Programming Lesson 3: List Comprehensions—Jali Purcell

Use Python Idle for this assignment.

When we run this expression in the shell:

```
>>> print([x*x for x in range(1,15) if x%2==0])
[4, 16, 36, 64, 100, 144, 196]
```

we get a list of the squares of the even numbers >=1 and <15. The expression

```
[x*x for x in range(1,15) if x%2==0]
```

is called a list comprehension.

The syntax for a list comprehension is

[expression-using-variable **for** variable **in** list-of-values **if** condition-on-variable]

which is equivalent to “make a list of the results of evaluating expression-using-variable for each value in the list-of-values for which the condition-on-variable is true.” That is, it returns the list of values obtained by applying the expression to each member of the list for which the condition is true. Note: the if part is optional.

1. What list prints in these examples? Write what you think will print, then run to code to see if you get those results.

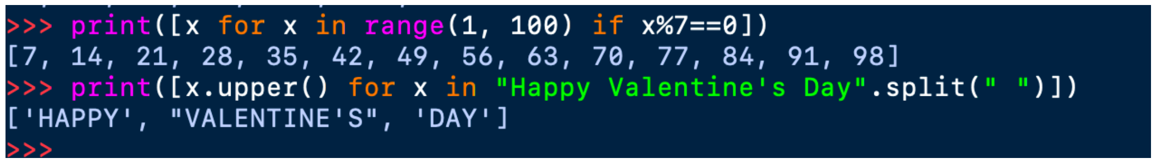
   a. print([x for x in range(1,100) if x%7==0])

      prints all numbers divisible by 7 up to 100, so 7, 14, 21, 28, 35, 42, 49,56, 63, 70, 77, 84, 91, 98

   b. print([x.upper() for x in "Happy Valentine’s Day".split(" ")])

      prints “Happy Valentine’s Day” in all caps, split at the spaces

   Paste here your transcript:



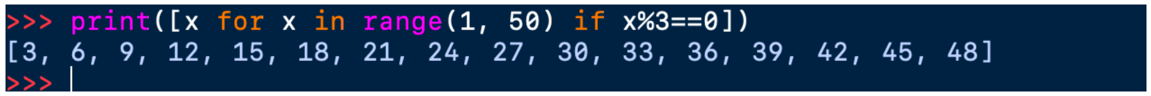
```
>>> print([x for x in range(1, 100) if x%7==0])
[7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98]
>>> print([x.upper() for x in "Happy Valentine's Day".split(" ")])
['HAPPY', "VALENTINE'S", 'DAY']
>>>
```

2. Use a list comprehension to create a list of the numbers from 1 to 50 that are divisible by 3. Paste here your comprehension along with a transcript showing it works correctly.

   Comprehension:

   print([x for x in range(1, 50) if x%3==0])

   Transcript:



```
>>> print([x for x in range(1, 50) if x%3==0])
[3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48]
>>> |
```

3. Write a list comprehension to make a list of all of the vowels in a string called mystring. Paste here your comprehension along with a transcript showing it works correctly.
   (Hint: the phrase **if letter in "aeiouAEIOU"** is a handy phrase to use.)

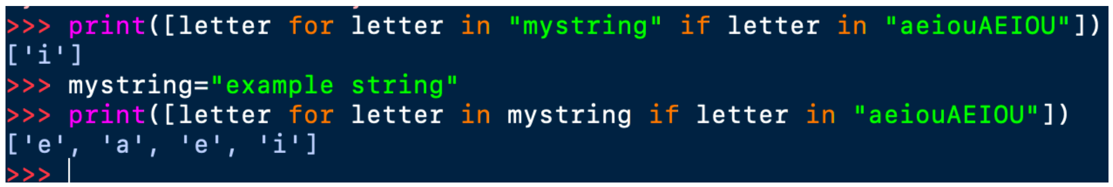
   Comprehension:

   print([letter for letter in “mystring” if letter in “aeiouAEIOU”])

   Or, if mystring is a variable string that is given beforehand,

   mystring=”example string”
   print([letter for letter in mystring if letter in “aeiouAEIOU”])

   Transcript:

