

Assignment - 2
Python Programming

Assignment Date	29 September 2022
Team Id	PNT20222TMID44152
Student Name	Mr. Mirshad KT
Student Roll Number	724019104009
Maximum Marks	2 Marks

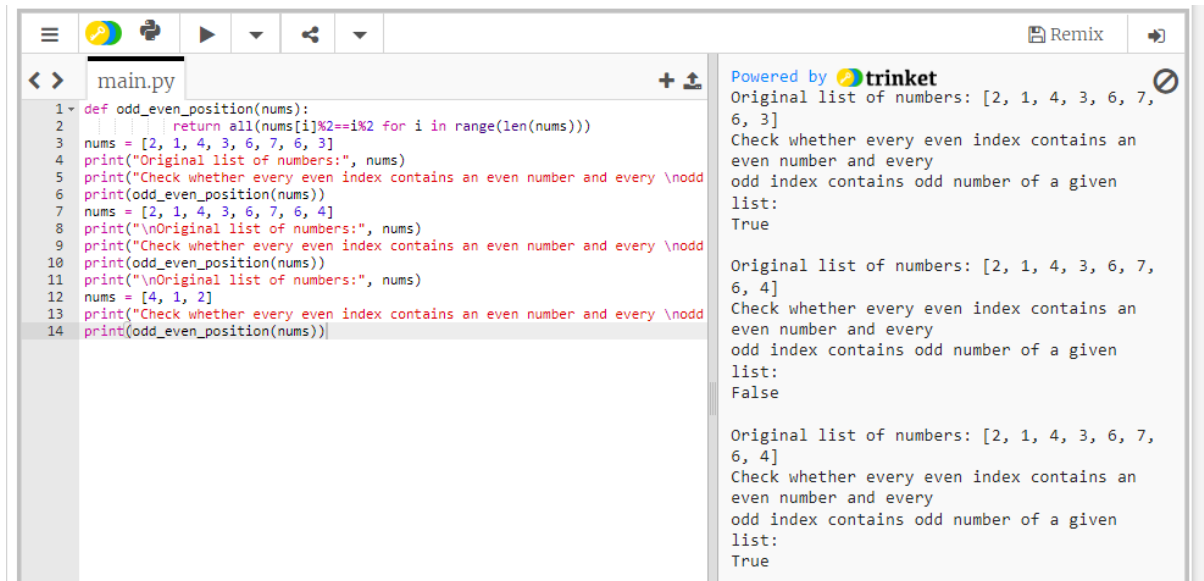
Question-1:

Write a Python program to check whether every even index contains an even number and every odd index contains odd number of a given list.

Solution:


```
def odd_even_position(nums):  
    return all(nums[i]%2==i%2 for i in range(len(nums)))  
  
nums = [2, 1, 4, 3, 6, 7, 6, 3]  
  
print("Original list of numbers:", nums)  
  
print("Check whether every even index contains an even number and every \nodd  
index contains odd number of a given list:")  
  
print(odd_even_position(nums))  
  
nums = [2, 1, 4, 3, 6, 7, 6, 4]  
  
print("\nOriginal list of numbers:", nums)  
  
print("Check whether every even index contains an even number and every \nodd  
index contains odd number of a given list:")  
  
print(odd_even_position(nums))  
  
print("\nOriginal list of numbers:", nums)  
  
nums = [4, 1, 2]  
  
print("Check whether every even index contains an even number and every \nodd  
index contains odd number of a given list:")  
  
print(odd_even_position(nums))
```

Output:



The screenshot shows a Trinket.io code editor with a file named `main.py`. The code defines a function `odd_even_position` that returns a list of booleans indicating whether each element in the input list is at an even index and is an even number. The code then tests this function with three different lists: `[2, 1, 4, 3, 6, 7, 6, 3]`, `[2, 1, 4, 3, 6, 7, 6, 4]`, and `[4, 1, 2]`. The output on the right shows the results of these tests.

```
1 def odd_even_position(nums):
2     return all(nums[i]%2==i%2 for i in range(len(nums)))
3
4 nums = [2, 1, 4, 3, 6, 7, 6, 3]
5 print("Original list of numbers:", nums)
6 print("Check whether every even index contains an even number and every \nodd
7 print(odd_even_position(nums))
8 nums = [2, 1, 4, 3, 6, 7, 6, 4]
9 print("\nOriginal list of numbers:", nums)
10 print("Check whether every even index contains an even number and every \nodd
11 print(odd_even_position(nums))
12 print("\nOriginal list of numbers:", nums)
13 nums = [4, 1, 2]
14 print("Check whether every even index contains an even number and every \nodd
15 print(odd_even_position(nums))
```

