



Python Statements



If, elif , else Statements



Complete Python Bootcamp

- Let's begin to learn about **control flow**
- We often only want certain code to execute when a particular condition has been met.
- For example, **if** my dog is hungry (some condition), then I will feed the dog (some action).



Complete Python Bootcamp

- To control this flow of logic we use some keywords:
 - **if**
 - **elif**
 - **else**



Complete Python Bootcamp

- Control Flow syntax makes use of colons and indentation (whitespace).
- This indentation system is crucial to Python and is what sets it apart from other programming languages.



Complete Python Bootcamp

- Syntax of an **if** statement

```
if some_condition:
```

```
    # execute some code
```



Complete Python Bootcamp

- Syntax of an **if/else** statement

```
if some_condition:
```

```
    # execute some code
```

```
else:
```

```
    # do something else
```



Complete Python Bootcamp

- Syntax of an **if/else** statement

if **some_condition:**

 # execute some code

elif **some_other_condition:**

 # do something different

else:

 # do something else



Let's explore these concepts!



For Loops



Complete Python Bootcamp

Many objects in Python are “iterable”, meaning we can iterate over every element in the object.

Such as every element in a list or every character in a string.

We can use for loops to execute a block of code for every iteration.



Complete Python Bootcamp

The term **iterable** means you can “iterate” over the object.

For example you can iterate over every character in a string, iterate over every item in a list, iterate over every key in a dictionary.



Complete Python Bootcamp

- Syntax of a for loop

```
my_iterable = [1,2,3]
for item_name in my_iterable:
    print(item_name)
```

```
>> 1
```

```
>> 2
```

```
>> 3
```



Complete Python Bootcamp

- Syntax of a for loop

```
my_iterable = [1,2,3]
```

```
for item_name in my_iterable:  
    print(item_name)
```

```
>> 1
```

```
>> 2
```

```
>> 3
```



Complete Python Bootcamp

- Syntax of a for loop

```
my_iterable = [1,2,3]
for item_name in my_iterable:
    print(item_name)
```

```
>> 1
```

```
>> 2
```

```
>> 3
```



Complete Python Bootcamp

- Syntax of a for loop

```
my_iterable = [1,2,3]
for item_name in my_iterable:
    print(item_name)
```

```
>> 1
```

```
>> 2
```

```
>> 3
```




Complete Python Bootcamp

- Syntax of a for loop

```
my_iterable = [1,2,3]
for item_name in my_iterable:
    print(item_name)
```

```
>> 1
```

```
>> 2
```

```
>> 3
```



Complete Python Bootcamp

- Syntax of a for loop

```
my_iterable = [1,2,3]
for item_name in my_iterable:
    print(item_name)
```

```
>> 1
```

```
>> 2
```

```
>> 3
```



Let's explore these concepts!



While Loops



Complete Python Bootcamp

While loops will continue to execute a block of code **while** some condition remains True.

For example, **while** my pool is not full, keep filling my pool with water.

Or **while** my dogs are still hungry, keep feeding my dogs.



Complete Python Bootcamp

- Syntax of a while loop

```
while some_boolean_condition:  
    #do something
```



Complete Python Bootcamp

- You can combine with an else if you want

```
while some_boolean_condition:
```

```
    #do something
```

```
else:
```

```
    #do something different
```



**Let's explore these
concepts!**



Useful Operators



List Comprehensions



Complete Python Bootcamp

List Comprehensions are a unique way of quickly creating a list with Python.

If you find yourself using a for loop along with `.append()` to create a list, List Comprehensions are a good alternative!

To do this, let's go to a Jupyter Notebook!



Python Statements Test Solutions