Tutorial II

Intermediate Python Programming

5 – Lists.

Resources created by Andy Colley (@MrAColley)

Learning Goals/Objectives

Be able to read, comprehend, trace, adapt and create Python code that:

- Uses lists to **store data**
- **Outputs** a range of items from a list
- **Searches** a list to find an item

repl.it

Output a Range of Items

Variable or List?

Variable - stores **one** piece of data with an identifier.

```
player1 = Mary
player2 = Sean
player3 = Atif
```

List - stores **more than one** piece of data with the same identifier.

```
players = [“Mary”, “Sean”, “Atif”]
```

repl.it

List Range - How To Code

We can output a range from a list by customising the `print` command.

```
fruits = [“apple”, “banana”, “grapes”, “strawberry”, “orange”]

print(fruits[2:4])
```

This will start at index 2 (grapes).

This will end at index 3 (strawberry). The end number in the brackets is NOT included.

Use the list name in the print command.

repl.it

List Range - Variations

```
fruits = [“apple”, “banana”, “grapes”, “strawberry”, “orange”]

print(fruits[:4])


print(fruits[2:])
```

Outputs from the beginning up to index 3 (strawberry).
The end number in the brackets is **NOT** included.

Outputs from index 2 (grapes) to the end of the list.

repl.it

Task - Predict & Run

```
1   # Task Predict & RuntimeError
2
3   # Add comments to the code to predict how it will work.
4   # Run the code to check your predictions
5
6   rockStars = ["John","Paul","George","Ringo","Freddie","Brian","John","Roger"]
7
8   print(rockStars[3:6])
9
10  print(rockStars[:5])
11
12  print(rockStars[4:])
```

repl.it

Task - Investigate

```
1   # Task Investigate
2
3   # Answer the questions about the code below
4
5   rockStars = ["John","Paul","George","Ringo","Freddie","Brian","John","Roger"]
6
7   listLength = len(rockStars)
8
9   print(rockStars[3:listLength])
10
11  # What does the len() function do?
12
13  # If 'Axl' was appended to the list, what would be the effect on the listLength variable?
14
15  # What would be the effect on the output?
16
```

repl.it

Task - Modify

```
18  # Task - Modify
19
20  # Copy the code and adapt it so that:
21
22  # It asks the user to input a number between 0 and the last index of the list (hint - you'll need to
    subtract 1 from the length of the list).
23
24  # It validates the input so that users can't enter a number smaller than 0 or bigger than index of
    the last item in the list.
25
26  # It asks the user to input a second number between the first number input + 1 and the last index of
    the list (hint - you'll need to subtract 1 from the length of the list).
27
28  # It validates the input so that users have to input a number bigger than the first input and less
    than or equal to  than index of the last item in the list.
29
30  # It outputs the items in the list between the two numbers input.
31
```

repl.it

Task - Make

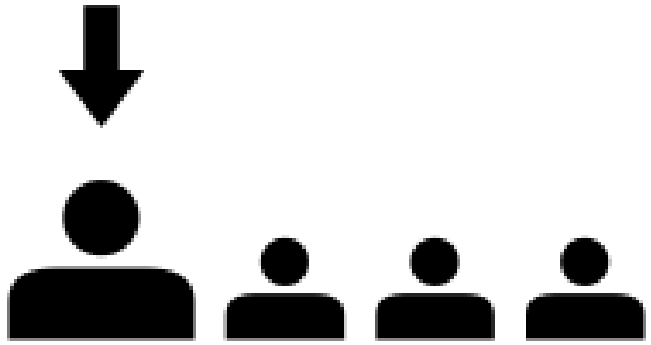
```
1   # Task - Make
2
3   # Write a program that:
4
5   # Stores 10 names in a list
6
7   # Asks the user to input 1 to choose output range, 2 to choose output from a point to the end
    of the list or 3 to choose output from the beginning of the list to a point.
8
9   # Validates the input and does not continue until it is suitable.
10
11  # Depending on the option chosen asks the user to input relevant start/end points.
12
13  # Outputs the relevant items from the list.
14
15  # Extra challenge - code the three different outputs as separate functions and call them when
    needed.
16
```

repl.it

Search A List

Linear

‘Resembling a line’.



Linear Search

Searching each item of data one after the other, starting with the first.

