Lesson 09

Asg 9.2

Lists

Make sure to run the code in the following cell before you start the assignment!!</i>

In [1]:
```
# set up notebook to display multiple output in one cell

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

print("The notebook is now set up to display multiple output in one cell'")
```

```
The notebook is now set up to display multiple output in one cell'
```

Question #1::

Note: For each part below, you are allowed only one statement to accomplish the requested task and print out the result.

Given: our_list = [31.6, "Honda", ["humility", "curiousity", "empathy"], "pi", 26, ["3", 2, '1'], "3", 2, '1']

a. write code to find the number of items in our_list

b. write code to access the third item of our_list

c. write code that uses a negative index to access the last item of our_list

d. write code to access the fifth, sixth, and seventh items of our_list

e. write code that uses negative indexes to access the second and third items of our_list

f. write code that incorporates exponentiation, the modulus operater, and other arithmetic operators to access the fourth element of our_list

g. write code that incorporates the len() function to access the sixth element of our_list

h. write code that does not includ a start index to access the first four elements of our_list

i. write code that does not include a stop index to print out the last three elelements of our_list

j. write code to access the second and third items of the first nested list in our_list

k. write code to determine if 'humility" is an item in our_list

l. write code to determine if ["3", 2, '1'] is an item in our_list

m. write a print statement that summarizes what you discovered from Parts k and l

n. write code to clone our_list and store the result in a variable named our_list_clone

o. write code to see if our_list equals our_list_clone (i.e. do they contain the exact same items?)

p. write code to see if our_list references the same object as our_list_clone

q. write a print statement that summarizes what you discovered from Parts o and p

r. create a Markdown cell where you include a reference diagram to support your answer to Part q

(Suggestion: You may want to use the link below in conjunction with the snipping tool / screen shot key to help you do this.)

Python Tutor

Answers for Question #1

In [1]:

```
our_list = [31.6, "Honda", ["humility", "curiousity", "empathy"], "pi", 26, ["3", 2, '

# letter a
print(len(our_list))
print("\n")

#letter b
print(our_list[2])
print("\n")

#letter c
print(our_list[-1])
print("\n")

#letter d
print(our_list[4:7])
print("\n")


#letter e
print(our_list[-8:-6])
print("\n")


#letter f
print("The formula for the area of a circle is %s*r**2."%(our_list[3]))
print("\n")

#letter g
print(our_list[len(our_list) - 4])
print("\n")

#letter h
```

```
print(our_list[:4])
print("\n")

#letter i
print(our_list[4:])
print("\n")

#letter j
print(our_list[2][1:3])
print("\n")

#letter k
print('humility' in our_list)
print("\n")

#letter l
print(["3", 2, '1'] in our_list)
print("\n")

#letter m
print("The in function does not check whether the lists inside of lists have the eleme
print("\n")

#letter n
our_list_clone = our_list[:]

#letter o
print(our_list_clone == our_list)
print("\n")

#letter p
print(our_list is our_list_clone)
print("\n")

#letter q
print("Despite the fact our_list has the same elements as our_list_clone, our_list doe
print("\n")
```

```
9

['humility', 'curiousity', 'empathy']

1

[26, ['3', 2, '1'], '3']

['Honda', ['humility', 'curiousity', 'empathy']]

The formula for the area of a circle is pi*r**2.

['3', 2, '1']

[31.6, 'Honda', ['humility', 'curiousity', 'empathy'], 'pi']

[26, ['3', 2, '1'], '3', 2, '1']

['curiousity', 'empathy']

False

True

The in function does not check whether the lists inside of lists have the element we a
re looking for.

True

False

Despite the fact our_list has the same elements as our_list_clone, our_list does not r
eference the same object as our_list_clone. That is why the answer to letter p is fals
e.
```

Question #2::

nba =['Bucks', 'Warriors', 'Nets', 'Lakers']
mlb = ["Brewers", 'Giants', 'Yankees', 'Dodgers']

a. write code to produce a list named pro_teams where ...

pro_teams = ['Bucks', 'Warriors', 'Nets', 'Lakers', "Brewers", 'Giants', 'Yankees', 'Dodgers', 'Packers', 'Bears']

b. write code to print out pro_teams

c. write code to produce a list named nba_twice where ...

nba_twice = ['Bucks', 'Warriors', 'Nets', 'Lakers','Bucks', 'Warriors', 'Nets', 'Lakers']

d. write code to print out nba_twice

e. write code to produce a list named pro_teams_2 where ...