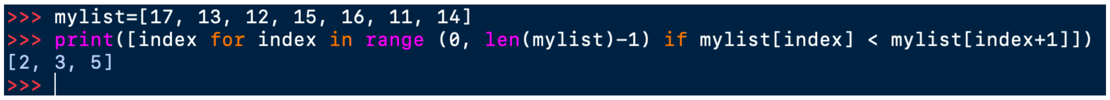


```
>>> print([letter for letter in "mystring" if letter in "aeiouAEIOU"])
['i']
>>> mystring="example string"
>>> print([letter for letter in mystring if letter in "aeiouAEIOU"])
['e', 'a', 'e', 'i']
>>> |
```

4. Suppose we have a list of numbers such as [17, 13, 12, 15, 16, 11, 14] and we want to find **the location of the first number in the list that is smaller than its successor**. In this example, 17 > 13 > 12 but 12 < 15, so we want position 2 (position 0 is 17, position 1 is 13, and position 2 is 12). We want to write a function that returns this position or returns -1 if the list is in descending order.

   a. First, complete the blanks to create a list of positions after which the list increases:

```
>>> mylist = [17, 13, 12, 15, 16, 11, 14]
>>> print([index for index in range( 0, len(mylist)-1) ) if mylist[index] < mylist[index+1]])
[2, 3, 5]
```



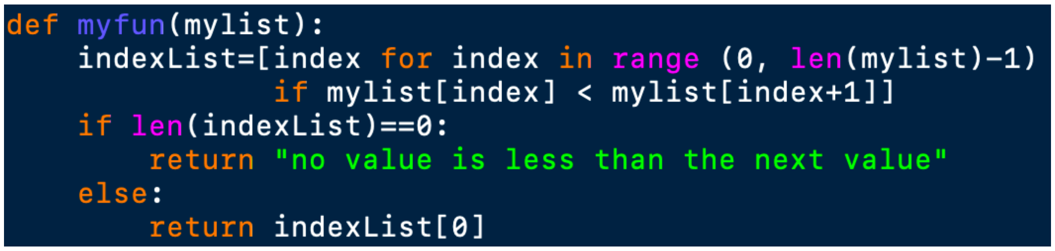
```
>>> mylist=[17, 13, 12, 15, 16, 11, 14]
>>> print([index for index in range (0, len(mylist)-1) if mylist[index] < mylist[index+1]])
[2, 3, 5]
>>> 
```

b. Now couch that comprehension in a function that returns the position we want.

def myfun(mylist):
  indexList = [index for index in range(<u>0, len(mylist)-1</u>) if mylist[index] < <u>mylist[index+1]</u>]
  if len(indexList)==0:
      return <u>“no value is less than the next value”</u>
  else:
      <u>return indexList[0]</u>

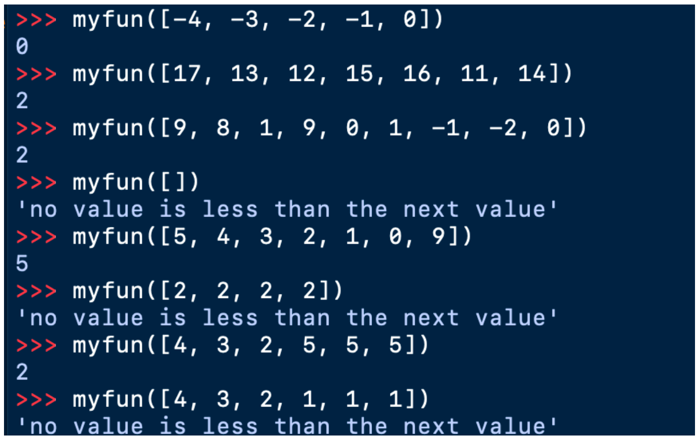
Paste here your final function and several examples showing that it works correctly.

