Artificial Intelligence

This lesson will teach you about the for loop with Dictionary, range functions, Nested for loop, functions in the Python Programming Language. By the end of this lab, you'll know the basic concepts about for loops, function, variables, and how to use these functions.

For Loop with Dictionary:

```
numNames = { 1:'One', 2: 'Two', 3: 'Three'}

for pair in numNames.items():
    print(pair)
```

Output
(1, 'One')
(2, 'Two')
(3, 'Three')

For Loop with Dictionary:

```
numNames = { 1:'One', 2: 'Two', 3: 'Three'}

for k,v in numNames.items():
    print("key = ", k , ", value =", v)
```

Output
key = 1, value = One
key = 2, value = Two
key = 3, value = Three

For Loop with the range() Function:

The range class is an immutable sequence type. The range() returns the range object that can be used with the for loop.
Example:

```
for i in range(5):
    print(i)
```

Output
0
1
2
3
4

Exit the For Loop:

The execution of the for loop can be stop and exit using the break keyword on some condition, as shown below.

Example:

```
for i in range(1, 5):
    if i > 2
        break
    print(i)
```

Output
1
2

Continue Next Iteration:

Use the continue keyword to skip the current execution and continue on the next iteration using the continue keyword on some condition, as shown below.

Example:

```
for i in range(1, 5):
    if i > 3
        continue
    print(i)
```

Output
1
2
3

For Loop with else Block:

The else block can follow the for loop, which will be executed when the for loop ends.

Example:

```
for i in range(2):
    print(i)
else:
    print('End of for loop')
```

Output

```
0
1
End of for loop
```

Nested for Loop:

If a loop (for loop or while loop) contains another loop in its body block, we say that the two loops are nested. If the outer loop is designed to perform m iterations and the inner loop is designed to perform n repetitions, the body block of the inner loop will get executed m X n times.

Example: Nested for loop

```
for x in range(1,4):
    for y in range(1,3):
        print('x = ', x, ', y = ', y)
```

Output

```
x =  1, y =  1
x =  1, y =  2
x =  2, y =  1
x =  2, y =  2
x =  3, y =  1
x =  3, y =  2
```

Functions:

A function is a reusable block of code which performs operations specified in the function. They let you break down tasks and allow you to reuse your code in different programs.

There are two types of functions:

- **Pre-defined functions**
- **User defined functions**

What is a Function?

You can define functions to provide the required functionality. Here are simple rules to define a function in Python:

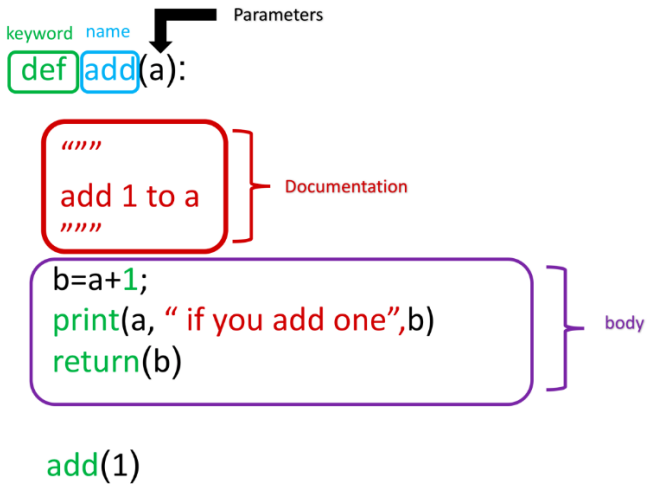
- Functions blocks begin def followed by the function name and parentheses ().
- There are input parameters or arguments that should be placed within these parentheses.
- You can also define parameters inside these parentheses.
- There is a body within every function that starts with a colon (:) and is indented.
- You can also place documentation before the body
- The statement return exits a function, optionally passing back a value

An example of a function that adds on to the parameter a prints and returns the output as b:

```
# First function example: Add 1 to a and store as b

def add(a):
    b = a + 1
    print(a, "if you add one", b)
    return(b)
```

The figure below illustrates the terminology:



We can obtain help about a function:

```
# Get a help on add function
help(add)
```

We can call the function:

```
#·Call·the·function·add()
add(1)
```

We can create different functions. For example, we can create a function that multiplies two numbers. The numbers will be represented by the variables a and b:

```
# Define a function for multiple two numbers

def Mult(a, b):
    c = a * b
    return(c)
```

The same function can be used for different data types. For example, we can multiply two integers:

```
# Use mult() multiply two integers
Mult(2, 3)
```

Variables:

The input to a function is called a formal parameter.

A variable that is declared inside a function is called a local variable. The parameter only exists within the function (i.e. the point where the function starts and stops).

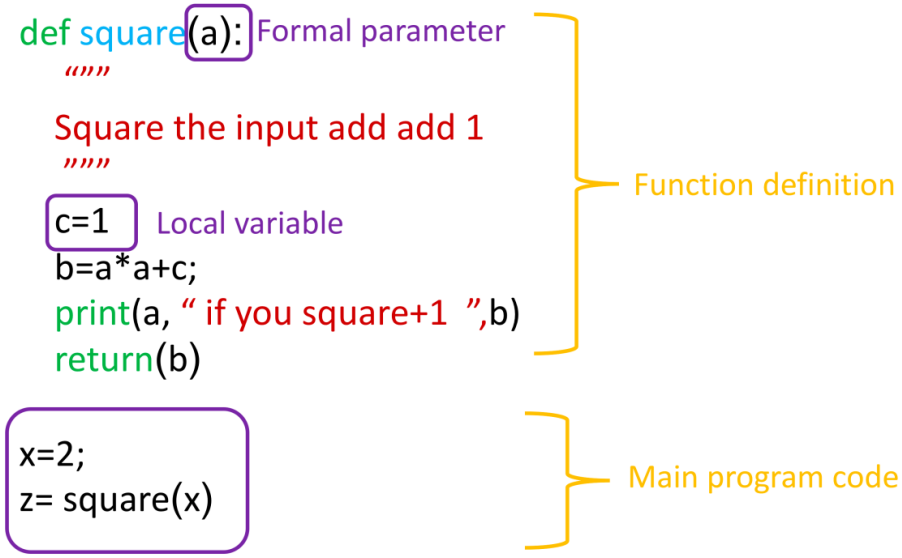
A variable that is declared outside a function definition is a global variable, and its value is accessible and modifiable throughout the program. We will discuss more about global variables at the end of the lab.

```
# Function Definition

def square(a):

    # Local variable b
    b = 1
    c = a * a + b
    print(a, "if you square + 1", c)
    return(c)
```

The labels are displayed in the figure



We can call the function with an input of **3**:

Pre-Defined Functions:

There are many pre-defined functions in Python, so let's start with the simple ones.

The print() function:

Build-in function print()

x = [10.0, 8.5, 9.5, 7.0, 7.0, 9.5, 9.0, 9.5]
print(x)
output
[10.0, 8.5, 9.5, 7.0, 7.0, 9.5, 9.0, 9.5]
The sum() function adds all the elements in a list or tuple:
Use sum() to add every element in a list or tuple together

sum(x)

output
70.0

Reference

https://colab.research.google.com/

Questions