

prac1

September 14, 2024

## 0.1 MAIN

**2**  $\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$ ,  $\frac{1}{2} \times \frac{1}{4} = \frac{1}{8}$ ,  $\frac{1}{4} \times \frac{1}{4} = \frac{1}{16}$ ,  $\frac{1}{8} \times \frac{1}{4} = \frac{1}{32}$ ,  $\frac{1}{16} \times \frac{1}{4} = \frac{1}{64}$ ,  $\frac{1}{32} \times \frac{1}{4} = \frac{1}{128}$ ,  $\frac{1}{64} \times \frac{1}{4} = \frac{1}{256}$ ,  $\frac{1}{128} \times \frac{1}{4} = \frac{1}{512}$ ,  $\frac{1}{256} \times \frac{1}{4} = \frac{1}{1024}$ ,  $\frac{1}{512} \times \frac{1}{4} = \frac{1}{2048}$ ,  $\frac{1}{1024} \times \frac{1}{4} = \frac{1}{4096}$ ,  $\frac{1}{2048} \times \frac{1}{4} = \frac{1}{8192}$ ,  $\frac{1}{4096} \times \frac{1}{4} = \frac{1}{16384}$ ,  $\frac{1}{8192} \times \frac{1}{4} = \frac{1}{32768}$ ,  $\frac{1}{16384} \times \frac{1}{4} = \frac{1}{65536}$ ,  $\frac{1}{32768} \times \frac{1}{4} = \frac{1}{131072}$ ,  $\frac{1}{65536} \times \frac{1}{4} = \frac{1}{262144}$ ,  $\frac{1}{131072} \times \frac{1}{4} = \frac{1}{524288}$ ,  $\frac{1}{262144} \times \frac{1}{4} = \frac{1}{1048576}$ ,  $\frac{1}{524288} \times \frac{1}{4} = \frac{1}{2097152}$ ,  $\frac{1}{1048576} \times \frac{1}{4} = \frac{1}{4194304}$ ,  $\frac{1}{2097152} \times \frac{1}{4} = \frac{1}{8388608}$ ,  $\frac{1}{4194304} \times \frac{1}{4} = \frac{1}{16777216}$ ,  $\frac{1}{8388608} \times \frac{1}{4} = \frac{1}{33554432}$ ,  $\frac{1}{16777216} \times \frac{1}{4} = \frac{1}{67108864}$ ,  $\frac{1}{33554432} \times \frac{1}{4} = \frac{1}{134217728}$ ,  $\frac{1}{67108864} \times \frac{1}{4} = \frac{1}{268435456}$ ,  $\frac{1}{134217728} \times \frac{1}{4} = \frac{1}{536870912}$ ,  $\frac{1}{268435456} \times \frac{1}{4} = \frac{1}{1073741824}$ ,  $\frac{1}{536870912} \times \frac{1}{4} = \frac{1}{2147483648}$ ,  $\frac{1}{1073741824} \times \frac{1}{4} = \frac{1}{4294967296}$ ,  $\frac{1}{2147483648} \times \frac{1}{4} = \frac{1}{8589934592}$ ,  $\frac{1}{4294967296} \times \frac{1}{4} = \frac{1}{17179869184}$ ,  $\frac{1}{8589934592} \times \frac{1}{4} = \frac{1}{34359738368}$ ,  $\frac{1}{17179869184} \times \frac{1}{4} = \frac{1}{68719476736}$ ,  $\frac{1}{34359738368} \times \frac{1}{4} = \frac{1}{137438953472}$ ,  $\frac{1}{68719476736} \times \frac{1}{4} = \frac{1}{274877906944}$ ,  $\frac{1}{137438953472} \times \frac{1}{4} = \frac{1}{549755813888}$ ,  $\frac{1}{274877906944} \times \frac{1}{4} = \frac{1}{1099511627776}$ ,  $\frac{1}{549755813888} \times \frac{1}{4} = \frac{1}{2199023255552}$ ,  $\frac{1}{1099511627776} \times \frac{1}{4} = \frac{1}{4398046511104}$ ,  $\frac{1}{2199023255552} \times \frac{1}{4} = \frac{1}{8796093022208}$ ,  $\frac{1}{4398046511104} \times \frac{1}{4} = \frac{1}{17592186044416}$ ,  $\frac{1}{8796093022208} \times \frac{1}{4} = \frac{1}{35184372088832}$ ,  $\frac{1}{17592186044416} \times \frac{1}{4} = \frac{1}{70368744177664}$ ,  $\frac{1}{35184372088832} \times \frac{1}{4} = \frac{1}{140737488355328}$ ,  $\frac{1}{70368744177664} \times \frac{1}{4} = \frac{1}{281474976710656}$ ,  $\frac{1}{140737488355328} \times \frac{1}{4} = \frac{1}{562949953421312}$ ,  $\frac{1}{281474976710656} \times \frac{1}{4} = \frac{1}{1125899906842624}$ ,  $\frac{1}{562949953421312} \times \frac{1}{4} = \frac{1}{2251799813685248}$ ,  $\frac{1}{1125899906842624} \times \frac{1}{4} = \frac{1}{4503599627370496}$ ,  $\frac{1}{2251799813685248} \times \frac{1}{4} = \frac{1}{9007199254740992}$ ,  $\frac{1}{4503599627370496} \times \frac{1}{4} = \frac{1}{18014398509481984}$ ,  $\frac{1}{9007199254740992} \times \frac{1}{4} = \frac{1}{36028797018963968}$ ,  $\frac{1}{18014398509481984} \times \frac{1}{4} = \frac{1}{72057594037927936}$ ,  $\frac{1}{36028797018963968} \times \frac{1}{4} = \frac{1}{144115188075855872}$ ,  $\frac{1}{72057594037927936} \times \frac{1}{4} = \frac{1}{288230376151711744}$ ,  $\frac{1}{144115188075855872} \times \frac{1}{4} = \frac{1}{576460752303423488}$ ,  $\frac{1}{288230376151711744} \times \frac{1}{4} = \frac{1}{1152921504606846976}$ ,  $\frac{1}{576460752303423488} \times \frac{1}{4} = \frac{1}{2305843009213693952}$ ,  $\frac{1}{1152921504606846976} \times \frac{1}{4} = \frac{1}{4611686018427387904}$ ,  $\frac{1}{2305843009213693952} \times \frac{1}{4} = \frac{1}{9223372036854775808}$ ,  $\frac{1}{4611686018427387904} \times \frac{1}{4} = \frac{1}{18446744073709551616}$ ,  $\frac{1}{9223372036854775808} \times \frac{1}{4} = \frac{1}{36893488147419103232}$ ,  $\frac{1}{18446744073709551616} \times \frac{1}{4} = \frac{1}{73786976294838206464}$ ,  $\frac{1}{36893488147419103232} \times \frac{1}{4} = \frac{1}{147573952589676412928}$ ,  $\frac{1}{73786976294838206464} \times \frac{1}{4} = \frac{1}{295147905179352825856}$ ,  $\frac{1}{147573952589676412928} \times \frac{1}{4} = \frac{1}{590295810358705651712}$ ,  $\frac{1}{295147905179352825856} \times \frac{1}{4} = \frac{1}{1180591620717411303424}$ ,  $\frac{1}{590295810358705651712} \times \frac{1}{4} = \frac{1}{2361183241434822606848}$ ,  $\frac{1}{1180591620717411303424} \times \frac{1}{4} = \frac{1}{4722366482869645213696}$ ,  $\frac{1}{2361183241434822606848} \times \frac{1}{4} = \frac{1}{9444732965739290427392}$ ,  $\frac{1}{4722366482869645213696} \times \frac{1}{4} = \frac{1}{18889465931478580854784}$ ,  $\frac{1}{9444732965739290427392} \times \frac{1}{4} = \frac{1}{37778931862957161709568}$ ,  $\frac{1}{18889465931478580854784} \times \frac{1}{4} = \frac{1}{75557863725914323419136}$ ,  $\frac{1}{37778931862957161709568} \times \frac{1}{4} = \frac{1}{151115727451828646838272}$ ,  $\frac{1}{75557863725914323419136} \times \frac{1}{4} = \frac{1}{302231454903657293676544}$ ,  $\frac{1}{151115727451828646838272} \times \frac{1}{4} = \frac{1}{604462909807314587353088}$ ,  $\frac{1}{302231454903657293676544} \times \frac{1}{4} = \frac{1}{1208925819614629174706176}$ ,  $\frac{1}{604462909807314587353088} \times \frac{1}{4} = \frac{1}{2417$

```
[6]: from abc import ABC, abstractmethod

class Shape(ABC):
    @abstractmethod
    def area(self):
        pass

class Rectangle(Shape):
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

class Triangle(Shape):
    def __init__(self, base, height):
        self.base = base #
        self.height = height #

    def area(self):
        return 0.5 * self.base * self.height

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14 * self.radius ** 2
```

```

rectangle = Rectangle(10, 10)
triangle = Triangle(10, 10)
circle = Circle(10)

(
    rectangle.area(),
    triangle.area(),
    circle.area(),
)

