямоугольник", "3": "Круг"}

14

15 match num:

16 case "1":

17 b = float(input("Введите сторону: "))

18 h = float(input("Введите высоту: "))

19 s = 0.5 \* b \* h

20 res[shapes[num]] = s

21

22 case "2":

23 a = float(input("Введите сторону a: "))

24 b = float(input("Введите сторону b: "))

25 s = a \* b

26 res[shapes[num]] = s

27

28 case "3":

29 r = float(input("Введите радиус: "))

30 s = pi \* r\*\*2

31 res[shapes[num]] = s

32 case \_:

33 return None

34 return res

35

36

37 if \_\_name\_\_ == "main":

38 res = task()

39 if res is None:

40 print("Ошибка ввода")

41 else:

42 print(res)

43

44 if \_\_name\_\_ == "builtins":

45 import sys

46 from io import StringIO

47

48 test\_input = "2\n2\n3"

49 sys.stdin = StringIO(test\_input)

50

51 res = task()

52 if res is None:

53 print("Ошибка ввода")

54 else:

55 print(res)

## Результаты выполнения

Ввод:

2  
2  
3

Вывод программы:

Введите номер фигуры:  
 1. Треугольник  
 2. Прямоугольник  
 3. Круг  
 Введите сторону a: Введите сторону b: {'Прямоугольник': 6.0}

## Отчет по файлу: task2.py

## Исходный код

1 def task(a: float, b: float, operation: str):

2 str\_expr = str(a) + operation + str(b)

3

4 try:

5 x = eval(str\_expr)

6

7 except Exception:

8 print("При вводе допущена ошибка. Попробуйте еще раз.")

9 return x

10

11

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

13 print(task("5", "2", "\*"))

14

15 if \_\_name\_\_ == "builtins":

16 import sys

17 from io import StringIO

18

19 sys.stdin = StringIO()

20 print(task("5", "2", "\*"))

## Результаты выполнения

Вывод программы:

10

## Отчет по файлу: task3.py

## Исходный код

1 def task():

2 lst1 = []

3

4 while sum(lst1) != 0 or len(lst1) == 0:

5 lst1.append(int(input()))

6

7 lst1 = [x\*\*2 for x in lst1]

8 return sum(lst1)

9

10

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

12 print(task())

13

14

15 if \_\_name\_\_ == "builtins":

16 import sys

17 from io import StringIO

18

19 sys.stdin = StringIO("1\n2\n-3\n")

20 print(task())

## Результаты выполнения

Ввод:

1  
2  
-3

Вывод программы:

14

## Отчет по файлу: task4.py

## Исходный код

1 from functools import reduce

2

3

4 def task():

5 N = int(input())

6 return reduce(lambda x, y: x + y, [[i] \* i for i in range(N)])[:N]

7

8

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

10 print(\*task())

11

12

13 if \_\_name\_\_ == "builtins":

14 import sys

15 from io import StringIO

16

17 sys.stdin = StringIO("10\n")

18 print(\*task())

## Результаты выполнения

Ввод:

10

Вывод программы:

1 2 2 3 3 3 4 4 4 4

## Отчет по файлу: task5.py

## Исходный код

1 def task():

2 A = [1, 2, 3, 4, 2, 1, 3, 4, 5, 6, 5, 4, 3, 2]

3 B = ["a", "b", "c", "c", "c", "b", "a", "c", "a", "a", "b", "c", "b", "a"]

4

5 result = {}

6

7 for a, b in zip(A, B):

8 if b in result:

9 result[b] += a

10 else:

11 result[b] = a

12

13 return result

14

15

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

17 print(task())

18

19 if \_\_name\_\_ == "builtins":

20 import sys

21 from io import StringIO

22

23 sys.stdin = StringIO()

24 print(task())

## Результаты выполнения

Вывод программы:

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

## Отчет по файлу: task6-12.py

## Исходный код

1 import pandas as pd

2 from sklearn.datasets import fetch\_california\_housing

3

4

5 def task():

6 # Загрузка данных

7 df: pd.DataFrame = fetch\_california\_housing(as\_frame=True).frame

8

9 # 8. Использовать метод info()

10 print(df.info)