(Move along the line)

repl.it

Resources created by Andy Colley (@MrAColley)

0 1 2 3 4

Pets[]

repl.it

Search A List - How To Code - Method 1

Using a **while** loop

1. Initialise a **counter** variable to 0 to keep track of how many times the loop has run.

```
counter = 0
found   = False
```

2. Initialise a **found** variable to false. We will change this to true when/if we find our item in the list.

repl.it

Search A List - How To Code - Method 1

Using a **while** loop

```
counter = 0
found  = False

while counter < len(list):
```

3. Start a while loop. Set the condition to be while the counter variable is less than the length of the list.

repl.it

Search A List - How To Code - Method 1

Using a **while** loop

```
counter = 0
found  = False

while counter < len(list):
      if list[counter] == itemLookingFor:
            found = True

      counter +=1
```

4. Use selection. Use **counter** as the index of the item in the list that is being examined. If it is the same as the item we're looking for then set **found** to True.

5. **OUTSIDE** the selection but **INSIDE** the loop, increment counter by 1.

repl.it

Search A List - How To Code - Method 1

```
counter = 0
found  = False

while counter < len(list):
     if list[counter] == itemLookingFor:
          found = True
     counter +=1

if found == True:
     print(itemLookingFor + " has been found in the list")
else:
     print(itemLookingFor + " is not in the list")
```

4. **AFTER** the loop,use selection to display a message about whether the item was found or not.

Search A List - How To Code - Method 1

```
list = [“pencil”, “ruler”, “pen”, “eraser”, “calculator”]
itemLookingFor = “pen”
counter = 0
found  = False

while counter < len(list):
     if list[counter] == itemLookingFo
          found = True
counter +=1

if found == True:
     print(itemLookingFor + “ has been found in the list”)
else:
     print(itemLookingFor + “ is not in the list”)
```

itemLookingFor	counter	found	list[counter]
pen	0	False	

Search A List - How To Code - Method 1

```
list = [“pencil”, “ruler”, “pen”, “eraser”, “calculator”]
itemLookingFor = “pen”
counter = 0
found  = False

while counter < len(list):
     if list[counter] == itemLookingFo
          found = True
counter +=1

if found == True:
     print(itemLookingFor + “ has been found in the list”)
else:
     print(itemLookingFor + “ is not in the list”)
```

itemLookingFor	counter	found	list[counter]
pen	0	False	
pen	0	False	pencil

Search A List - How To Code - Method 1

```
list = [“pencil”, “ruler”, “pen”, “eraser”, “calculator”]
itemLookingFor = “pen”
counter = 0
found  = False

while counter < len(list):
     if list[counter] == itemLookingFo
          found = True
counter +=1

if found == True:
     print(itemLookingFor + “ has been found in the list”)
else:
     print(itemLookingFor + “ is not in the list”)
```

itemLookingFor	counter	found	list[counter]
pen	0	False	
pen	0	False	pencil
pen	1	False	ruler

Search A List - How To Code - Method 1

```
list = [“pencil”, “ruler”, “pen”, “eraser”, “calculator”]
itemLookingFor = “pen”
counter = 0
found  = False

while counter < len(list):
     if list[counter] == itemLookingFo
          found = True
counter +=1

if found == True:
     print(itemLookingFor + “ has been found in the list”)
else:
     print(itemLookingFor + “ is not in the list”)
```

itemLookingFor	counter	found	list[counter]
pen	0	False	
pen	0	False	pencil
pen	1	False	ruler
pen	2	True	pen

Search A List - How To Code - Method 1

```
list = [“pencil”, “ruler”, “pen”, “eraser”, “calculator”]
itemLookingFor = “pen”
counter = 0
found  = False

while counter < len(list):
     if list[counter] == itemLookingFo
          found = True
counter +=1

if found == True:
     print(itemLookingFor + “ has been found in the list”)
else:
     print(itemLookingFor + “ is not in the list”)
```

itemLookingFor	counter	found	list[counter]
pen	0	False	
pen	0	False	pencil
pen	1	False	ruler
pen	2	True	pen
pen	3	True	eraser

Search A List - How To Code - Method 1

```
list = [“pencil”, “ruler”, “pen”, “eraser”, “calculator”]
itemLookingFor = “pen”
counter = 0
found  = False

while counter < len(list):
     if list[counter] == itemLookingFo
          found = True
counter +=1

if found == True:
     print(itemLookingFor + “ has been found in the list”)
else:
     print(itemLookingFor + “ is not in the list”)
```

itemLookingFor	counter	found	list[counter]
pen	0	False	
pen	0	False	pencil
pen	1	False	ruler
pen	2	True	pen
pen	3	True	eraser
pen	4	True	calculator