```

[6]: (100, 50.0, 314.0)

3  
: +, -, /, //, abs - , pow \*\* - .

[7]: `import operator`

```

def safe_eval(expression: str):
    allowed_operators = {
        '+': operator.add,
        '-': operator.sub,
        '**': operator.pow,
        '*': operator.mul,
        '//': operator.floordiv,
        '/': operator.truediv,
    }

    for op in allowed_operators:
        if op in expression:
            left, right = expression.split(op)
            left, right = float(left.strip()), float(right.strip())
            return allowed_operators[op](left, right)

    return "Unknown or unsupported operation"

safe_eval("10+10")

```

[7]: 20.0

4 , ( ) , 0

```
[8]: def sum_of_squares_until_zero():
    numbers = []
    total_sum = 0

    while True:
        num = int(input("      : "))
        numbers.append(num)
        total_sum += num
        if total_sum == 0:
            break

    return sum(x ** 2 for x in numbers)

sum_of_squares_until_zero()

#1
#-1
```

[8]: 2

5 , N, 1 2 2 3 3 3 4.

N. , N = 7,

– print(\*list).

```
[9]: def generate_seq(N: int) -> None:
    sequence = []

    for i in range(1, N + 1):
        sequence.extend([i] * i)

    sequence = sequence[:N]

    print(*sequence)

N = int(input("N: "))
generate_seq(N)
# 7
```

1 2 2 3 3 3 4

6 :

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

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

, – , – ,

: {'a' : 10, 'b' : 15, 'c' : 6}

```
[10]: from collections import defaultdict

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_defaultdict = defaultdict(int)

for key, value in zip(B, A):
    result_defaultdict[key] += value

result_defaultdict
```

```
[10]: defaultdict(int, {'a': 17, 'b': 11, 'c': 17})
```

7 , sklearn:

```
from sklearn.datasets import fetch_california_housing
data = fetch_california_housing(as_frame=True)
```

```
[1]: !pip install scikit-learn pandas
```

```
Requirement already satisfied: scikit-learn in
/home/codespace/.local/lib/python3.12/site-packages (1.5.1)
Requirement already satisfied: pandas in
/home/codespace/.local/lib/python3.12/site-packages (2.2.2)
Requirement already satisfied: numpy>=1.19.5 in
/home/codespace/.local/lib/python3.12/site-packages (from scikit-learn) (2.1.0)
Requirement already satisfied: scipy>=1.6.0 in
/home/codespace/.local/lib/python3.12/site-packages (from scikit-learn) (1.14.0)
Requirement already satisfied: joblib>=1.2.0 in
/home/codespace/.local/lib/python3.12/site-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/home/codespace/.local/lib/python3.12/site-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/home/codespace/.local/lib/python3.12/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/home/codespace/.local/lib/python3.12/site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in
/home/codespace/.local/lib/python3.12/site-packages (from pandas) (2024.1)
Requirement already satisfied: six>=1.5 in
/home/codespace/.local/lib/python3.12/site-packages (from python-
dateutil>=2.8.2->pandas) (1.16.0)
```

```
[11]: from sklearn.datasets import fetch_california_housing
data = fetch_california_housing(as_frame=True)
```

```
[12]: import pandas as pd
```

```
data: dict[str, pd.DataFrame]
df: pd.DataFrame = data["frame"]
df
```

```
[12]:
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	\
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	
...	...	...	...	...	...	...	...	
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	

  

	Longitude	MedHouseVal
0	-122.23	4.526
1	-122.22	3.585
2	-122.24	3.521
3	-122.25	3.413
4	-122.25	3.422
...	...	...
20635	-121.09	0.781
20636	-121.21	0.771
20637	-121.22	0.923
20638	-121.32	0.847
20639	-121.24	0.894

[20640 rows x 9 columns]