11

12 # 9. Узнать, есть ли пропущенные значения

13 print("\nКоличество пропущенных значений по каждому столбцу:")

14 print(df.isna().sum())

15

16 # 10. Вывести записи, где средний возраст домов в районе более 50 лет и население более 2500 человек

17 filtered\_df = df.loc[(df["HouseAge"] > 50) & (df["Population"] > 2500)]

18 print(

19 "\nЗаписи, где средний возраст домов в районе более 50 лет и население более 2500 человек:"

20 )

21 print(filtered\_df)

22

23 # 11. Узнать максимальное и минимальное значения медианной стоимости дома

24 median\_house\_value = df["MedHouseVal"]

25 print("\nМаксимальное значение медианной стоимости дома:", median\_house\_value.max())

26 print("Минимальное значение медианной стоимости дома:", median\_house\_value.min())

27

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

29 mean\_values = df.apply(lambda x: x.mean())

30 print("\nСредние значения признаков:")

31 print(mean\_values)

32

33

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

35 task()

36

37 if \_\_name\_\_ == "builtins":

38 import sys

39 from io import StringIO

40

41 sys.stdin = StringIO()

42 task()

## Результаты выполнения

Вывод программы:

<bound method DataFrame.info of MedInc HouseAge AveRooms ... Latitude Longitude MedHouseVal  
0 8.3252 41.0 6.984127 ... 37.88 -122.23 4.526  
1 8.3014 21.0 6.238137 ... 37.86 -122.22 3.585  
2 7.2574 52.0 8.288136 ... 37.85 -122.24 3.521  
3 5.6431 52.0 5.817352 ... 37.85 -122.25 3.413  
4 3.8462 52.0 6.281853 ... 37.85 -122.25 3.422  
... ... ... ... ... ... ... ...  
20635 1.5603 25.0 5.045455 ... 39.48 -121.09 0.781  
20636 2.5568 18.0 6.114035 ... 39.49 -121.21 0.771  
20637 1.7000 17.0 5.205543 ... 39.43 -121.22 0.923  
20638 1.8672 18.0 5.329513 ... 39.43 -121.32 0.847  
20639 2.3886 16.0 5.254717 ... 39.37 -121.24 0.894  
  
[20640 rows x 9 columns]>  
  
Количество пропущенных значений по каждому столбцу:  
MedInc 0  
HouseAge 0  
AveRooms 0  
AveBedrms 0  
Population 0  
AveOccup 0  
Latitude 0  
Longitude 0  
MedHouseVal 0  
dtype: int64  
  
Записи, где средний возраст домов в районе более 50 лет и население более 2500 человек:  
 MedInc HouseAge AveRooms ... Latitude Longitude MedHouseVal  
460 1.4012 52.0 3.105714 ... 37.87 -122.26 1.75000  
4131 3.5349 52.0 4.646119 ... 34.13 -118.20 1.93600  
4440 2.6806 52.0 4.806283 ... 34.08 -118.21 1.53000  
5986 1.8750 52.0 4.500000 ... 34.10 -117.71 2.12500  
7369 3.1901 52.0 4.730942 ... 33.97 -118.21 1.67600  
8227 2.3305 52.0 3.488860 ... 33.78 -118.20 1.62500  
13034 6.1359 52.0 8.275862 ... 38.69 -121.15 2.25000  
15634 1.8295 52.0 2.628169 ... 37.80 -122.41 2.43800  
15652 0.9000 52.0 2.237474 ... 37.80 -122.41 5.00001  
15657 2.5166 52.0 2.839075 ... 37.79 -122.41 2.75000  
15659 1.7240 52.0 2.278566 ... 37.79 -122.41 2.25000  
15795 2.5755 52.0 3.402576 ... 37.77 -122.42 3.25000  
15868 2.8135 52.0 4.584329 ... 37.76 -122.41 2.60300  
  
[13 rows x 9 columns]  
  
Максимальное значение медианной стоимости дома: 5.00001  
Минимальное значение медианной стоимости дома: 0.14999  
  
