Unit 3 Functions, Scoping and Abstractions

Functions:

In Python, a function is a group of related statements that performs a specific task.

Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable.

Furthermore, it avoids repetition and makes the code reusable.

A function takes an i nput, performs a computation, and produces an output. Python has user-defined functions, anonymous functions and built-in functions. While creating a functions we can use 2 keywords

1. def (Mandatory)
2. return (Optional)

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

Syntax of Function

```
def function_name(parameters):

    """docstring"""

    statement(s)
```

Above shown is a function definition that consists of the following components.

Keyword def that marks the start of the function header.
A function name to uniquely identify the function. Function naming follows the same rules of writing identifiers in Python.
Parameters (arguments) through which we pass values to a function. They are optional.

A colon (:) to mark the end of the function header.

Optional documentation string (docstring) to describe what the function does.

One or more valid python statements that make up the function body.

Statements must have the same indentation level (usually 4 spaces).

An optional return statement to return a value from the function.

Prepared by Abhi Shah

Calling a Function

Once we have defined a function, we can call it from another function, program, or even the Python prompt. To call a function we simply type the function name with appropriate parameters.

Function call syntax: Function_name(parameters or arguments)

Without calling it will not give any output.

In [2]:
```
#Example of writing a function and calling a function
def greet(name):
    """
    This function greets to
    the person passed in as
    a parameter
    """
    print("Hello, " + name + ". Good morning!")
greet("Python")
```

```
Hello, Python. Good morning!
```

In [3]:
```
#Another example of function
def my_function():
    print("Hello from a function")

my_function()
```

```
Hello from a function
```

Return Values

To let a function return a value, use the return statement:

In [4]:
```
#Example of return values
def my_function(x):
    return 5 * x

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

```
15
25
45
```

#Defining Categories of User Defined Functions:

1. Function with no parameters and no return type
2. Function with parameters and no return type
3. Function with parameters and with return type
4. Function with no parameters and with return type

Prepared by Abhi Shah

In [6]:
```
#Example of Function with no parameters and no return type.
def square():
    s=int(input("Enter number:"))
    a=s**2
    print(a)
square()
```

```
Enter number:2
4
```

In [9]:
```
#Example of Function with parameters and no return type.
def square(s):
    a=s**2
    print(a)
s=int(input("Enter number:"))
square(s)
```

```
Enter number:2
4
```

In [14]:
```
#Example of Function with parameters and with return type.
def square(s):
    a=s**2
    return a
s=int(input("Enter number:"))
print(square(s))
```

```
Enter number:2
4
```

In [15]:
```
#Example of Function with no parameters and with return type.
def square():
    a=s**2
    return a
s=int(input("Enter number:"))
print(square())
```

```
Enter number:2
4
```

In [17]:
```
#Example to subtract one number from another number and to take that two nu
def subtract(a,b):
    return a-b
a=int(input("Enter a:"))
b=int(input("Enter b:"))
print(subtract(a,b))
```

```
Enter a:4
Enter b:2
2
```

In [18]:
```
#Example to make a function which accept a number and tells whether it is +
def test(a):
    if (a==0):
        print("The number is Zero")
    elif(a>0):
        print("The number is +ve")
    else:
        print("The number is -ve")
a=int(input("Enter number to be tested:"))
test(a)
```

```
Enter number to be tested:-1
The number is -ve
```

In [20]:
```
#Example of returning multiple values:
def cal(a,b):
    sum=a+b
    sub=a-b
    mult=a*b
    div=a/b
    return sum,sub,mult,div
a=int(input("Enter a:"))
b=int(input("Enter b:"))
t=cal(a,b)
for i in t:
    print(i)
```

```
Enter a:4
Enter b:2
6
2
8
2.0
```

##Function Specification: Function Specifications One of the the most important aspects of writing a quality Python function is proper specification. While the term may sound generic, a specification actually has a very precise definition and implementation for a Python function. Debugging: It is the process of finding out why a program does not work, and then fixing it.

Docstring: In the programming worls, it is a standard practice to document your code to make for better reading. This can be in the form of comments or doc string. The program should be easily understood by third party.

Commenting and docstrings will also make for easier recall on individual as well as work projects and open source collaboration.

Docstring is a string literal which helps to describe what the function does, not how to use it.

From docstring, 3rd party can understand, better grasp of parameters and conditions what is returned and any errors that may arise.

In practice, a specification is a docstring, a “string literal” that occurs as the first statement in a function, module, class or method, formed by a bracketed set of “”“. 2 types of function specification:

1. One-Liner

2. Multi Liner.

The function specification is everything between the sets of “””. When Python sees this docstring at the front of a function definition, it automatically is stored as the “**doc**” associated with the function. With this specification in place, any user that loads this function can access its **doc** by typing help(radians_to_degrees), which will print the following to the terminal: Here is an example of a simple function I wrote with a specification:

In [26]:
```
#Example of one liner
def square(a):
    """Argument passed into x returns x squared"""
    return a*a
a=int(input("Enter number:"))
print(square(a))
print(square.__doc__)
```

```
Enter number:2
4
Argument passed into x returns x squared
```

In [27]:
```
#Example of multi liner
def radians_to_degrees(theta):
    """
    Returns: theta converted to degrees

    Value return has type float

    Parameter theta: the angle in radians
    Precondition: theta is a float
    """
    return theta * (180.0/3.14159)
print(radians_to_degrees.__doc__)
```

```
    Returns: theta converted to degrees

    Value return has type float

    Parameter theta: the angle in radians
    Precondition: theta is a float
```

Parameters and Arguments

```
The following are the types of arguments that we can use to call a
function:

