



Loops (for, while)

1. For Loops:

The structure of a for loop is as follows:

```
for variable name in range ( number of times to repeat ) :  
    statements to be repeated
```

The syntax is important here. The word **for** must be in lowercase, the first line must end with a colon, and the statements to be repeated **must** be indented. Indentation is used to tell Python which statements will be repeated.

Example 1: the following program will print the word **Hello** ten times:

```
for i in range(10):  
    print('Hello')
```

- **The range function:** the value we put in the range function determines how many times we will loop. The way range works is it produces a list of numbers from zero the value minus one. For instance, range (5) produces five values: 0,1,2,3, and 4.

If we want the list of values to start at a value other than 0, we can do that by specifying the starting value. The statement range (1, 5) will produce the list 1, 2, 3, 4. This brings up one quirk of the range function, it stops one short of where we think it should. If



College of Engineering & Technology
Computer Techniques Engineering Department
Artificial Intelligence – Stage 3
(Python)



we wanted the list to contain the number 1 through 5 (including 5), then we would have to do `range(1, 6)`.

Another thing we can do is to get the list of values to go up by more than one at a time. To do this, we can specify an optional step as the third argument. The statement `range(1, 10, 2)` will step through the list by twos, producing 1, 3, 5, 7, 9.

To get the list of values to go backwards, we can use a step of -1. For instance, `range(5, 1, -1)` will produce the values 5, 4, 3, 2, in that order. Note that the range function stops one short of the ending values. Here are a few more examples:

Statement	Values generated
<code>range(10)</code>	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
<code>range(1, 10)</code>	1, 2, 3, 4, 5, 6, 7, 8, 9
<code>range(3, 7)</code>	3, 4, 5, 6
<code>range(2, 15, 3)</code>	2, 5, 8, 11, 14
<code>range(9, 2, -1)</code>	9, 8, 7, 6, 5, 4, 3

- **The loop variable:** there is one part of a for loop that is a little tricky, and that is the loop variable. In the example below, the loop variable is variable `i`. the output of the following example will be the numbers 0,1,99. Each printed on its own line.

```
for i in range(100):  
    print(i)
```



College of Engineering & Technology
Computer Techniques Engineering Department
Artificial Intelligence – Stage 3
(Python)



When the loop first starts, Python set the variable *i* to 0. Each time we look back up, python increase the value of *i* by 1.

Exercises 1: what is the output of the following program:

```
for i in range(5,0,-1):  
    print(i, end=' ')  
print('Blast off!!!')
```

```
5 4 3 2 1 Blast off!!!
```

Exercises 2: what is the output of the following program:

```
for i in range(4):  
    print('*'*6)
```

```
* * * * *  
* * * * *  
* * * * *  
* * * * *
```

2. While Loops: we have already learned about for loops, which allow us to repeat things about specified number of times. Sometimes, though, we need to repeat something, but we don't know ahead of time exactly how many times it has to be repeated. In this situation that would call for a while loop.



College of Engineering & Technology
Computer Techniques Engineering Department
Artificial Intelligence – Stage 3
(Python)



In the following example, a while loop will allow user to repeatedly enter temperatures. A simple way for the user to indicate that they are done is to have them enter a nonsense temperature like -1000.

```
temp = 0
while temp != -1000:
    temp = eval(input('Enter a temperature (-1000 to quit): '))
    print('In Fahrenheit that is', 9/5*temp+32)
```

Look at the while statement first. It says that we will keep looping, that is, keep getting input and converting temperatures, as long as the temperature entered is not -1000. As soon as -1000 is entered, the while loop stops.

While – else: with the else statement we can run a block of code once when the condition no longer is true:

```
i = 1
while i < 6:
    print(i)
    i += 1
else:
    print("i is no longer less than 6")
```

While – break: with the break statement we can stop the loop even if the while condition is true:

```
i = 1
while i < 6:
    print(i)
    if i == 3:
        break
    i += 1
```



While – Continue: with the continue statement we can stop the current iteration, and continue with the next.

```
i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)
```

Exercises

1. Write a program that outputs 100 lines, numbered 1 to 100, each with your name on it. The output should look like the output below.

```
1 Your name
2 Your name
3 Your name
4 Your name
...
100 Your name
```

2. Write a program that prints out a list of the integers from 1 to 20 and their squares. The output should look like this.

```
1 --- 1
2 --- 4
3 --- 9
...
20 --- 400
```

3. Write a program that uses a for loop to print the numbers 8,11,14,17,20,.....,83,86,89.
4. Write a program that uses a for loop to print the numbers 100,98,96,.....,4,2.
5. Write a program that uses a while loop to read integer positive number then print its digits.



College of Engineering & Technology
Computer Techniques Engineering Department
Artificial Intelligence – Stage 3
(Python)



6. What is the output of the following program:

```
c = 'y'
while c=='y':
    name = input("Enter Student Name: ")
    age = eval(input("Enter Student Age: "))
    c = input("Do you want to continue (y/n): ")
```

7. Use a while loop to print a triangle like the one below. Allow the user to specify how high the triangle should be.

```
*
**
***
****
```

8. Use a while loop to print an upside-down triangle like the one below. Allow the user to specify how high the triangle should be.

```
****
***
**
*
```

9. Use for loops to print a diamond like the one below. Allow the user to specify how high the diamond should be.

```
  *
 ***
*****
*****
 *****
  ***
   *
```