Search A List - How To Code - Method 1

```
list = [“pencil”, “ruler”, “pen”, “eraser”, “calculator”]
itemLookingFor = “pen”
counter = 0
found  = False

while counter < len(list):
     if list[counter] == itemLookingFo
          found = True
counter +=1

if found == True:
     print(itemLookingFor + “ has been found in the list”)
else:
     print(itemLookingFor + “ is not in the list”)
```

itemLookingFor	counter	found	list[counter]
pen	0	False	
pen	0	False	pencil
pen	1	False	ruler
pen	2	True	pen
pen	3	True	eraser
pen	4	True	calculator
	5		

Search A List - How To Code - Method 2

```
list = [“pencil”, “ruler”, “pen”, “eraser”, “calculator”]
itemLookingFor = “pen”
counter = 0
found  = False

while counter < len(list):
     if list[counter] == itemLookingFor and found == False:
          found = True
counter +=1
```

itemLookingFor	counter	found	list[counter]
pen	0	False	
pen	0	False	pencil
pen	1	False	ruler
pen	2	True	pen
	3		

Search A List - How To Code - Method 3

```
list = [“pencil”, “ruler”, “pen”, “eraser”, “calculator”]
itemLookingFor = “pen”


if itemLookingFor in list:
     print(itemLookingFor + “ has been found in the list”)
else:
     print(itemLookingFor + “ is not in the list”)
```

Task - Predict & Run

```
1  # Task - Predict & Run
2
3  # Add comments to the code to predict what it will do when run.
4  # Run the code to test your predictions
5
6  ######### Example 1 ######################
7
8  rockStars = ["John","Paul","George","Ringo","Freddie","Brian","John","Roger"]
9
10 counter = 0
11 found = False
12
13 while counter < len(rockStars):
14
15   if rockStars[counter] == "George":
16     found = True
17
18   counter +=1
19
20 if found == True:
21   print("George is in the list")
22 else:
23   print("George is not in the list")
```

repl.it

Task - Predict & Run

```
25  ######## Example 2 ######################
26
27  if "George" in rockStars:
28    print("George is in the list")
29  else:
30    print("George is not in the list")
31
32  ######## Example 3 ######################
33
34  counter = 0
35  found = False
36
37  while counter < len(rockStars) and found == False:
38
39    if rockStars[counter] == "George":
40      found = True
41
42    counter +=1
43
44  if found == True:
45    print("George is in the list")
46  else:
47    print("George is not in the list")
```

repl.

Task - Investigate

```
1  ################ Task - Investigate
2
3  # Answer the questions about the code below
4
5  rockStars = ["John","Paul","George","Ringo","Freddie","Brian","John","Roger"]
6
7  counter = 0
8  found = False
9
10 while counter < len(rockStars):
11
12   if rockStars[counter] == "George":
13     found = True
14
15   counter +=1
16
17 if found == True:
18   print("George is in the list")
19 else:
20   print("George is not in the list")
21
22 # What is the purpose of the counter variable?
23
24 # What is the purpose of the found variable?
25
26 # What does the line counter +=1 do?
27
28 # If line 12 was changed to if rockStars[1] what effect would it have on the output and why?
29
30 # Why is the selection on line 17 not indented?
```

repl.it

Resources created by Andy Colley (@MrAColley)

Task - Modify

```
33  ############## Task - Modify
34
35  # Copy the code from above and adapt it so that
36
37  # It outputs the list to the user.
38  # The user has to input the item to be found.
39  # It ignores case on the input
40  # The program ends the loop when it has found the item
41
42
43  ############## Task - Modify 2
44
45  # Copy the code from above and adapt it so that it uses the python if...in instead of a while loop.
46
```

repl.it

Task - Make

```
1   # Task - Make
2
3   # Write a program that:
4
5   # Initialises a list of the top 10 selling artists of all time (do some
    research and find out who they are)
6
7   # Asks the user to guess an artist who could be in the list.
8
9   # Ignores case for the comparisons below.
10
11  # If the artist input is in the list, output a suitable message.
12
13  # If the input is not in the list, output a suitable message.
14
```