Powered by  **trinket**

Original list of numbers: [2, 1, 4, 3, 6, 7, 6, 3]
Check whether every even index contains an even number and every odd index contains odd number of a given list:
True

Original list of numbers: [2, 1, 4, 3, 6, 7, 6, 4]
Check whether every even index contains an even number and every odd index contains odd number of a given list:
False

Original list of numbers: [2, 1, 4, 3, 6, 7, 6, 4]
Check whether every even index contains an even number and every odd index contains odd number of a given list:
True

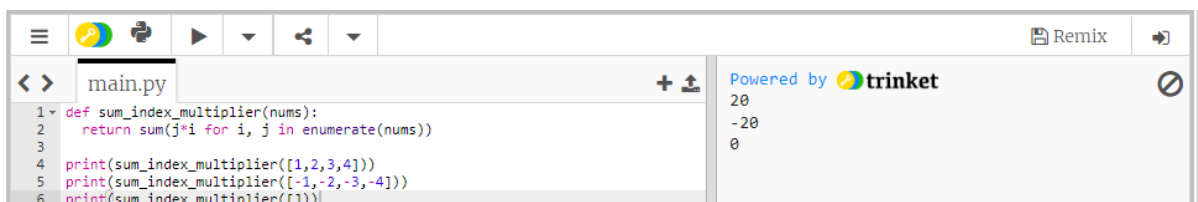
Question-2:

Write a Python program to compute the sum of all items of a given array of integers where each integer is multiplied by its index. Return 0 if there is no number.

SOLUTION:

```
def sum_index_multiplier(nums):  
    return sum(j*i for i, j in enumerate(nums))  
  
print(sum_index_multiplier([1,2,3,4]))  
print(sum_index_multiplier([-1,-2,-3,-4]))  
print(sum_index_multiplier([]))
```

OUTPUT:



The screenshot shows a Python code editor interface. The code is as follows:

```
1 def sum_index_multiplier(nums):  
2     return sum(j*i for i, j in enumerate(nums))  
3  
4 print(sum_index_multiplier([1,2,3,4]))  
5 print(sum_index_multiplier([-1,-2,-3,-4]))  
6 print(sum_index_multiplier([]))
```

The output on the right side of the editor is:

```
20  
-20  
0
```

The editor is powered by trinket, as indicated by the logo in the top right corner.

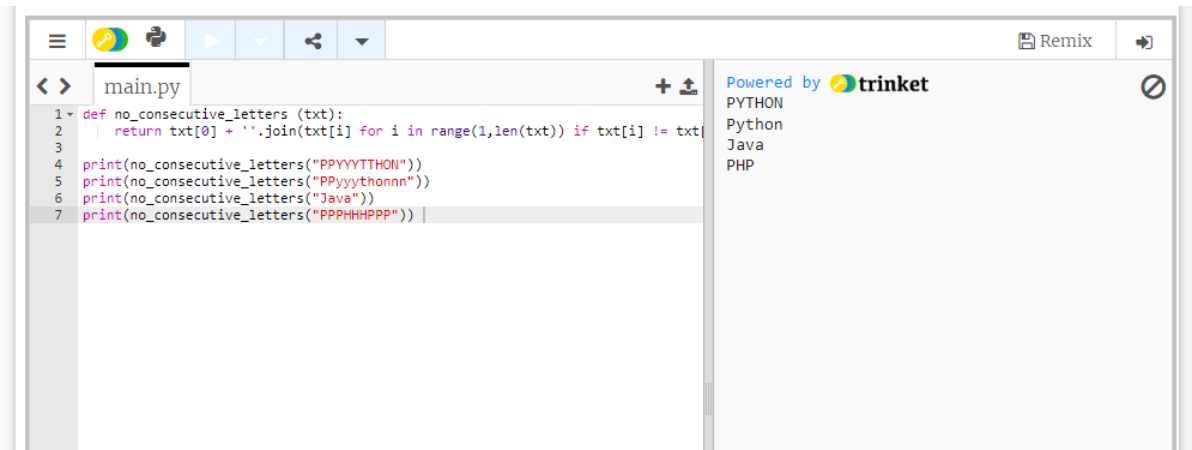
Question-3:

Write a Python program to create a new string with no duplicate consecutive letters from a given string.

