**Python Loops**

Loops are used when you need a section of code to repeat itself. Python loops include the **for** loop and the **while** loop.

**The for Loop**

The **for** loop is used to iterate over the items of a sequence, this includes strings and lists. Let’s look at a basic **for** loop that executes a statement 5 times:

|  |  |
| --- | --- |
| **for x in range(5):**  **print ('Number', x)** | ***Result:*** **Number 0****Number 1****Number 2****Number 3****Number 4** |

The **range** statement (in the **for** loop example above) defines the parameters used within the loop. In the above example the **range** function contains one parameter **(5)**, indicating five executions.

The **range** function can also contain two or three parameters, for example:

|  |  |  |
| --- | --- | --- |
| **for x in range(2, 5):** **print ('Number', x)** |  | **for x in range(0,5,2):** **print ('Number', x)** |
| ***Result:*** **Number 2****Number 3****Number 4** |  | ***Result:*** **Number 0****Number 2****Number 4** |
| **Explanation:**Generates a sequence of numbers from **2** to **5** (excluding **5**) incrementing by **1**. |  | **Explanation:**Generates a sequence of numbers from **0** to **5** (excluding **5**) incrementing by **2**. |

*Run the flowing for loop and explain the output:*

**for x in range(5,2,-1):**

 **print ('Number', x)**

*Explanation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

 *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

 *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

 *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

More on **for** Loops

|  |  |
| --- | --- |
| *A* ***for*** *loop which iterate through a string:* | *A* ***for*** *loop which iterate through a list:* |
| **string1 = 'ABC'****for x in string1:** **print (x)** | **list1 = ['Cat', 'Dog', 'Rat']****for x in list1:** **print (x)** |

*Explain how the following nested* ***for*** *loop operates by examining its output result:*

|  |  |
| --- | --- |
| **for x in range(1, 4):** **for y in range(1, 4):** **print (x, '\*', y, '=' , x\*y)** | ***Result:* 1 \* 1 = 1** **1 \* 2 = 2** **1 \* 3 = 3** **2 \* 1 = 2** **2 \* 2 = 4** **2 \* 3 = 6** **3 \* 1 = 3** **3 \* 2 = 6** **3 \* 3 = 9** |

*Explanation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

 *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

 *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

 *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*This is fun, let’s create an empty list and populate it:*

|  |  |
| --- | --- |
| **# Create an empty list****list2 = []****# Populate the list****for i in range(3):** **print ("Adding", i, "to the list.")** **list2.append(i)****print(list2)** | ***Result:* Adding to the list: 0** **Adding to the list: 1** **Adding to the list: 2*****Result:* [0, 1, 2]** |

*In the above example we used the append function to add each item to the list. Append adds (appends) to the end of the list.*

**The while Loop**

Unlike the **for** loop which is used to repeat a section of code a given number of times, the **while** loop is used to repeat a section of code until a defined condition is met.

*A basic* ***while*** *loop:*

**num = 0**

**while (num < 3):**

 **num = num + 1**

 **print ('loop',num)**

**print ('Out of the loop!')**

*Exercise: Does position count? Determine the output of the following statements:*

|  |  |
| --- | --- |
| **num = 0****while (num < 3):** **num = num + 1** **print('loop',num)****print('Out of the loop!')** | **num = 0****while (num < 3):** **print('loop',num)** **num = num + 1****print('Out of the loop!')** |

*In the above statements, the* ***num = num + 1*** *and the* ***print('loop',num)****statements**have been reversed. Will the output be identical? Try it.*

*Explanation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

 *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

 *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

 *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Exercise: What is an infinite loop? Examine the following statements and attempt to determine the output.*

|  |  |
| --- | --- |
| **num1 = 0****while (num1 < 3):** **num1 = num1 + 1** **name=input('Enter your name: ')** **print (name,'this is loop',num1)****print ('Out of the loop!')** | **num1 = 0****num2 = 0****while (num1 == 0):** **num2=num2 + 1** **name=input('Enter your name: ')** **print (name,'this is loop',num2)****print ('Out of the loop!')****# A loop becomes an infinite loop if a condition remains true.** |
|  |  |

**The Basic if Statement (revisited)**

The **if** statement used check if a condition is true.

*Exercise: Examine the following basic if statement:*

**age = int(input('Please enter your age: '))**

**if age >= 65:**

 **print('You are entitled to a 10% discount.')**

**print('Please pay now.')**

*In the above statement if the (****if age >= 65****) condition is true (if the age entered is 65 or above), the customer is entitled to a 10% discount.*

**The break Command**

The **break** command is used to break out of a loop.

