DAY-2

Material

Python Numbers:

There are three numeric types in Python:

- int
- float
- complex
- **Int**
  - Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.
- **Float**
  - Float, or "floating point number" is a number, positive or negative, containing one or more decimals.
  - Float can also be scientific numbers with an "e" to indicate the power of 10.
- **Complex**
  - Complex numbers are written with a "j" as the imaginary part
- **Type Conversion**
  - You can convert from one type to another with the int(), float(), and complex() methods.
  - **Note:** You cannot convert complex numbers into another number type.
- **Random Number**
  - Python does not have a random() function to make a random number, but Python has a built-in module called random that can be used to make random numbers.

Python Casting

- **Specify a Variable Type**

  There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

  Casting in python is therefore done using constructor functions:

  - int() - constructs an integer number from an integer literal, a float literal (by rounding down to the previous whole number), or a string literal (providing the string represents a whole number)
  - float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
  - str() - constructs a string from a wide variety of data types, including strings, integer literals and float literals

Python Strings

- **String Literals**
  - String literals in python are surrounded by either single quotation marks, or double quotation marks.
  - 'hello' is the same as "hello".
  - You can display a string literal with the `print()` function
- **Assign String to a Variable**
  - Assigning a string to a variable is done with the variable name followed by an equal sign and the string.
- **Multiline Strings**
  - You can assign a multiline string to a variable by using three quotes Or three single quotes.
- **Strings are Arrays**
  - Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.
  - However, Python does not have a character data type, a single character is simply a string with a length of 1.
  - Square brackets can be used to access elements of the string.
- **Slicing**
  - You can return a range of characters by using the slice syntax.
  - Specify the start index and the end index, separated by a colon, to return a part of the string.
- **Negative Indexing**
  - Use negative indexes to start the slice from the end of the string
- **String Length**
  - To get the length of a string, use the `len()` function.
- **String Methods**
  - Python has a set of built-in methods that you can use on strings.
- **Check String**
  - To check if a certain phrase or character is present in a string, we can use the keywords `in` or `not in`.
- **String Concatenation**
  - To concatenate, or combine, two strings you can use the + operator.
- **String Format**
  - As we learned in the Python Variables chapter, we cannot combine strings and numbers.
  - But we can combine strings and numbers by using the `format()` method!
  - The `format()` method takes the passed arguments, formats them, and places them in the string where the placeholders `{}`
  - The format() method takes unlimited number of arguments, and are placed into the respective placeholders
  - You can use index numbers {0} to be sure the arguments are placed in the correct placeholders.

- **Escape Character**
  - To insert characters that are illegal in a string, use an escape character.

- An escape character is a backslash `\` followed by the character you want to insert.
- You will get an error if you use double quotes inside a string that is surrounded by double quotes.
- To fix this problem, use the escape character `\"`.
- The escape character allows you to use double quotes when you normally would not be allowed
- Other escape characters used in Python:

Code	Result
\'	Single Quote
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Octal value
\xhh	Hex value

Python Booleans

- Booleans represent one of two values: True or False.
- **Boolean Values**
  - In programming you often need to know if an expression is `True` or `False`.
  - You can evaluate any expression in Python, and get one of two answers, `True` or `False`.
  - When you compare two values, the expression is evaluated and Python returns the Boolean answer
  - When you run a condition in an if statement, Python returns `True` or `False`
- **Evaluate Values and Variables**
  - The `bool()` function allows you to evaluate any value, and give you `True` or `False` in return
- **Most Values are True**
  - Almost any value is evaluated to `True` if it has some sort of content.
  - Any string is `True`, except empty strings.
  - Any number is `True`, except `0`.
  - Any list, tuple, set, and dictionary are `True`, except empty ones.
- **Some Values are False**
  - In fact, there are not many values that evaluates to False, except empty values, such as (), [], {}, "", the number 0, and the value None. And of course the value False evaluates to False.
  - One more value, or object in this case, evaluates to `False`, and that is if you have an objects that are made from a class with a `__len__` function that returns `0` or `False`
- **Functions can Return a Boolean**

- Python also has many built-in functions that returns a boolean value, like the isinstance() function, which can be used to determine if an object is of a certain data type

Python Operators

- Operators are used to perform operations on variables and values.
- Python divides the operators in the following groups:
    - Arithmetic operators
    - Assignment operators
    - Comparison operators
    - Logical operators
    - Identity operators
    - Membership operators
    - Bitwise operators
- **Python Arithmetic Operators**
    - Arithmetic operators are used with numeric values to perform common mathematical operations

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

- **Python Assignment Operators**
    - Assignment operators are used to assign values to variables.

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x – 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3

<<=	x <<= 3	x = x << 3

- **Python Comparison Operators**
  - Comparison operators are used to compare two values

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

- **Python Logical Operators**

  - Logical operators are used to combine conditional statements

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
Or	Returns True if one of the statements is true	x < 5 or x < 4
Not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

- **Python Identity Operators**
  - Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location.

Operator	Description	Example
is	Returns true if both variables are the same object	x is y
is not	Returns true if both variables are not the same object	x is not y

- **Python Membership Operators**
  - Membership operators are used to test if a sequence is presented in an object

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

- **Python Bitwise Operators**
  - Bitwise operators are used to compare (binary) numbers

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

Python Lists

- A list is a collection which is ordered and changeable. In Python lists are written with square brackets.
- **Access Items:**
  You access the list items by referring to the index number.
- **Negative Indexing:**
  Negative indexing means beginning from the end, -1 refers to the last item, -2 refers to the second last item etc.
- **Range of Indexes:**
  - You can specify a range of indexes by specifying where to start and where to end the range.
  - When specifying a range, the return value will be a new list with the specified items.
- Remember that the first item has index 0.
- By leaving out the start value, the range will start at the first item
- By leaving out the end value, the range will go on to the end of the list.
- **Range of Negative Indexes:**
  - Specify negative indexes if you want to start the search from the end of the list
- **Change Item Value:**
  - To change the value of a specific item, refer to the index number
- **Loop Through a List:**
  - You can loop through the list items by using a for loop
- **Check if Item Exists**
  - To determine if a specified item is present in a list use the in keyword
- **List Length**
  - To determine how many items a list has, use the len() function

- **Add Items**
  - To add an item to the end of the list, use the append() method
- **Remove Item**
  - There are several methods to remove items from a list.
  - The remove() method removes the specified item
  - The pop() method removes the specified index, (or the last item if index is not specified)
  - The del keyword removes the specified index
  - The del keyword can also delete the list completely
- **Copy a List**
  - You cannot copy a list simply by typing list2 = list1, because: list2 will only be a reference to list1, and changes made in list1 will automatically also be made in list2.
  - There are ways to make a copy, one way is to use the built-in List method copy().

- **Join Two Lists**

  - There are several ways to join, or concatenate, two or more lists in Python.
  - One of the easiest ways are by using the + operator.
  - Another way to join two lists are by appending all the items from list2 into list1, one by one
  - you can use the extend() method, which purpose is to add elements from one list to another list

- **The list() Constructor**
  - It is also possible to use the list() constructor to make a new list.