```
8 info() :
```

```
[15]: df.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   MedHouseVal   20640 non-null float64
dtypes: float64(9)
memory usage: 1.4 MB

```

```

9   ,   ,   isna().sum()

```

```
[16]: df.isna().sum()
```

```

[16]: MedInc      0
      HouseAge    0
      AveRooms    0
      AveBedrms   0
      Population  0
      AveOccup    0
      Latitude    0
      Longitude   0
      MedHouseVal 0
      dtype: int64

```

```

10   ,   50   2500   ,   loc()

```

```
[17]: df.loc[(df['HouseAge'] > 50) & (df['Population'] > 2500)]
```

```

[17]:
      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  MedHouseVal
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

11

```
[18]: (
    df['MedHouseVal'].min(),
    df['MedHouseVal'].max(),
)
```

```
[18]: (np.float64(0.14999), np.float64(5.00001))
```

12                      apply(),

```
[19]: df.apply(lambda x: x.mean())
```

```
[19]: 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
```

13

## 0.2 STARS

1\*

```

    ,      «g»      «-.»      morze

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': '...-'}

    :
    ,
    :
    ,
    «Help»
    «... . -.. -..».
    : Ignition
sequence start

.. --. -. . - .. --- -.
... . --. - .. - . - . -.. .
... - . - . - . -

```

```

[1]: 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': '...-'}

text = "Ignition sequence start"
result = ""

for word in text.split():
    for letter in word.lower():
        result += morze[letter] + " "
    result += "\n"

print(result)

```

```

.. --. -. . - .. --- -.
... . --. - .. - . - . -.. .
... - . - . - . -

```

2\*

```

:
,
name

,
" ",
,
,
: name
, 1 (name1, name2 ),
i, namei

:
n (1 n 100000). n
32 ,
:
n - : "OK"
,

```



```
[5]: def registration_system(queries):
    name_counts = {}
    results = []

    for name in queries:
        if name not in name_counts:
            name_counts[name] = 1
            results.append("OK")

        else:
            i = name_counts[name]
            new_name = f"{name}{i}"
            while new_name in name_counts:
                i += 1
                new_name = f"{name}{i}"

            name_counts[name] = i + 1
            name_counts[new_name] = 1
            results.append(new_name)

    return results

n = int(input())
queries = [input().strip()[:32] for _ in range(n)]
results = registration_system(queries)
print("result =>")
for result in results:
    print(result)

# 4
# name
# name1
# name
# name1
```

```
result =>
OK
OK
name2
name11
```

```
3
:      - w ("write"),      - r ("read"),      - x ("execute").
:
:
:
-   n -
-   n                               (w, x, r),
```

```
- m - « » ( : write, read, execute).
:
OK, - Access denied.
```

```
[27]: # filename: [file rrr]
files = {

}

commands = {
    "write": "w",
    "executex": "x",
    "read": "r",
}
file_n = int(input())

for n in range(file_n):
    s = input().split()
    files[s[0]] = s[1:]

    """
    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

    """
```

```
[27]: '\n3\npython.exe x\nbook.txt r w\nnotebook.exe r w x\n5\nread python.exe\nread
book.txt\nwrite notebook.exe\nexecute notebook.exe\nwrite book.txt\n\n'
```

```
[28]: cmd_n = int(input())

for n in range(cmd_n):
    s = input().split()
    command = commands[s[0]]
    file = s[1]
    if file in files and command in files.get(file):
        print("OK")
```

```
else:  
    print("ERROR")
```

ERROR

OK