Practice problems

September 25, 2024

[1]:
```
l=[]
for i in range(200, 320):
    if (i%7==0) and (i%5!=0):
        l.append(i)
y=tuple(l)
print(y)
```

```
(203, 217, 224, 231, 238, 252, 259, 266, 273, 287, 294, 301, 308)
```

[2]:
```
myDict = {"name": "John", "country": "Norway"}
mySeparator = ","

x = mySeparator.join(myDict)

print(x)
```

```
name,country
```

[11]:
```
values=input()
l=values.split(" ")
print(type(l))
l= [int(i) for i in l]
x=l.index(max(l))
print(x)
t=tuple(l)
print(l)
print(t)
```

```
5 4 2 5 8
<class 'list'>
4
[5, 4, 2, 5, 8]
(5, 4, 2, 5, 8)
```

[4]:
```
values = ["  apple  ", " banana ", "  cherry  "]
cleaned_values = [value.strip() for value in values]  # Removes leading and␣
 ↪trailing spaces from each string
print(cleaned_values)
```

```
['apple', 'banana', 'cherry']
```

[22]:
```
user_in = input("Enter comma-separated values: ").split(",")
l1=[int(j.strip()) for j in user_in]
x=len(l1)
min_value=min(l1)
for i in range(0,x-1):
    first_min_index=l1.index(min_value)
    #print(first_min_index)
    l1.append(l1.pop(first_min_index))
dict_freq={i:l1.count(i) for i in l1}
print(l1)
print(dict_freq)
```

```
Enter comma-separated values: 5, 4,0,2,4,5
[5, 4, 2, 4, 5, 0]
{5: 2, 4: 2, 2: 1, 0: 1}
```

[23]:
```
user_in = input().split(",")
l1=[int(j.strip()) for j in user_in]
x=len(l1)
min_value=min(l1)
for i in range(0,x-1):
    first_min_index=l1.index(min_value)
    #print(first_min_index)
    l1.append(l1.pop(first_min_index))
dict_freq={i:l1.count(i) for i in l1}
print(l1)
print(dict_freq)
```

```
1,2,3,0, 4,2,0,1
[1, 2, 3, 4, 2, 1, 0, 0]
{1: 2, 2: 2, 3: 1, 4: 1, 0: 2}
```

[26]:
```
tuples = []
while True:
    user_input = input("Enter a tuple (name, age, height) or press Enter to stop:
 ↪ ")
    if not user_input:
        break
    name, age, height = user_input.split(",")
    # Append the tuple with appropriate types
    tuples.append((name, int(age), int(height)))

# Sort the tuples using a simple bubble sort based on name, age, and height
for i in range(len(tuples)):
    for j in range(0, len(tuples) - i - 1):
        # Compare by name, then by age, then by height
```

```
        if (tuples[j][0] > tuples[j + 1][0] or
            (tuples[j][0] == tuples[j + 1][0] and tuples[j][1] > tuples[j +␣
 ↪1][1]) or
            (tuples[j][0] == tuples[j + 1][0] and tuples[j][1] == tuples[j +␣
 ↪1][1] and tuples[j][2] > tuples[j + 1][2])):
            # Swap if the current tuple is greater than the next tuple
            tuples[j], tuples[j + 1] = tuples[j + 1], tuples[j]

# Print the sorted tuples
print(tuples)
```

```
Enter a tuple (name, age, height) or press Enter to stop: Jony,17,91
Enter a tuple (name, age, height) or press Enter to stop: Tom,19,80
Enter a tuple (name, age, height) or press Enter to stop: Jaon,21,85
Enter a tuple (name, age, height) or press Enter to stop:
[('Jaon', 21, 85), ('Jony', 17, 91), ('Tom', 19, 80)]
```

[31]:
```
tuples = []
max_users = 3
for i in range(max_users):
    user_input = input()
    if not user_input:
        break
    name, age, height = user_input.split(",")
    # Append the tuple with appropriate types
    tuples.append((name, int(age), int(height)))

# Sort the tuples using a simple bubble sort based on name, age, and height
for i in range(len(tuples)):
    for j in range(0, len(tuples) - i - 1):
        # Compare by name, then by age, then by height
        if (tuples[j][0] > tuples[j + 1][0] or
            (tuples[j][0] == tuples[j + 1][0] and tuples[j][1] > tuples[j +␣
 ↪1][1]) or
            (tuples[j][0] == tuples[j + 1][0] and tuples[j][1] == tuples[j +␣
 ↪1][1] and tuples[j][2] > tuples[j + 1][2])):
            # Swap if the current tuple is greater than the next tuple
            tuples[j], tuples[j + 1] = tuples[j + 1], tuples[j]

# Print the sorted tuples
print(tuples)
```

```
Jony,17,91
Tom,19,80
Jaon,21,85
[('Jaon', 21, 85), ('Jony', 17, 91), ('Tom', 19, 80)]
```