SOLUTION:

```
def no_consecutive_letters (txt):  
    return txt[0] + ''.join(txt[i] for i in range(1,len(txt)) if txt[i] != txt[i-1])  
  
print(no_consecutive_letters("PPYYTTHON"))  
print(no_consecutive_letters("PPyythonnn"))  
print(no_consecutive_letters("Java"))  
print(no_consecutive_letters("PPPHHHPPP"))
```

OUTPUT:



The screenshot shows a web-based Python IDE interface. On the left, a code editor window titled 'main.py' contains the following Python code:

```
1 def no_consecutive_letters (txt):  
2     return txt[0] + ''.join(txt[i] for i in range(1,len(txt)) if txt[i] != txt[i-1])  
3  
4 print(no_consecutive_letters("PPYYTTHON"))  
5 print(no_consecutive_letters("PPyythonnn"))  
6 print(no_consecutive_letters("Java"))  
7 print(no_consecutive_letters("PPPHHHPPP"))
```

On the right, the output panel shows the results of the program execution:

```
Powered by trinket  
PYTHON  
Python  
Java  
PHP
```

Question-4:

Write a Python program to check whether a given sequence is linear, quadratic or cubic.

Sequences are sets of numbers that are connected in some way.

Linear sequence:

A number pattern which increases or decreases by the same amount each time is called a linear sequence. The amount it increases or decreases by is known as the common difference.

Quadratic sequence:

In quadratic sequence, the difference between each term increases, or decreases, at a constant rate.

Cubic sequence:

Sequences where the 3rd difference are known as cubic sequence.

SOLUTION:

```
def Seq_Linear_Quadratic_Cubic(seq_nums):

    seq_nums = [seq_nums[x] - seq_nums[x-1] for x in range(1, len(seq_nums))]

    if len(set(seq_nums)) == 1: return "Linear Sequence"

    seq_nums = [seq_nums[x] - seq_nums[x-1] for x in range(1, len(seq_nums))]

    if len(set(seq_nums)) == 1: return "Quadratic Sequence"

    seq_nums = [seq_nums[x] - seq_nums[x-1] for x in range(1, len(seq_nums))]

    if len(set(seq_nums)) == 1: return "Cubic Sequence"

nums = [0,2,4,6,8,10]

print("Original Sequence:",nums)

print("Check the said sequence is Linear, Quadratic or Cubic?")

print(Seq_Linear_Quadratic_Cubic(nums))

nums = [1,4,9,16,25]

print("\nOriginal Sequence:",nums)
```

```
print("Check the said sequence is Linear, Quadratic or Cubic?")
```

```
print(Seq_Linear_Quadratic_Cubic(nums))
```

```
nums = [0,12,10,0,-12,-20]
```

```
print("\nOriginal Sequence:",nums)
```

```
print("Check the said sequence is Linear, Quadratic or Cubic?")
```

```
print(Seq_Linear_Quadratic_Cubic(nums))
```