*Exercise: Compare the following loop statements:*

|  |  |
| --- | --- |
| **for x in range(10):** **print(x)** | **for x in range(10):** **print(x)** **if x == 3:** **break** |

*The* ***break*** *command will break out of the loop if the (****if x == 3****) given condition is true.*

*Exercise: What is the difference between “codeA” and “codeA”?*

**# codeA.py**

**num=int(input('Enter a number over 10: '))**

**while (num <=10):**

 **num = int(input('You must enter a number over 10: '))**

**print('You Entered Number: ',num)**

**# codeB.py**

**while True:**

 **num = int(input('Enter a number over 10: '))**

 **if num >10:**

 **break**

**print('You Entered Number: ',num)**

*Explanation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

 *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

 *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

 *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*In the latter example the while condition is set to* ***True****.* ***True*** *is a Boolean value, Boolean values are either* ***True****or****False*** *and are used with* ***if*** *and* ***while*** *commands.*

**Python Operators**

In the example “**1 + 2**”, the **+** is the operator, **1** and **2** are the operands. The **+** is an arithmetic operator. Python supports a number of operators including arithmetic, comparison, logic and membership operators to name a few.

**Arithmetic Operators**

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Description | Example**a=3 b=10** | Answer |
| **/** | Division | **b / a** | **3.33** |
| **\*** | Multiplication | **a \* b** | **30** |
| **+** | Addition | **a + b** | **13** |
| **-** | Subtraction | **b - a** | **7** |
| **%** | Modulus (divides left operand with right and returns remainder) | **b % a** | **1** |
| **\*\*** | Exponent (left operand to the power of right operand) | **b \*\* a** | **1000** |
| **//** | Floor Division (returns the quotient of the division, no decimals) | **B // a** | **3** |

**Comparison Operators**

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Description | Example**a=3 b=10** | Answer |
| **==** | Equal (checks if both operands are equal) | **a == b** | **False** |
| **!=** | Not equal (checks if both operands are not equal) | **a != b** | **True** |
| **>** | Greater than (left operand greater than right) | **a > b** | **False** |
| **>=** | Greater than or equal  | **a >= b** | **False** |
| **<** | Less than (left operand less than right) | **a < b** | **True** |
| **<=** | Less than or equal  | **a <= b** | **True** |

**Logic Operators**

***and****,* ***or*** *and* ***not*** *Operators*

|  |  |  |
| --- | --- | --- |
| **and** operator | **or** operator | **not** operator |
| **a=3****b=7****if (a==3 and b==7):** **print('True')** | **a=3****b=7****if (a==3 or b==7):** **print('True')** | **a=3****b=8****if not(a==3 and b==7):** **print('True')** |
| *Both conditions true?* | *Either condition true?* | *This one is tricky, it reverses the logic state of the operands, if the condition is true it returns false (won’t print the text ‘True’).* |

*Will the output of the following statements print the text ‘True’? Try it.*

|  |  |  |
| --- | --- | --- |
| **a=3****b=7****if (a==2 and b==7):** **print('True')** | **a=3****b=7****if (a==2 or b==7):** **print('True')** | **a=3****b=7****if not(a==2 and b==7):** **print('True')** |

**Membership Operators**

***in*** *and* ***not******in*** *Operators*

|  |  |
| --- | --- |
| **list1 = [5, 10, 15, 20]****num1 = 5****if (num1 in list1):** **print ('Found')** | **list1 = [5, 10, 15, 20]****num1 = 1****if (num1 not in list1):** **print ('Not Found')** |
| **word = 'Hello World!'****letter ='W'****if (letter in word):** **print ('Found')** | **word = 'Hello World!'****letter ='w'****if (letter not in word):** **print ('Not Found')** |
| ***in*** *returns true if a given collection contains a given variable.* | ***not in*** *returns true is a given collection does not contain a given variable.**Why was the letter ‘****w****’ was not found?*  |

Continuing from the above question: “*Why was the letter ‘****w****’ was not found?* Inspect the following **if** conditions in an attempt to answer this question:

**if (letter.lower()in word or letter.upper()in word):**

**if (letter.lower()in word and letter.upper()in word):**

*Explanation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

 *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Exercise: Inspect and comment on the following code:*

**value = int(input('Enter a number between 0 and 10: '))**

**if value not in range(0,11):**

 **print ('True')**

*Explanation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

 *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

 *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

 *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**Boolean Expressions**

Boolean expressions are used to evaluate and produce Boolean values (**True** or **False**). In the following examples the **==** operator is used to test if two values are equal.

|  |  |
| --- | --- |
| **print(5==** **5)****print(type(5== 5))** | ***Result:* True*****Result:* <class 'bool'>** |
| **print(5==** **6)** | ***Result:* False** |
| **a='Hel'****b='lo'****print (a+b =='Hello')** | ***Result:* True** |