[2]:
```
items=[x for x in input().split(',')]
items.sort()
print(','.join(items))
```

```
hello,1,know,25,alphabet
1,25,alphabet,hello,know
```

[3]:
```
lines = []
while True:
    s = input()
    if s:
        lines.append(s.upper())
    else:
        break;

for sentence in lines:
    print(sentence)
```

```
Hello world
Practice makes perfect

HELLO WORLD
PRACTICE MAKES PERFECT
```

[4]:
```
s = input()
d={"DIGITS":0, "LETTERS":0}
for c in s:
    if c.isdigit():
        d["DIGITS"]+=1
    elif c.isalpha():
        d["LETTERS"]+=1
    else:
        pass
print("LETTERS", d["LETTERS"])
print("DIGITS", d["DIGITS"])
```

```
hello world! 123
LETTERS 10
DIGITS 3
```

[ ]:
```
l3=[]
n=int(input("number of inputs"))
for i in range(n):
    x=input()
    if x=="":
        break
    l3.append(x)
```

```
print(l3)
print(tuple(l3))

#t=tuple(l)
#print(l)
#print(t)
```

[8]:
```
user_input = input("Enter a sequence of numbers separated by spaces: ")
nums = list(map(int, user_input.split()))
print(len(nums))
# Initialize a list to store maximum sums up to each index
max_sums = [0] * len(nums)
print(max_sums)
# Handle the first element
max_sums[0] = nums[0]
# Handle the second element if it exists
if len(nums) > 1:
    max_sums[1] = max(nums[0], nums[1])
    # Iterate through the list starting from the third element
for i in range(2, len(nums)):
    # Calculate the maximum sum including the current number
    include_current = max_sums[i - 2] + nums[i]
    # The maximum sum excluding the current number
    exclude_current = max_sums[i - 1]
    # Store the maximum of including or excluding the current number
    max_sums[i] = max(include_current, exclude_current)

# The last element in max_sums will hold the result
largest_sum = max_sums[-1]
print(largest_sum)
```

```
Enter a sequence of numbers separated by spaces: 6 6 -1 8
4
[0, 0, 0, 0]
14
```

[11]:
```
# Input list of integers
user_input = input("Enter a sequence of numbers separated by spaces: ")
nums = list(map(int, user_input.split()))

# Initialize maximum sums
max_sums = [0] * len(nums)

# Handle the first element
max_sums[0] = nums[0]

# Handle the second element if it exists
if len(nums) > 1:
```

```
    max_sums[1] = max(nums[0], nums[1])

# Iterate through the list starting from the third element
for i in range(2, len(nums)):
    include_current = max_sums[i - 2] + nums[i]
    exclude_current = max_sums[i - 1]
    max_sums[i] = max(include_current, exclude_current)

# The last element in max_sums will hold the result
largest_sum = max_sums[-1]

# Print the result
print(largest_sum)
```

```
Enter a sequence of numbers separated by spaces: 1 8 0 -1 6 3
14
```

[12]:
```
from typing import List


def max_nonadjacent_sum(arr: List[int]) -> int:
    including = 0
    excluding = 0
    for elem in arr:
        # updating maximum sum including and excluding the current element
        including, excluding = max(excluding + elem, elem), max(excluding,␣
 ↪including)
    return max(including, excluding)


if __name__ == "__main__":
    print(max_nonadjacent_sum([2, 4, 6, 8]))
    print(max_nonadjacent_sum([5, 1, 1, 5]))
    print(max_nonadjacent_sum([-5, 1, 1, -5]))
    print(max_nonadjacent_sum([1, 8, 0, -1, 6, 3]))
```

```
12
10
1
14
```