pro_teams_2 = ['Bucks', 'Warriors', 'Nets', 'Lakers', "Brewers", 'Giants', 'Yankees', 'Dodgers', 'Packers', 'Bears','Packers', 'Bears']

f. write code to print out pro_teams_2

Answers for Question #2

In [40]:
```
nba =['Bucks', 'Warriors', 'Nets', 'Lakers']
mlb = ["Brewers", 'Giants', 'Yankees', 'Dodgers']
nfl = ['Packers', 'Bears']

#letter a
pro_teams = nba + mlb + nfl
```

```
#letter b
print(pro_teams)
print("\n")

#letter c
nba_twice = nba * 2

#letter d
print(nba_twice)

#letter e
pro_teams_2 = pro_teams + nfl

#letter f
print(pro_teams_2)
```

```
['Bucks', 'Warriors', 'Nets', 'Lakers', 'Brewers', 'Giants', 'Yankees', 'Dodgers',
'Packers', 'Bears']


['Bucks', 'Warriors', 'Nets', 'Lakers', 'Bucks', 'Warriors', 'Nets', 'Lakers']
['Bucks', 'Warriors', 'Nets', 'Lakers', 'Brewers', 'Giants', 'Yankees', 'Dodgers',
'Packers', 'Bears', 'Packers', 'Bears']
```

Question #3::

Create a list called mylist with the following 9 items: "I, am, excited, to, be, learning, about, Python, lists!".

Part 1: Beginning with the empty list ...

i. use the append() method to add the first three items to mylist
ii. use the extend() method to add the next three items to mylist
iii. use concatenation to add the last three items to mylist
iv. write code to print out the final version of mylist
v. write code to print out the type of the final version of mylist
vi. write code to print out a blank line

Part 2: Using the results from Part 1 above ...

i. convert mylist to a string named mystring
ii. write code to print out mystring
ii. write code to print out the type of mystring

Answers for Question #3

In [11]:
```python
mylist = []
mylist.append('I')
mylist.append('am')
mylist.append('excited')
print(mylist)
print("\n")
extended_portion_of_list = ['to', 'be', 'learning']
mylist.extend(extended_portion_of_list)
print(extended_portion_of_list)
print("\n")
concatenated_portion_of_list = ['about', 'python', 'lists!']
mylist += concatenated_portion_of_list
print(mylist)
print("\n")

#Part 2
mystring = ' '.join(mylist)
print(mystring)
print("\n")
print(type(mystring))
```

```
['I', 'am', 'excited']


['to', 'be', 'learning']


['I', 'am', 'excited', 'to', 'be', 'learning', 'about', 'python', 'lists!']


I am excited to be learning about python lists!


<class 'str'>
```

Question #4::

spam_phrases_1 = ['ADDITIONAL INCOME', 'Earn extra cash', 'Make $', 'Work at home', 'Incredible deal', 'No hidden Costs']

a. write code to print out spam_phrases_1
b. write code to print out the type of spam_phrases_1
c. write code to print out a blank line

d. write code to assign the value 'Eliminate bad credit' as the fourth value in spam_phrases_1 and store the result in a variable named spam_phrases_2

e. write code to print out spam_phrases_2
f. write code to print out a blank line

g. write code to remove the sixth item from spam_phrases_2 and store the result in a variable named spam_phrases_3
h. write code to print out spam_phrases_3
i. write code to print out a blank line

j. write code to add the following phrases to spam_phrases_3 and store the result in a variable named spam_phrases_4: 'Lower monthly payment', 'Multi level marketing', "This isn't spam", 'One time mailing'
k. write code to print out spam_phrases_4
l. write code to print out a blank line

m. write code to remove the second, third, and fourth terms from spam_phrases_4 and store the result in a variable named spam_phrases_5
n. write code to print out spam_phrases_5
o. write code to print out a blank line

Given: new_spam = ['No medical exams', 'Reverses aging', 'Online pharmacy']

p. use concatenation to add new_spam to spam_phrases_5 and store the result in a variable named spam_phrases_6
q. write code to print out spam_phrases_6
r. write code to print out a blank line

s. write code to remove 'ADDITIONAL INCOME' and 'Lower monthly payment' from spam_phrases_6 and store the result in a variable named spam_phrases_7
t. write code to print out spam_phrases_7

