2

Lecture 2

Loops, Lists, and Functions

Last time...

- print()
- Setting variables
- Types
- Numerical, Comparison, and Boolean Operators
- ‘if’ clauses

Things we missed!

- Extra comparison operators (==, !=)
- These are “equal to” and “not equal to” respectively!
- For example, `2 == 3` would return False.

You should now be able to:

- Return output to the user
- Receive input from the user
- Store, use, and compare data

Before we resume

We have a few things we want to go over before we dive into new content!

Marco Polo

```
print("Marco")
response = input()
if response == "Polo":
    print("Yay!")
else:
    print("Incorrect!")
```

Marco Polo

```
print("Marco")
response = input()
if response == "Polo":
    print("Yay!")
else:
    print("Incorrect!")
```

Goal: If the user answers correctly, play again.

Marco Polo

```
print("Marco")
response = input()
if response == "Polo":
    print("Yay!")
else:
    print("Incorrect!")
```

Goal: If the user answers correctly, play again.

Is this a good solution?

What if we want to have it continue up to n times, or indefinitely?

Marco Polo

```
print("Marco")
response = input()
if response == "Polo":
    print("Yay!")
else:
    print("Incorrect!")
```

Rephrased: *While the user answers correctly, keep playing*

While Loop

```
while some_condition:
  print("Do something")
  # If the condition fails, then exits the loop
```

Don't try running this just yet!

Marco Polo

```
while True:
  print("Marco")
  response = input()
  if response == "Polo":
    print("Yay!")
  else:
    print("Incorrect!")
```

This will keep running...forever!!!

Marco Polo

```
keep_playing = True
while keep_playing:
   print("Marco")
   response = input()
   if response == "Polo":
     print("Yay! Game Continuing")
   else:
     print("Incorrect! Exiting!")
     keep_playing = False
```

This will now exit correctly

Challenge:

How do we print out the numbers 1 to n using a while loop?

Challenge:

What about printing out all the multiples of 3 less than n?

For Loop

```
n = int(input("Enter a number:"))
counter = 1

while counter <= n:

  print(counter)

  counter = counter + 1
```

***Currently**: Start with setting a counter to one. Whilst the counter is less than n, print it and increment it.*

For Loop

```
n = int(input("Enter a number:"))
counter = 1

while counter <= n:

  print(counter)

  counter = counter + 1
```

Currently: Start with setting a counter to one. Whilst the counter is less than n, print it and increment it.

Is there a better way to rephrase this?

For Loop

```
n = int(input("Enter a number:"))
counter = 1

while counter <= n:

  print(counter)

  counter = counter + 1
```

Rephrased: For each integer between 1 and n inclusive, print it.

For Loop

```
n = int(input("Enter a number:"))
for i in range(1, n+1):
  print(i)
```

Rephrased: For each integer between 1 and n inclusive, print it.

For Loop

```
n = int(input("Enter a number:"))
for i in range(1, n+1):
  print(i)
```

***Note**: range(x, y) includes x but not y, hence why we use n+1 here*

Challenge:

Print out all the multiples of 3 less than 3n, using a for loop

For loops are incredibly powerful in Python, we'll come back to them later!

Lists

```
x = [1,2,3,4]

print(x)

x.append(5)

print(x)
```

Lists

```
y = []
print(y)
y.append(1)
print(y)
```

Lists

To retrieve a certain element of a list

```
mylist = [1,2,3,4,5]

mynumber = mylist[3]

print(mynumber)
```

What number does this print?

Lists

To retrieve a certain element of a list

```
mylist = [1,2,3,4,5]
mynumber = mylist[-1]
print(mynumber)
```

You can also go from the end of the list!

Lists

Finding the length of a list

```
my_list = [1,2,3,4,5]

list_length = len(my_list)

print(list_length)
```

Try:

- Playing around with your own lists
- Are there any constraints on what can be in lists?
- What do == and the other comparison operators do on lists?
- Arithmetic operations on lists
- What happens when you try to retrieve the 5th element of a list that actually only has 3 elements?

Lists

To remove a certain element of a list

```
mylist = [1,2,3,4,5]

mylist.remove(1)

print(mylist)
```

Lists

To remove a certain element of a list

```
mylist = [1,2,3,4,5]
mylist.remove(1)
print(mylist)
```

What happens if your initial list has multiple of the value you want to remove?