Средние значения признаков:  
MedInc 3.870671  
HouseAge 28.639486  
AveRooms 5.429000  
AveBedrms 1.096675  
Population 1425.476744  
AveOccup 3.070655  
Latitude 35.631861  
Longitude -119.569704  
MedHouseVal 2.068558  
dtype: float64

## Отчет по файлу: extra1.py

## Исходный код

1 def task(text: str):

2 morze = {

3 "a": ".-",

4 "b": "-…",

5 "c": "-.-.",

6 "d": "-..",

7 "e": ".",

8 "f": "..-.",

9 "g": "--.",

10 "h": "….",

11 "i": "..",

12 "j": ".---",

13 "k": "-.-",

14 "l": ".-..",

15 "m": "--",

16 "n": "-.",

17 "o": "---",

18 "p": ".--.",

19 "q": "--.-",

20 "r": ".-.",

21 "s": "…",

22 "t": "-",

23 "u": "..-",

24 "v": "…-",

25 "w": ".--",

26 "x": "-..-",

27 "y": "-.--",

28 "z": "--..",

29 }

30

31 words = text.lower().split()

32 morze\_words = []

33 for word in words:

34 morze\_words.append(" ".join(morze[letter] for letter in word))

35

36 return "\n".join(morze\_words)

37

38

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

40 text = "Ignition sequence start"

41 print(task(text))

42

43 if \_\_name\_\_ == "builtins":

44 import sys

45 from io import StringIO

46

47 sys.stdin = StringIO()

48 text = "Ignition sequence start"

49 print(task(text))

## Результаты выполнения

Вывод программы:

.. --. -. .. - .. --- -.  
… . --.- ..- . -. -.-. .  
… - .- .-. -

## Отчет по файлу: extra2.py

## Исходный код

1 def task(n, requests):

2 database = {}

3

4 for name in requests:

5 if name not in database:

6 database[name] = 0

7 print("OK")

8 else:

9 database[name] += 1

10 new\_name = f"{name}{database[name]}"

11 while new\_name in database:

12 database[name] += 1

13 new\_name = f"{name}{database[name]}"

14 database[new\_name] = 0

15 print(new\_name)

16

17

18 if \_\_name\_\_ == "main":

19 n = int(input())

20 requests = [input().strip() for \_ in range(n)]

21

22 task(n, requests)

23

24

25 if \_\_name\_\_ == "builtins":

26 import sys

27 from io import StringIO

28

29 sys.stdin = StringIO("3\nb\nb\nb")

30

31 n = int(input())

32 requests = [input().strip() for \_ in range(n)]

33

34 task(n, requests)

## Результаты выполнения

Ввод:

3  
b  
b  
b

Вывод программы:

OK  
b1  
b2

## Отчет по файлу: extra3.py

## Исходный код

1 def task(n, file\_data, m, queries):

2 permissions = {}

3

4 for i in range(n):

5 data = file\_data[i].split()

6 file\_name = data[0]

7 rights = data[1:]

8 permissions[file\_name] = rights

9

10 operations = {"write": "w", "read": "r", "execute": "x"}

11

12 for query in queries:

13 operation, file\_name = query.split()

14 if operations[operation] in permissions.get(file\_name, set()):

15 print("OK")

16 else:

17 print("Access denied")

18

19

20 if \_\_name\_\_ == "main":

21 n = int(input())

22 file\_data = [input().strip() for \_ in range(n)]

23 m = int(input())

24 queries = [input().strip() for \_ in range(m)]

25

26 task(n, file\_data, m, queries)

27

28 if \_\_name\_\_ == "builtins":

29 import sys

30 from io import StringIO

31

32 sys.stdin = StringIO(

33 "3\n"

34 "python.exe x\n"

35 "book.txt r w\n"

36 "notebook.exe r w x\n"

37 "5\n"

38 "read python.exe\n"

39 "read book.txt\n"

40 "write notebook.exe\n"

41 "execute notebook.exe\n"

42 "write book.txt"

43 )

44

45 n = int(input())

46 file\_data = [input().strip() for \_ in range(n)]

47 m = int(input())

48 queries = [input().strip() for \_ in range(m)]

49

50 task(n, file\_data, m, queries)

## Результаты выполнения

Ввод:

3  
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 программы:

Access denied  
OK  
OK  
OK  
OK