Code:



```
def myfun(mylist):
    indexList=[index for index in range (0, len(mylist)-1)
               if mylist[index] < mylist[index+1]]
    if len(indexList)==0:
        return "no value is less than the next value"
    else:
        return indexList[0]
```

Transcript showing example calls:



```
>>> myfun([-4, -3, -2, -1, 0])
0
>>> myfun([17, 13, 12, 15, 16, 11, 14])
2
>>> myfun([9, 8, 1, 9, 0, 1, -1, -2, 0])
2
>>> myfun([])
'no value is less than the next value'
>>> myfun([5, 4, 3, 2, 1, 0, 9])
5
>>> myfun([2, 2, 2, 2])
'no value is less than the next value'
>>> myfun([4, 3, 2, 5, 5, 5])
2
>>> myfun([4, 3, 2, 1, 1, 1])
'no value is less than the next value'
```

5. We can use a list comprehension to create a list of objects of a class. For example, run this code:

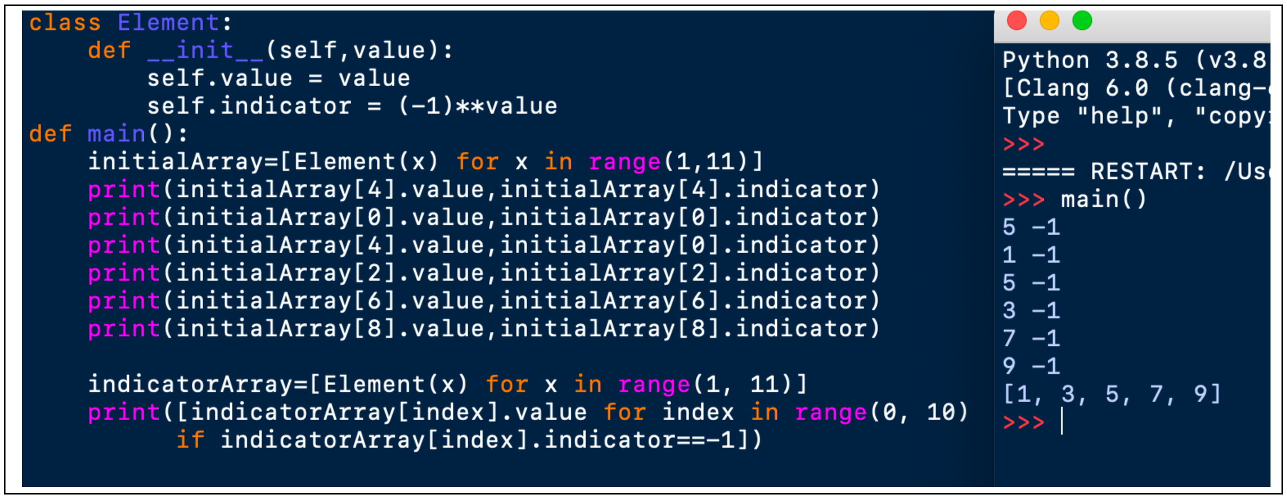
```
class Element:
  def __init__(self,value):
    self.value = value
    self.indicator = (-1)**value
def main():
  initialArray=[Element(x) for x in range(1,11)]
  print(initialArray[4].value,initialArray[4].indicator)
```

Now add a little code to the main function to print the list of values of elements for which the indicator is -1. Use a list comprehension. Paste here your final code and show that it works correctly.
Your code:

```
indicatorArray=[Element(x) for x in range(1, 11)]
print([indicatorArray[index].value for index in range(0, 10) if indicatorArray[index].indicator==-1])
```

Transcript showing it works correctly:

NOTE: I added additional print statements to check the values and indicator's, to make sure the array printed by my list comprehension was correct.



```
class Element:
    def __init__(self,value):
        self.value = value
        self.indicator = (-1)**value
def main():
    initialArray=[Element(x) for x in range(1,11)]
    print(initialArray[4].value,initialArray[4].indicator)
    print(initialArray[0].value,initialArray[0].indicator)
    print(initialArray[4].value,initialArray[0].indicator)
    print(initialArray[2].value,initialArray[2].indicator)
    print(initialArray[6].value,initialArray[6].indicator)
    print(initialArray[8].value,initialArray[8].indicator)

    indicatorArray=[Element(x) for x in range(1, 11)]
    print([indicatorArray[index].value for index in range(0, 10)
          if indicatorArray[index].indicator==-1])
```

```
Python 3.8.5 (v3.8
[Clang 6.0 (clang-
Type "help", "copy
>>>
===== RESTART: /Us
>>> main()
5 -1
1 -1
5 -1
3 -1
7 -1
9 -1
[1, 3, 5, 7, 9]
>>> 
```