Try:

- What happens when you try to remove an element from list that isn't actually in the list?
- How do you remove the nth element of a list?
- If you have a list of size n, how do you remove every single element of the list (and get back the empty list)?

Lists

```
list_one = [1,2,3,4,5]

list_two = list_one

list_one.append(6)

print(list_one)

print(list_two)
```

Lists

```
list_one = [1,2,3,4,5]

list_two = list_one

list_one.append(6)

print(list_one)

print(list_two)
```

What happens here?

Why?

Lists

```
list_one = [1,2,3,4,5]
```

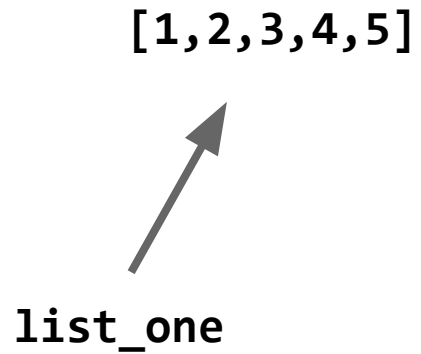
Somewhere in memory.....

[1,2,3,4,5]

Lists

```
list_one = [1,2,3,4,5]
```

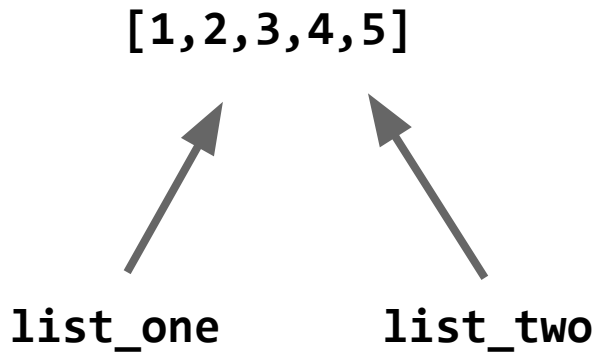
Somewhere in memory.....



Lists

```
list_one = [1,2,3,4,5]
list_two = list_one
```

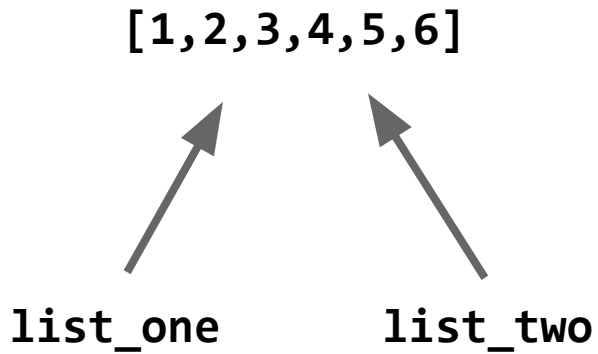
Somewhere in memory.....



Lists

```
list_one = [1,2,3,4,5]
list_two = list_one
list_one.append(6)
```

Somewhere in memory.....



Lists

Checking if two lists are actually the same

```
list_one = [1,2,3,4,5]
list_two = list_one
```

```
id(list_one)
id(list_two)
```

Lists

Checking if two lists are actually the same

```
list_one = [1,2,3,4,5]
list_two = list_one
```

```
id(list_one)
id(list_two)
```

```
list_one is list_two
```

Try:

- The id() function with other types we've covered
- The 'is' operator with other types we've covered
- Play around with what happens to the id() or variables that we change/modify e.g. strings that we concatenate, lists that we append/remove, etc.

Lists

Cloning a List

```
list_one = [1,2,3,4,5]

list_two = list_one[:]

list_one.append(6)

print(list_two)
```

Lists

Cloning a List

```
list_one = [1,2,3,4,5]
list_two = list_one[:]
list_one.append(6)
print(list_two)
```

Taking a 'slice'

Lists

You can also specify which parts of the list you want to clone

```
list_one = [1,2,3,4,5]

list_two = list_one[0:2]

print(list_two)
```

Try:

- Test id() and ‘is’ on lists that you’ve sliced/cloned
- Try play around with the slice indices. Are there any limitations?

So much more you can do on lists!

- pop()
- copy()
- insert()
- reverse()
- sort()
- index()
- See the official documentation!

Lists

Questions?

More Challenges:

- Given a list, how do you print out every single element of the list?
- How do you print out every single element of a list, but in reverse order? What about every other element of a list?
- Let the user input as many numbers as they want, one at a time. When the user inputs “add”, prints the sum of all the numbers to the screen.