Answers for Question #4

In [26]:
```
spam_phrases_1 = ['ADDITIONAL INCOME', 'Earn extra cash', 'Make $', 'Work at home',
print(type(spam_phrases_1))
print("\n")
spam_phrases_1[3] = "Eliminate bad credit"
spam_phrases_2 = spam_phrases_1[:]
print(spam_phrases_2)
print(type(spam_phrases_2))
print("\n")
```

```
del spam_phrases_2[5]
spam_phrases_3 = spam_phrases_2[:]
print(spam_phrases_3)
print("\n")

portion_to_be_extended = ['Lower monthly payment', 'Multi level marketing', 'This isn
spam_phrases_3.extend(portion_to_be_extended)
spam_phrases_4 = spam_phrases_3[:]
print(spam_phrases_4)
print("\n")

del spam_phrases_4[1:3]
spam_phrases_5 = spam_phrases_4[:]
print(spam_phrases_5)
print("\n")

new_spam = ['No medical exams', 'Reverses aging', 'Online pharmacy']
spam_phrases_6 = spam_phrases_5 + new_spam
print(spam_phrases_6)
print("\n")

spam_phrases_6.remove('ADDITIONAL INCOME')
spam_phrases_6.remove('Lower monthly payment')
spam_phrases_7 = spam_phrases_6[:]
print(spam_phrases_7)
```

```
<class 'list'>


['ADDITIONAL INCOME', 'Earn extra cash', 'Make $', 'Eliminate bad credit', 'Incredib
le deal', 'No hidden Costs']
<class 'list'>


['ADDITIONAL INCOME', 'Earn extra cash', 'Make $', 'Eliminate bad credit', 'Incredib
le deal']


['ADDITIONAL INCOME', 'Earn extra cash', 'Make $', 'Eliminate bad credit', 'Incredib
le deal', 'Lower monthly payment', 'Multi level marketing', "This isn't spam", 'One
time mailing']


['ADDITIONAL INCOME', 'Eliminate bad credit', 'Incredible deal', 'Lower monthly paym
ent', 'Multi level marketing', "This isn't spam", 'One time mailing']


['ADDITIONAL INCOME', 'Eliminate bad credit', 'Incredible deal', 'Lower monthly paym
ent', 'Multi level marketing', "This isn't spam", 'One time mailing', 'No medical ex
ams', 'Reverses aging', 'Online pharmacy']


['Eliminate bad credit', 'Incredible deal', 'Multi level marketing', "This isn't spa
m", 'One time mailing', 'No medical exams', 'Reverses aging', 'Online pharmacy']
```

Question #5::

greater_metro = ['BCHS', 'BEHS', 'Arrowhead', "SH', 'Muskego','Men Falls', 'Heritage Christian', 'Brookfield Academy', 'WA Hale', 'WA Central', 'Franklin']

a. write code to print greater_metro

b. write code that uses item assignment to replace 'Arrowhead' with 'Gtown' and 'Muskego' with "MUHS' in greater_metro

c. write code to print greater_metro

d. write code to that utilizes the empty list [ ] to delete 'Heritage Christian' and 'Brookfield Academy' from greater_metro

e. write code to print greater_metro

f. write code to simultaneously update greater_metro by inserting'Tosa East' and 'Tosa West' as the third and fourth items of greater_metro

g. write code to print greater_metro

h. write code that uses the del statement to delete 'WA Central' and 'Franklin' from greater_metro

i. write code to print greater_metro

j. write code that prints out greater_metro in alphabetical order

Answers for Question #5

In [44]:
```
greater_metro = ['BCHS', 'BEHS', 'Arrowhead', 'SH', 'Muskego','Men Falls', 'Heritage
print(greater_metro)
print("\n")

index_of_Arrowhead = greater_metro.index('Arrowhead')
greater_metro[index_of_Arrowhead] = 'Gtown'

index_of_Muskego = greater_metro.index('Muskego')
```

```python
greater_metro[index_of_Muskego] = 'MUHS'
print(greater_metro)
print("\n")


greater_metro_filtered = ['BCHS', 'BEHS', 'Arrowhead', 'SH', 'Muskego','Men Falls', |
greater_metro = list(filter(None, greater_metro_filtered))
print(greater_metro)
print("\n")

greater_metro[2] = 'Tosa East'
greater_metro[3] = 'Tosa West'
print(greater_metro)
print("\n")

index_of_WA_Central = greater_metro.index('WA Central')
del greater_metro[index_of_WA_Central]

index_of_Franklin = greater_metro.index('Franklin')
del greater_metro[index_of_Franklin]
print(greater_metro)
print("\n")

print(sorted(greater_metro))
```

```
['BCHS', 'BEHS', 'Arrowhead', 'SH', 'Muskego', 'Men Falls', 'Heritage Christian', 'B
rookfield Academy', 'WA Hale', 'WA Central', 'Franklin']


['BCHS', 'BEHS', 'Gtown', 'SH', 'MUHS', 'Men Falls', 'Heritage Christian', 'Brookfie
ld Academy', 'WA Hale', 'WA Central', 'Franklin']


['BCHS', 'BEHS', 'Arrowhead', 'SH', 'Muskego', 'Men Falls', 'WA Hale', 'WA Central',
'Franklin']


['BCHS', 'BEHS', 'Tosa East', 'Tosa West', 'Muskego', 'Men Falls', 'WA Hale', 'WA Ce
ntral', 'Franklin']


['BCHS', 'BEHS', 'Tosa East', 'Tosa West', 'Muskego', 'Men Falls', 'WA Hale']


['BCHS', 'BEHS', 'Men Falls', 'Muskego', 'Tosa East', 'Tosa West', 'WA Hale']
```

Question #6::

Create a markdown cell to identify / describe at least two important ways that list are different than strings.

Answers for Question #6

One major difference is that strings are immutable while lists are mutable. </br> Another major difference is that strings are a sequence of characters while lists can be a list of anything. </br>

Question #7::

a. run the following code and then explain in a Markdown cell what you observed about how the pop() list method works

alist_1 = [9, 18, 27, 36, 45, 54, 63, 72, 81, 90]
print(alist_1.pop())
print(alist_1)

print()

alist_2 = [9, 18, 27, 36, 45, 54, 63, 72, 81, 90]
print(alist_2.pop(4))
print(alist_2)

b. write code that will print out alist_1 in reverse order

c. days = "The days of the week are Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday."

... write code that uses the appropriate string method along with the delimiter "day" to convert the string days to a list

Answers for Question #7

In [63]:
```
# Code cell for Part a above

alist_1 = [9, 18, 27, 36, 45, 54, 63, 72, 81, 90]
#print(alist_1.pop())
#print(alist_1)

alist_2 = [9, 18, 27, 36, 45, 54, 63, 72, 81, 90]
#print(alist_2.pop(4))
#print(alist_2)
```

```
#Part B
print(alist_1[::-1])
print("\n")
#Part C
days = "The days of the week are Sunday, Monday, Tuesday, Wednesday, Thursday, Frid
listOfDays = days.split(' ')[6:]
listOfDays.remove('and')
print(listOfDays)
```

```
[90, 81, 72, 63, 54, 45, 36, 27, 18, 9]


['Sunday,', 'Monday,', 'Tuesday,', 'Wednesday,', 'Thursday,', 'Friday,', 'Saturda
y.']
```

Markdown cell for Part a

Pop removes the element at the specified position

Question #8::

Given:

x = (2 ** 3 ** 2 // 50 / 5)
y = (((3.14 + 2.71) * 100) ** 0 + 1)
z = 5

For each of the Parts a - d below create a Markdown cell to provide your answer. In those Markdown cells create reference diagrams to support your answer (Hint: use the Python Tutor link that is found below to help you create the reference diagrams.)

A code cell has also been provided that contains some initial lines of code for you to run. Update this code cell with the proper code that you would need to support your answers to Parts a - d

first_list = [x, z]
second_list = [y, z]

a. Do the first_list and second_list contain the same elements?

b. Do first_list and second_list reference the same object?

Add the following line of code in the code cell that has been provided ...

first_list = second_list

c. Do first_list and second_list now reference the same object?

Add the following line of code in the code cell that has been provided ...

first_list[1] = 15

d. What would the output be for the following print command? What do you notice when you run this print command?

print(second_list)

Python Tutor

Answers for Question #8

In [65]:
```
# Code cell for Question #8

x = (2 ** 3 ** 2 // 50 / 5)
y = (((3.14 + 2.71) * 100) ** 0 + 1)
z = 5

first_list = [x, z]
second_list = [y, z]
first_list = second_list
print(first_list)
print(second_list)
```

```
[2.0, 5]
[2.0, 5]
```

Markdown cell for 8 a

Yes, first_list and second_list contain the same element.

Markdown cell for 8 b

No, first_list and second_list reference different objects

Markdown cell for 8 c

Yes, now first_list and second_list reference the same objects

Markdown cell for 8 d

The first_list and second_list both got updated because they point to the same list.

Question #9::

a. create a string named ds_advice that contains the following passage:

"Ben Franklin created a step-by-step plan to improve his writing. If you want to master data science, you can actually learn from what Franklin did. He broke everything down and set out to improve at every level. he set out to improve vocabulary, improve sentence structure, and improve organization. He also identified excellent writers and attempted to rewrite and mimic their work himself. Then he would compare his version to the original, so he could identify specific areas that he still needed to improve."

b. write code to print out ds_advice

c. write code to print out on a single line of output the number of characters in ds_advice and the data type of ds_advice

d. write code that uses the appropriate string method to convert ds_advice into a list named ds_advice_list

e. write code to print out ds_advice_list

g. write code to print out on a single line of output the number of items in ds_advice_list and the data type of ds_advice_list

h. write code to print out the position that the item 'improve' first occurs in ds_advice_list

i. write code to print out the number of times the item 'improve' occurs in ds_advice_list

</div>

Answers for Question #9

In [71]:
```
ds_advice = "Ben Franklin created a step-by-step plan to improve his writing. If y
print(ds_advice)
print("\n")

print(len(ds_advice), type(ds_advice))
ds_advice_list = ''.join(ds_advice)
print(ds_advice_list)
print("\n")
print(ds_advice_list.index('improve'))
print("\n")
print(ds_advice_list.count('improve'))
```

```
Ben Franklin created a step-by-step plan to improve his writing. If you want to m
aster data science, you can actually learn from what Franklin did. He broke every
thing down and set out to improve at every level. he set out to improve vocabular
y, improve sentence structure, and improve organization. He also identified excel
lent writers and attempted to rewrite and mimic their work himself. Then he would
compare his version to the original, so he could identify specific areas that he
still needed to improve.


511 <class 'str'>
Ben Franklin created a step-by-step plan to improve his writing. If you want to m
aster data science, you can actually learn from what Franklin did. He broke every
thing down and set out to improve at every level. he set out to improve vocabular
y, improve sentence structure, and improve organization. He also identified excel
lent writers and attempted to rewrite and mimic their work himself. Then he would
compare his version to the original, so he could identify specific areas that he
still needed to improve.


44


6
```

Question #10::

a. Create a list called myList with the following six items: 76, 92.3, “hello”, True, 4, 76.

Using the following guidelines, begin with the empty list and add 4 statements to create myList:

1st Statement: Use the append() method to add the first item to the list.
2nd Statement: Use the append() method to add the second item to the list.
3rd Statement: Use the extend() method to add the third and fourth items to the list.
4th Statement: Use concatenation to add the fifth and sixth items to the list.

b. append “apple” and 76 to the list and print the result

c. insert the value “cat” at position 3 and print the result

d. insert the value 99 at the start of the list and print the result

e. find the index of “hello” and print the result

f. count the number of 76s in the list and print the result

g. remove the first occurrence of 76 from the list and print the result

h. Remove True from the list using pop and index and print the result

Answer for Question #10

In [91]:
```
myList=[]
myList.append(76)
myList.append(92.3)
myList.extend(["hello", True])
myList += [4, 76]
myList.append('apple')
```

```python
myList.append(76)
print(myList)
print("\n")
myList[3:3] = ['cat']
print(myList)
print("\n")
myList[0:0] = [99]
print(myList)
print("\n")
print(myList.index("hello"))
print("\n")
print(myList.count(76))
print("\n")
myList.remove(76)
print(myList)
print("\n")
indexOfBool = myList.index(True)
myList.pop(indexOfBool)
print(myList)
print("\n")
```

```
[76, 92.3, 'hello', True, 4, 76, 'apple', 76]


[76, 92.3, 'hello', 'cat', True, 4, 76, 'apple', 76]


[99, 76, 92.3, 'hello', 'cat', True, 4, 76, 'apple', 76]


3


3


[99, 92.3, 'hello', 'cat', True, 4, 76, 'apple', 76]


[99, 92.3, 'hello', 'cat', 4, 76, 'apple', 76]
```

Note:

- Once you are satisfied with the results, submit your .ipynb notebook and a pdf and/or html file to Google Classroom.

-Your files should include all output, i.e. run each cell and save your file before submitting.

```
In [ ]:
```