[5]:
```
# Input list of integers
user_input = input()
nums = list(map(int, user_input.split()))

# Initialize maximum sums
max_sums = [0] * len(nums)
```

```
# Handle the first element
max_sums[0] = nums[0]

# Handle the second element if it exists
if len(nums) > 1:
    max_sums[1] = max(nums[0], nums[1])

# Iterate through the list starting from the third element
for i in range(2, len(nums)):
    include_current = max_sums[i - 2] + nums[i]
    exclude_current = max_sums[i - 1]
    max_sums[i] = max(include_current, exclude_current)

# The last element in max_sums will hold the result
largest_sum = max_sums[-1]

# Print the result
print(largest_sum)
```

```
4 12 -1 3 5 6 9
26
```

[11]:
```
rows=int(input("No. of rows:"))
columns=int(input("No.of Coloumns:"))

m1=[]
for i in range(rows):
    r1=[]
    for j in range(columns):
        c1=int(input())
        r1.append(c1)
    #data1=list(map(int,input().split()))
    m1.append(r1)
print(m1)
print(type(m1))
for
```

```
No. of rows:2
No.of Coloumns:2
1
2
3
4
[[1, 2], [3, 4]]
<class 'list'>
```

[2]:
```
rows=int(input("No. of rows:"))
columns=int(input("No.of Coloumns:"))
```

```
m1=[]
for i in range(rows):
    data1=list(map(int,input().split()))
    m1.append(data1)
print(m1)
```

```
No. of rows:3
No.of Coloumns:3
1 2 3
4 5 6
7 8 9
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

[2]:
```
# Get the number of rows and columns from the user
rows = int(input("Enter the number of rows: "))
cols = int(input("Enter the number of columns: "))

# Initialize the matrix
matrix = []

# Get the matrix elements from the user
for i in range(rows):
    while True:  # Loop until the user enters the correct number of values
        # Read a row of space-separated integers
        row = list(map(int, input(f"Enter {cols} elements for row {i + 1}: ").
 ↪split()))

        # Check if the number of elements in the row matches the number of␣
 ↪columns
        if len(row) == cols:
            matrix.append(row)
            break  # Exit the loop if the input is valid
        else:
            print(f"Error: You must enter exactly {cols} integers. Please try␣
 ↪again.")

# Initialize the maximum element
max_element = float('-inf')

# Find the maximum element in the matrix
for row in matrix:
    max_element = max(max_element, max(row))

# Print the maximum element
print(max_element)
```

```
Enter the number of rows: 3
```

```
Enter the number of columns: 3
Enter 3 elements for row 1: 1 2 3
Enter 3 elements for row 2: 4 5 6
Enter 3 elements for row 3:  7 8 9
9
```

[4]:
```python
while True:
    try:
        user_input = input("Please enter an integer: ")
        number = int(user_input)  # Attempt to convert input to an integer
        print(f"You entered the integer: {number}")
        break  # Exit the loop if input is valid
    except ValueError:  # Handle the case where conversion fails
        print("That's not a valid integer. Please try again.")
```

```
Please enter an integer: c
That's not a valid integer. Please try again.
Please enter an integer: 3.2
That's not a valid integer. Please try again.
Please enter an integer: 3
You entered the integer: 3
```

[10]:
```python
user_input = int(input())
list(user_input)
```

```
123
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-10-9b866efbd245> in <module>
      1 user_input = int(input())
----> 2 list(user_input)

TypeError: 'int' object is not iterable
```

[1]:
```python
from typing import List


def first_missing_positive_integer(arr: List[int]) -> int:
    # placing the positive elements (< length) in their proper position
    # proper position index = element - 1
    # if after the palcement is complete, index of the 1st element not in its␣
 ↪proper
    # position is the answer
    length = len(arr)
    for i in range(length):
        correctPos = arr[i] - 1
```

```
        while 1 <= arr[i] <= length and arr[i] != arr[correctPos]:
            arr[i], arr[correctPos] = arr[correctPos], arr[i]
            correctPos = arr[i] - 1
    # finding the first missing positive integer
    for i in range(length):
        if i + 1 != arr[i]:
            return i + 1
    return length + 1


if __name__ == "__main__":
    print(first_missing_positive_integer([3, 4, 2, 1]))
    print(first_missing_positive_integer([3, 4, -1, 1]))
    print(first_missing_positive_integer([1, 2, 5]))
    print(first_missing_positive_integer([-1, -2]))
```

