Python_basic_pragramming_22

In [ ]:
```
1.Create a function that takes three parameters where:
x is the start of the range (inclusive).
y is the end of the range (inclusive).
n is the divisor to be checked against.

Return an ordered list with numbers in the range that are divisible by the third
parameter n.
Return an empty list if there are no numbers that are divisible by n. Examples:
list_operation(1, 10, 3) [3, 6, 9]
list_operation(7, 9, 2) [8]
list_operation(15, 20, 7) []
```

In [1]:
```
def list_operation(start,end,divisor):
    out_list = []
    for ele in range(start,end+1):
        if ele%divisor == 0:
            out_list.append(ele)
    print(f'Output: {out_list}')

list_operation(1, 10, 3)
list_operation(7, 9, 2)
list_operation(15, 20, 7)
```

```
Output: [3, 6, 9]
Output: [8]
Output: []
```

In [ ]:
```
2.Create a function that takes in two lists and returns True if the second list
follows the first list by one element, and False otherwise. In other words,
determine if the second list is the first list shifted to the right by 1.
Examples:
simon_says([1, 2], [5, 1]) True
simon_says([1, 2], [5, 5]) False
simon_says([1, 2, 3, 4, 5], [0, 1, 2, 3, 4]) True
simon_says([1, 2, 3, 4, 5], [5, 5, 1, 2, 3]) False

Notes:
1. Both input lists will be of the same length, and will have a minimum length
   of 2.
2. The values of the 0-indexed element in the second list and the n-1th indexed
   element in the first list do not matter.
```

```
In [2]: def simon_says(in_list_1,in_list_2):
            if len(in_list_1) == len(in_list_1) and len(in_list_1) >=2 and
            len(in_list_1) >=2:
                if(in_list_1[:-1] == in_list_2[1:]):
                    print(f'{in_list_1,in_list_2} {True}')
                else:
                    print(f'{in_list_1,in_list_2} {False}')

        simon_says([1, 2], [5, 1])
        simon_says([1, 2], [5, 5])
        simon_says([1, 2, 3, 4, 5], [0, 1, 2, 3, 4])
        simon_says([1, 2, 3, 4, 5], [5, 5, 1, 2, 3])
```

```
([1, 2], [5, 1]) True
([1, 2], [5, 5]) False
([1, 2, 3, 4, 5], [0, 1, 2, 3, 4]) True
([1, 2, 3, 4, 5], [5, 5, 1, 2, 3]) False
```

```
In [ ]: 3.A group of friends have decided to start a secret society. The name will be
        the first letter of each of their names, sorted in a alphabetical order ?
        Create a function that takes in a list of names and returns the name of the
        secret society ?

        Examples:
        society_name(["Adam", "Sarah", "Malcolm"]) "AMS"
        society_name(["Harry", "Newt", "Luna", "Cho"]) "CHLN"
        society_name(["Phoebe", "Chandler", "Rachel", "Ross", "Monica", "Joey"])
```

```
In [3]: def society_name(in_list):
            out_string = []
            for ele in in_list:
                out_string.append(ele[0])
            output = ''.join(sorted(out_string))
            print(f'{in_list} {output}')

        society_name(["Adam", "Sarah", "Malcolm"])
        society_name(["Harry", "Newt", "Luna", "Cho"])
        society_name(["Phoebe", "Chandler", "Rachel", "Ross", "Monica", "Joey"])
```

```
['Adam', 'Sarah', 'Malcolm'] AMS
['Harry', 'Newt', 'Luna', 'Cho'] CHLN
['Phoebe', 'Chandler', 'Rachel', 'Ross', 'Monica', 'Joey'] CJMPRR
```

```
In [ ]: 4.An isogram is a word that has no duplicate letters. Create a function that
        takes a string and returns either True or False depending on whether or not
        it's an "isogram".
        Examples:
        is_isogram("Algorism") True
        is_isogram("PasSword") False
        # Not case sensitive.
        is_isogram("Consecutive") False

        Notes:
        Ignore letter case (should not be case sensitive).
        All test cases contain valid one word strings.
```

In [4]:
```
def is_isogram(in_string):
    lower_in_string = in_string.lower()
    if len(lower_in_string) == len(set(lower_in_string)):
        print(f'{in_string} {True}')
    else:
        print(f'{in_string} {False}')

is_isogram("Algorism")
is_isogram("PasSword")
is_isogram("Consecutive")
```

```
Algorism True
PasSword False
Consecutive False
```

In [ ]:
```
5.Create a function that takes a string and returns True or False,
depending on whether the characters are in order or not ?
Examples:
is_in_order("abc") True
is_in_order("edabit") False
is_in_order("123") True
is_in_order("xyzz") True

Notes:
You don't have to handle empty strings.
```

In [5]:
```
def is_in_order(in_string):
    in_string_sorted = ''.join(sorted(in_string))
    if in_string == in_string_sorted:
        print(f'{in_string} {True}')
    else:
        print(f'{in_string} {False}')

is_in_order("abc")
is_in_order("edabit")
is_in_order("123")
is_in_order("xyzz")
```

```
abc True
edabit False
123 True
xyzz True
```