Python Crash Course

Booleans

True or False in Python. ... Boolean values are the two constant objects False and True. They are used to represent truth values

In [1]:
```
Bachelor  = True
```

In [2]:
```
Bachelor
```

Out[2]: True

You can Compare Two Values to see if the statement or values on both sides are equal or not

In [3]:
```
5 == 3
```

Out[3]: False

In [4]:
```
1 + 3 == 2 * 2 == 2 + 2
```

Out[4]: True

In [5]:
```
2.0 == 2
```

Out[5]: True

Remeber to use a double = sign for comparasion , a single = means assignment

In [6]:
```
name = 'Sam'
```

In [7]:
```
name == 'Sam'
```

Out[7]: True

Remeber Captial Alphabets are not equal to normal ones

In [8]:
```
"Sam" == "sAm"
```

Out[8]: False

Tuples

A tuple is a sequence of immutable (Non-changable or fixed) Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets. Creating a tuple is as simple as putting different comma-separated values.

In [9]:
```
t = (1,2,3)
```

In [10]:
```
t[0]
```

Out[10]:
```
1
```

You cannot assign values to a tuple nor you can append it

In [11]:
```
t[0] = 'NEW'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-11-93bfe9be1549> in <module>
----> 1 t[0] = 'NEW'

TypeError: 'tuple' object does not support item assignment
```

Otherwise its pretty much same as a list

In [12]:
```
t1 = ('ali','ahmed','ammar')
t1
```

Out[12]:
```
('ali', 'ahmed', 'ammar')
```

In [13]:
```
t2 = ("Ali",'Ammar',1)
```

Sets

A Set is an unordered collection data type that is iterable, mutable, and has no duplicate elements

You can include elements of Different Datatypes

In [14]:
```
s1= {1,2,4,3,"h"}          #collection of different data type, always print in o
s1
```

Out[14]:
```
{1, 2, 3, 4, 'h'}
```

No Duplicates, And Numbers are in Order

In [15]:
```
{1,2,3,1,2,1,2,3,3,3,3,2,2,2,1,1,2}  # unique elemnents are included only
```

Out[15]:
```
{1, 2, 3}
```

You add new Elements like this

In [16]:
```
s1.add(6)
s1
```

Out[16]:
```
{1, 2, 3, 4, 6, 'h'}
```

Comparison Operators

A comparison operator in python, also called python relational operator, compares the values of two operands and returns True or False based on whether the condition is met

In [17]:
```
1 > 2      #greater than
```

Out[17]:
```
False
```

In [18]:
```
1 < 2     # less than
```

Out[18]:
```
True
```

In [19]:
```
1 == 1    #equal to
```

Out[19]:
```
True
```

In [20]:
```
1 >= 1    #greater than or equal to
```

Out[20]:
```
True
```

In [21]:
```
1 <= 4    # less than or equal to
```

Out[21]:
```
True
```

In [22]:
```
'Win' == 'win'
```

Out[22]:
```
False
```

In [27]:
```
'Win' != "win"     #not equals to
```

Out[27]:
```
True
```

Logic Operators

The logical operators in Python are used to combine the true or false values of variables (or expressions) so you can figure out their resultant truth value. Three logical operators are available in Python:

The And Operator

```
In [28]: (1 < 2)
```

```
Out[28]: True
```

```
In [29]: (1 < 2) and (2 > 3)        #AND
```

```
Out[29]: False
```

```
In [30]: 5 == 5 and 0 < 3
```

```
Out[30]: True
```

The Or Operator

```
In [31]: (1 > 2) or (2 < 3)          # OR
```

```
Out[31]: True
```

```
In [32]: 20 + 3 >= 23 or 5 != 5
```

```
Out[32]: True
```

```
In [33]: (1 == 2) or (2 == 3) or (4 == 4)
```

```
Out[33]: True
```

The Not Operator

```
In [34]: not True           #NOT
```

```
Out[34]: False
```

```
In [35]: not 5 > 3
```

```
Out[35]: False
```

```
In [36]: not (5 < 3 and 5 < 33)
```

```
Out[36]: True
```

```
In [37]: not (5 < 6 or 5 < 3 and 5 < 33)
```

```
Out[37]: False
```

Conditional Statements

if,elif, else Statements

Decision making is required when we want to execute a code only if a certain condition is satisfied.
The if…elif…else statement is used in Python for decision making.

To use an if statement you write if then pass in a condition followed by a colon, the condition part comes in the next line after indentation (space).

In [40]:
```
age = 12

if age < 14:
    print("sorry you're too young for this ride")
```

```
sorry you're too young for this ride
hello
```

Notice only the indented part is actually inside the if condition

In [41]:
```
age = 14

if age < 14:
    print("sorry you're too young for this ride")

print('Always print')
```

```
Always print
```

Similarly you can also add an else condition like below

In [43]:
```
Memeber_of_club = False

if Memeber_of_club == True:
    print('Welcome')
else:
    print('Sorry You are not allowed here')
```

```
Welcome
```

Note: Bollean Conditions like above can also be checked like this: `if condition:`
No Need to do `== True`

In [44]:
```
Memeber_of_club = True
if Memeber_of_club:
    print('Welcome')
```

```
Welcome
```

In Other Programming Languages there is an `else if` check for multiple conditions, just like that python also has an else if but its called `elif` in python 3.

In [45]:
```
status = 'bad'

if status == 'good':
    print('He did pretty well')
elif status == 'bad':
    print('He really messed up')
else:
    print('I dont know how he did')
```

```
He really messed up
```

Now look at this automatic datatype finder, which spits out the datatype of the entered value

In [48]:
```
var1 = input("Enter anything")
var1 = eval(var1)

if (type(var1) == int):
    print("Type of the variable is Integer")
elif (type(var1) == float):
    print("Type of the variable is Float")
elif (type(var1) == complex):
    print("Type of the variable is Complex")
elif (type(var1) == bool):
    print("Type of the variable is Bool")
elif (type(var1) == str):
    print("Type of the variable is String")
elif (type(var1) == tuple):
    print("Type of the variable is Tuple")
elif (type(var1) == dict):
    print("Type of the variable is Dictionaries")
elif (type(var1) == list):
    print("Type of the variable is List")
else:
    print("Type of the variable is Unknown")
```

```
Enter anything[1]
Type of the variable is List
```

Assignment 3: Make an Auto Grader

So your task is to make an auto grader which will be a program that asks user their percentage and based on the percentage prints their Grade i.e. A+, A, B, C, and F. The threshold of the grades can be decided by you for e.g you can choose to give A+ over 90 or 90+
The code will be pretty similar to the above code

In [ ]:
```
#ADD CODE HERE
                    # ~ 12 lines of code
```