```
5
2
3
1
```

[28]:
```
l1=list(map(int,input().split()))
#l1=[x for x in l1 if x>=0]
for i in l1[:]:
    #print(i)
    if i<0:
      l1.remove(i)
print(l1)
l2=[]
if l1:
    a=min(l1)
    b=max(l1)
    print(a)
    print(b)
    #l2=list(range(a,b+1))
    for j in range(a, b+1):
        l2.append(j)
    print(l2)
```

```
3 4 -1 1
[3, 4, 1]
1
4
[1, 2, 3, 4]
```

[9]:
```
l1=list(map(int,input().split()))
l1=[x for x in l1 if x>=0]
```

```
print(l1)
```

```
3 4 -1 1
[3, 4, 1]
```

[13]:
```
l1 = [1, -2, 3, -4, 5]  # Example list

# Iterate over a copy of the list while modifying the original list
for item in l1[:]:  # l1[:] creates a shallow copy of the list
    if item < 0:
        l1.remove(item)

print(l1)  # Output will be [1, 3, 5]
```

```
[1, 3, 5]
```

[29]:
```
nums=list(map(int,input().split()))
n = len(nums)

# Step 1: Place each number in its correct index
for i in range(n):
    while 1 <= nums[i] <= n and nums[nums[i] - 1] != nums[i]:
        # Swap nums[i] with nums[nums[i] - 1]
        nums[nums[i] - 1], nums[i] = nums[i], nums[nums[i] - 1]

# Step 2: Find the first index where nums[i] != i + 1
for i in range(n):
    if nums[i] != i + 1:
        print(i + 1)
        break
else:
    # If all numbers are in place, the missing number is n + 1
    print(n + 1)
```

```
3 4 -1 1
2
```

[1]:
```
nums=list(map(int,input().split()))
for i in nums:
    print(i)
```

```
5 8 9 6 1 -4
5
8
9
6
1
-4
```

[1]:
```
nums=list(map(int,input().split()))
print(nums)
for i in range(len(nums)):
    print(i)
```

```
5 4 2 3
[5, 4, 2, 3]
0
1
2
3
```

[3]:
```
arr_input = input("Enter the array elements (separated by spaces): ")

# Split the input string into individual elements
arr = arr_input.split()

# Convert the elements to characters
for i in range(len(arr)):
    arr[i] = arr[i].strip("'")  # Remove single quotes if present
print(arr)
```

```
Enter the array elements (separated by spaces): B B G R
['B', 'B', 'G', 'R']
```

[5]:
```
L1= [5,8,3,2]  # Question 1
for x in range(len(L1)):
    print(x)
```

```
0
1
2
3
```

[6]:
```
x=3
print(eval('x**2'))
```

```
9
```

[12]:
```
for name in ['X','Y','Z']:
    if name[0]=='Y':
        print(name)
```

```
Y
```

[ ]:
```
for name in [X,Y,Z]:
    if name['0']='Y':
        print(name)
```

[18]:
```
D1= {5:"hi",8:3.5,3:"no",2:"a"}  # Question 1
for x in D1:
    z=D1.values()
print(z)
```

```
dict_values(['hi', 3.5, 'no', 'a'])
```

[25]:
```
D1= {5:"hi",8=3.5,3:"no",2:a]  # Question 1
for x in D1
z=D1.values
print(z)
```

```
  File "<ipython-input-25-db77e6b37ded>", line 1
    D1= {5:"hi",8=3.5,3:"no",2:a]  # Question 1
                ^
SyntaxError: invalid syntax
```

[26]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
C = {8,9,10}
set1=A.intersection_update(B,C)
```

[ ]:
```
l = ['ab','cd']
for i in l:
 l.append('ef')
print(l)
```

[1]:
```
print(5==1)
```

```
False
```

[3]:
```
number = input()
#number = 12345  # You can change this number for testing
unique = True
# Iterate through each digit in the number
for i in range(len(number)):
    current_digit = number[i]
    #print(i)
    # Check the remaining digits to see if the current digit is repeated
    for j in range(i + 1, len(number)):
        if current_digit == number[j]:
            unique = False
            break
# Output the result
if unique == True:
    print("UNIQUE")
```

```
else:
    print("NOT UNIQUE")
```

```
55241
0
1
2
3
4
NOT UNIQUE
```

[ ]:
```
a = int(input())
b = str(a)
l = list(b)
for i in l:
    if l.count(i)>1:
        print(l.count(i))
        print("NOT UNIQUE")
        break
    else:
        print("UNIQUE")
        break
```

[ ]:

[ ]: