LISTS

Introduction

PE4_1

1. A – C, determine the output displayed by the lines of code. Save your code as *PE4_1.py*.

A	`print("Python")`
	`print("Python"[0])`
	`print("Python"[-1])`
	`print("Python"[:])`
		B	`str = "Python 123"`
			`print(str)`
			`print(str[0])`
			`print(str[-1])`
			`print(str[:])`

Output		Output	

C	`strNum = "0, 1, 2, 3, 4, 5, 6, 7, 8, 9"`
	`print(strNum[1], strNum[-1], len(strNum))`
	`print(strNum[:len(strNum)])`
	`print(strNum[1]+strNum[-3])`		

Output			

Python Objects (1 of 2)

- The python documentation use the term **object** to refer to any instance of a data type.
- Python's Core objects are **numbers**, **strings**, **lists**, **tuples**, **files**, **sets** and **dictionary**.

Python Objects (2 of 2)

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.
- When choosing a collection type, it is useful to understand the properties of that type. Choosing the right type for a particular data set could mean retention of meaning, and, it could mean an increase in efficiency or security.

The List Object

- A list is an ordered sequence of Python objects
  - *Objects can be of any type*
  - *Objects do not have to be the same type.*
  - *Constructed by writing items enclosed in square brackets … items separated by commas.*

```
team = ["Seahawks", 2014, "CenturyLink Field"]
nums = [5, 10, 4, 5]
words = ["spam", "ni"]
```

Built-in Functions/Keywords/List Methods (1 of 4)

Ex1_lists.py

- https://www.w3schools.com/python/python_ref_functions.asp

```
team = ["Seahawks", 2014, "CenturyLink Field"]
nums = [5, 10, 4, 5]
words = ["spam", "ni"]
```

Functions or Methods	Example	Value	Description
len	len(words)	2	number of items in list
max	max(nums)	10	greatest (items must have same type)
min	min(nums)	4	least (items must have same type)
sum	sum(nums)	24	total (items must be numbers)

Built-in Functions/Keywords/List Methods (2 of 4)

Ex2_functions_methods.py

- https://www.w3schools.com/python/python_ref_list.asp

```
team = ["Seahawks", 2014, "CenturyLink Field"]
nums = [5, 10, 4, 5]
words = ["spam", "ni"]
```

Keywords or Methods	Example	Value	Description
del	del words[-1]	['spam']	removes item with stated index
pop	nums.pop(2)	[5, 10, 5]	removes the item at the specified position, and the popped valued can be returned
remove	nums.remove(5)	[10, 5]	removes first occurrence of the specified value
clear	team.clear()	[]	[] is the empty list

Built-in Functions/Keywords/List Methods (3 of 4)

Methods or Operators	Example	Value	Description
append	nums.append(7)	[10, 5, 7]	inserts object at end of list
insert	words.insert(1, “eggs")	[‘spam’, ‘eggs’]	insert new item before item of given index
extend	nums.extend([1, 2])	[10, 5, 7, 1, 2]	inserts new list’s items at end of list
count	nums.count(7)	1	number of occurrences of an object
index	nums.index(7)	2	index of first occurrence of an object
+	['a', 1] + [2, 'b']	['a', 1, 2, 'b']	concatenation; same as ['a', 1].extend([2,'b'])
*	[0] * 3	[0, 0, 0]	list repetition

Built-in Functions/Keywords/List Methods (4 of 4)

Functions or Methods	Example	Value	Description
sort	words.sort()	[‘eggs’, ‘spam’]	permanent alphabetical sort of the list
sort	nums.sort(reverse=True)	[10, 7, 5, 2, 1]	permanent alphabetical sort of the list in reverse
reverse	fruits.reverse()	[‘mango’, ‘apple’, ‘orange’]	reverses the order of the items
sorted	sorted(words)	[‘eggs’,’python’ ‘spam’]	temporary alphabetical sorted the copy of the list
sorted	sorted(words,reverse=True)	[‘python’, ‘java’, ‘c++’]	temporary alphabetical sorted the copy of the list in reverse

PE4_2

2. Use list methods to code below. Save the code as *PE4_2.py*.

a) Create an empty list called *n*.
b) Add 2 and 4 into the list.
c) Print the list.
d) Add 0, 1 and 3 in proper order.
e) Print the list.
f) Add 5 in proper order.
g) Print the list.
h) Remove 0 from the list.
i) Print the list.
j) Remove and print 2 from the list.
k) Print the list.
l) Remove and print 4 from the list.
m) Print the list.
n) Add all the removed numbers and print the sum.
o) Change the first item to 100 and last item to 9.9.
p) Copy the list *n* to a *newNum* list.
q) Clear the list *n* and print it.
r) Print the *newNum* list.
s) Delete the list n.

Example Output

```
[2, 4]

[0, 1, 2, 3, 4]

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

[1, 2, 3, 4, 5]

[1, 3, 4, 5]
2

[1, 3, 5]
4

Sum of all removed numbers = 6

[100, 3, 9.9]

Original list = []
New list = [100, 3, 9.9]
```

Slices (1 of 2)

Ex3a_listIndexes.py

- A slice of a list is a sub list specified with colon notation
  - *Analogous to a slice of a string*
- `grades = ['A', 'B', 'C', 'D', 'F']`

Slice Notation	Meaning
`grades[m:n]`	list consisting of the items of list1 having indices m through n-1
`grades[:]`	list consisting of the items of list1 having indices from the beginning to the end
`grades[m:]`	list consisting of the items of list1 having indices from m to the end
`grades[:n]`	list consisting of the items of list1 having indices from the beginning to n-1

Slices (2 of 2)

Ex3b_listSlices.py

- `grades = ['A', 'B', 'C', 'D', 'F']`

Example	Value
grades[1:3]	['B', 'C']
grades[-4:-2]	['B', 'C']
grades[:4]	['A', 'B', 'C', 'D']
grades[2:]	['C', 'D', 'F']
grades[:]	['A', 'B', 'C', 'D', 'F']
grades[1:len(grades)]	['B', 'C', 'D', 'F']
(grades[1:3])[1]	'C' (This expression is usually written as *grades [1:3][1])*
grades[3:2]	[], the list having no items; that is, the *empty list*
del grades	The list, *grades* is deleted

PE4_3 & PE4_4

- Write your codes and run

The *split and join* Methods (1 of 2)

Ex4_split_join.py

- The *split* and *join* are two methods that are inverses of each other
- The *split* method turns a single string into a list of substrings
- The string consisting of the comma character is called *separator* for the statement below
- *Split* method splits the string at the specified separator, and returns a list

```
>>> letters = "a,b,c"
>>> print(letters.split(','))
```

['a', 'b', 'c']

Note: The split method will play a vital role in PE4_5

The *split and join* Methods (2 of 2)

Ex4_split_join.py

- The *join* method is the inverse of the *split* method
- The *join* method turns a list of strings into a single string

```
>>> letters = ['a', 'b', 'c’]
>>> print(',’.join(letters))
```

a,b,c

PE4_5

- Write your codes and run

Copying Lists (1 of 2)

Ex5_listCopy.py

- Consider results of this program

```
list1 = ['a', 'b']  # Lists are mutable objects
list2 = list1       # list2 will point to the same memory location as list1
list2[1] = 'c'      # Changes the value of the second item in the list object
print(list1)
```

[Run]

```
['a', 'c']
```

- All because lists are mutable

Copying Lists (2 of 2)

- Now note change in line 2

```
list1 = ['a', 'b']  # Lists are mutable objects
list2 = list(list1) # list2 now points to different memory location
list2[1] = 'c'      # Changes the value of the second item in the list object
print(list1)
```

[Run]

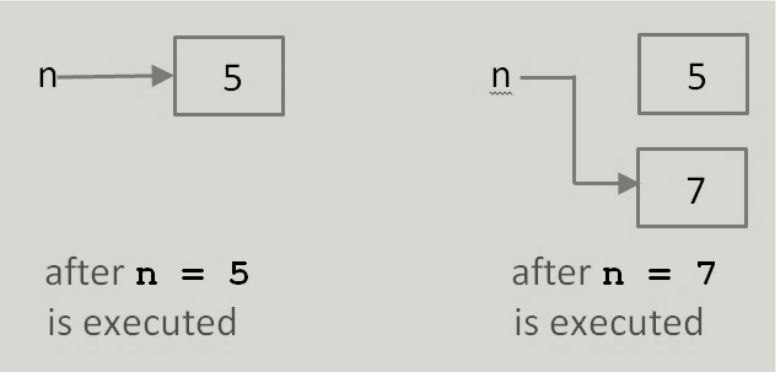
```
['a', 'b']
```

- Third line of code will not affect memory location pointed to by list1

Immutable and Mutable Objects (1 of 4)

- An object is an entity
  - *Holds data.*
  - *Has operations and/or methods that can manipulate the data.*
- When variable created with assignment statement
  - *Value on the right side becomes an object in memory*
  - *Variable references (points to) object*

```
n = 5
n = 7
```



Immutable and Mutable Objects (2 of 4)

- When list altered
  - *Changes made to the object in list's memory location*
- Contrast when value of variable is number, string, or tuple … when value changed,
  - *Python designates a new memory location to hold the new value*
  - *And the variable references that new object*

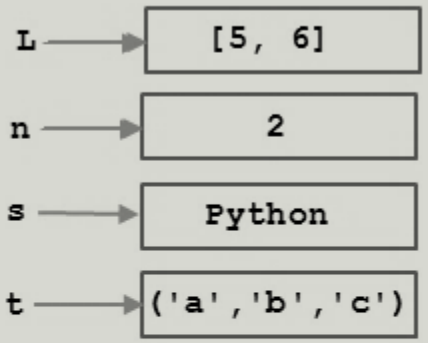
Immutable and Mutable Objects (3 of 4)

- Another way to say this
  - *Lists can be changed in place*
  - *Numbers, strings, and tuples cannot*
- Objects changed in place are mutable
- Objects that cannot be changed in place are immutable

Immutable and Mutable Objects (4 of 4)

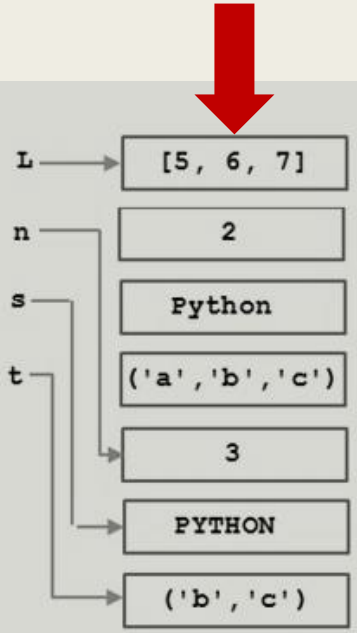
Ex6_mutable_immutable.py

```
L = [5, 6]
n = 2
s = "Python"
t = ('a','b','c')
L.append(7)
n += 1
s = s.upper()
t = t[1:]
```



After first 4 lines of code have been executed

```
L = [5, 6]
n = 2
s = "Python"
t = ('a','b','c')
L.append(7)
n += 1
s = s.upper()
t = t[1:]
```



After all 8 lines of code have been executed

Summary (1 of 2)

- A list is an ordered sequence of items. Items are referred by their position (called their index) starting at 0 from the left end or their position (starting at -1) from their right end.
- Slices of a list are defined in much the same way as slices of strings.
- An object whose data cannot be modified in place is said to be immutable. Numbers, strings, and tuples are immutable.
- Lists are mutable.

Summary (2 of 2)

- **Keyword:** *del*
- **List functions:** *list, len, max, min, sum, sorted*
  - sum() only applies to lists of numbers
  - sorted() only applies to lists containing items with the same data types
  - When ***max*** and ***min*** are applied to lists containing strings, lexicographical order is used to compare two strings.
- **List methods:** *append, clear, copy, count, extend, index, insert, pop, remove, reverse, sort, split, join*
- The ***list*** function is used to creating a new list. For instance,
  the value of ***list***((‘p’, ‘y’)) is [‘p’, ‘y’] # convert a tuple to a list
  the value of ***list***(“py”) is [‘p’, ‘y’] # convert a string to a list

Quiz 4

- Quiz 4A has 10 questions in 15 minutes, 10 pts
  - *10 multiple choice/true or false questions, 1 pt. for each question*
  - *Quiz 3A has two attempt, the higher grade will be selected*
  - *Submit Quiz 4A (at least 1-minute)* ***before*** *the due time to Blackboard*

- Quiz 4B has 2 code questions, 15 pts
  - *Write the Python code based on the given question*
  - *Each question will be given during the* ***last*** *10-minute of each session of week 4*
  - *Quiz 4B-1 on session A, and Quiz 4B-2 on session B*
  - *Quiz 4B has one attempt*

Discussion Board: DB4

- Instruction:
  1) Choose any **one** of the questions from **PE4_1**, any **one** of the questions from **PE4_6**, and any **one** of the questions from **PE4_7**. Please **avoid** selecting the exact same questions. Make sure to indicate the **question #** you're working on in the thread title as soon as you open your thread. Then you can **explain and edit your questions** (1.2 pt).
  2) After posting your explanation, check each other's answers, and ask questions or make comments (0.3 pt).
  3) Submit your posts before the due date. Let's learn from each other.