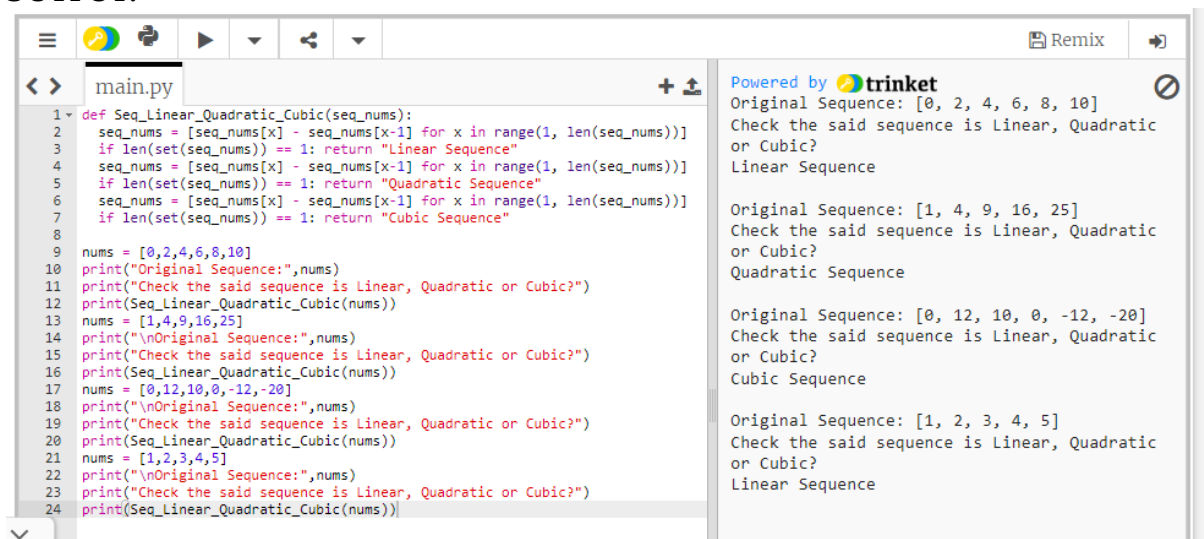
```
nums = [1,2,3,4,5]
```

```
print("\nOriginal Sequence:",nums)
```

```
print("Check the said sequence is Linear, Quadratic or Cubic?")
```


```
print(Seq_Linear_Quadratic_Cubic(nums))
```

OUTPUT:



The screenshot shows a Trinket.io code editor with a file named 'main.py'. The code defines a function 'Seq_Linear_Quadratic_Cubic' that checks if a sequence is Linear, Quadratic, or Cubic based on the second differences. It then tests three sequences: [0, 2, 4, 6, 8, 10], [1, 4, 9, 16, 25], and [0, 12, 10, 0, -12, -20]. The output on the right shows the results for each sequence: Linear, Quadratic, and Cubic respectively.

```
1 def Seq_Linear_Quadratic_Cubic(seq_nums):
2   seq_nums = [seq_nums[x] - seq_nums[x-1] for x in range(1, len(seq_nums))]
3   if len(set(seq_nums)) == 1: return "Linear Sequence"
4   seq_nums = [seq_nums[x] - seq_nums[x-1] for x in range(1, len(seq_nums))]
5   if len(set(seq_nums)) == 1: return "Quadratic Sequence"
6   seq_nums = [seq_nums[x] - seq_nums[x-1] for x in range(1, len(seq_nums))]
7   if len(set(seq_nums)) == 1: return "Cubic Sequence"
8
9 nums = [0,2,4,6,8,10]
10 print("Original Sequence:",nums)
11 print("Check the said sequence is Linear, Quadratic or Cubic?")
12 print(Seq_Linear_Quadratic_Cubic(nums))
13 nums = [1,4,9,16,25]
14 print("\nOriginal Sequence:",nums)
15 print("Check the said sequence is Linear, Quadratic or Cubic?")
16 print(Seq_Linear_Quadratic_Cubic(nums))
17 nums = [0,12,10,0,-12,-20]
18 print("\nOriginal Sequence:",nums)
19 print("Check the said sequence is Linear, Quadratic or Cubic?")
20 print(Seq_Linear_Quadratic_Cubic(nums))
21 nums = [1,2,3,4,5]
22 print("\nOriginal Sequence:",nums)
23 print("Check the said sequence is Linear, Quadratic or Cubic?")
24 print(Seq_Linear_Quadratic_Cubic(nums))
```

Powered by  trinket

Original Sequence: [0, 2, 4, 6, 8, 10]
Check the said sequence is Linear, Quadratic or Cubic?
Linear Sequence

Original Sequence: [1, 4, 9, 16, 25]
Check the said sequence is Linear, Quadratic or Cubic?
Quadratic Sequence

Original Sequence: [0, 12, 10, 0, -12, -20]
Check the said sequence is Linear, Quadratic or Cubic?
Cubic Sequence

Original Sequence: [1, 2, 3, 4, 5]
Check the said sequence is Linear, Quadratic or Cubic?
Linear Sequence