1. Default arguments
2. Keyword arguments
3. Required arguments
4. Variable-length arguments
```

1. Default Arguments A default argument is a kind of parameter that takes as input a default value if no value is supplied for the argument when the function is called. Default arguments are demonstrated in the following instance.

Prepared by Abhi Shah

In [28]:
```
## Python code to demonstrate the use of default arguments
# defining a function
def function( num1, num2 = 40 ):
    print("num1 is: ", num1)
    print("num2 is: ", num2)


# Calling the function and passing only one argument
print( "Passing one argument" )
function(10)

# Now giving two arguments to the function
print( "Passing two arguments" )
function(10,30)
```

```
Passing one argument
num1 is:  10
num2 is:  40
Passing two arguments
num1 is:  10
num2 is:  30
```

2. Keyword Arguments

The arguments in a function called are connected to keyword arguments. If we provide keyword arguments while calling a function, the user uses the parameter label to identify which parameters value it is.

Since the Python interpreter will connect the keywords given to link the values with its parameters, we can omit some arguments or arrange them out of order. The function() method can also be called with keywords in the following manner:

In [29]:
```
# Python code to demonstrate the use of keyword arguments

# Defining a function
def function( num1, num2 ):
    print("num1 is: ", num1)
    print("num2 is: ", num2)

# Calling function and passing arguments without using keyword
print( "Without using keyword" )
function( 50, 30)

# Calling function and passing arguments using keyword
print( "With using keyword" )
function( num2 = 50, num1 = 30)
```

```
Without using keyword
num1 is:  50
num2 is:  30
With using keyword
num1 is:  30
num2 is:  50
```

3. Required Arguments or Positional Arguments The arguments given to a function while calling in a pre-defined positional sequence are required arguments. The count of

required arguments in the method call must be equal to the count of arguments provided while defining the function.

We must send two arguments to the function function() in the correct order, or it will return a

In [30]:
```
#Example of Required or Positional Argumemnts:
def sub(a,b):
    print(a-b)
sub(100,200)
sub(200,100)
```

```
-100
100
```

4. Variable-Length Arguments We can use special characters in Python functions to pass as many arguments as we want in a function. There are two types of characters that we can use for this purpose:

args -These are Non-Keyword Arguments *kwargs - These are Keyword Arguments.

In [31]:
```
# Python code to demonstrate the use of variable-length arguments

def sum(*n):
    total=0
    for n1 in n:
        total=total+n1
    print("The Sum=",total)
sum(10)
sum(10,20)
sum(10,20,30)
```

```
The Sum= 10
The Sum= 30
The Sum= 60
```

In [32]:
```
#Python code for mixing more than variable
def f(n1,*s):
    print(n1)
    for s1 in s:
        print(s1)
f(10)
f(10,20,30)
f(10,"A")
```

```
10
10
20
30
10
A
```

#Global and Local Variables: All variables in a program may not be accessible at all locations in that program. This depends on where you have declared a variable.

The scope of a variable determines the portion of the program where you can access a particular identifier. There are two basic scopes of variables in Python −

Global variables Local variables Global vs. Local variables Variables that are defined inside a function body have a local scope, and those defined outside have a global scope.

This means that local variables can be accessed only inside the function in which they are declared, whereas global variables can be accessed throughout the program body by all

In [34]:
```
total = 0; # This is global variable.
# Function definition is here
def sum(arg1, arg2):
   # Add both the parameters and return them."
    total = arg1 + arg2; # Here total is local variable.
    print ("Inside the function local total : ", total)
    return total

# Now you can call sum function
sum(10, 20)
print ("Outside the function global total : ", total)
```

```
Inside the function local total :  30
Outside the function global total :  0
```

#Exercise:

In [35]:
```
#1. Write a Python function to find the Max of three numbers.
def max_of_two( x, y ):
    if x > y:
        return x
    return y
def max_of_three( x, y, z ):
    return max_of_two( x, max_of_two( y, z ) )
print(max_of_three(3, 6, -5))
```

```
6
```

In [36]:
```
#2. Write a Python function to calculate the factorial of a number (a non-n
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
n=int(input("Input a number to compute the factiorial : "))
print(factorial(n))
```

```
Input a number to compute the factiorial : 5
120
```

3. Write a Python function to check whether a number is perfect or not. Go to the editor According to Wikipedia : In number theory, a perfect number is a positive integer that is equal to the sum of its proper positive divisors, that is, the sum of its positive divisors excluding the number itself (also known as its aliquot sum). Equivalently, a perfect number is a number that is half the sum of all of its positive divisors (including itself). Example : The first perfect number is 6, because 1, 2, and 3 are its proper positive divisors, and 1 + 2 + 3 = 6. Equivalently, the number 6 is equal to half the sum of all its positive divisors: (1 + 2 + 3 + 6) / 2 = 6. The next perfect number is 28 = 1 + 2 + 4 + 7 + 14. This is followed by the perfect numbers 496 and 8128.

In [37]:
```
#3.
def perfect_number(n):
    sum = 0
    for x in range(1, n):
        if n % x == 0:
            sum += x
    return sum == n
print(perfect_number(6))
```

```
True
```

In [38]:
```
#4. Write a Python Program to make a simple calculator using function:
def add(P, Q):
   # This function is used for adding two numbers
   return P + Q
def subtract(P, Q):
   # This function is used for subtracting two numbers
   return P - Q
def multiply(P, Q):
   # This function is used for multiplying two numbers
   return P * Q
def divide(P, Q):
   # This function is used for dividing two numbers
   return P / Q
# Now we will take inputs from the user
print ("Please select the operation.")
print ("a. Add")
print ("b. Subtract")
print ("c. Multiply")
print ("d. Divide")

choice = input("Please enter choice (a/ b/ c/ d): ")

num_1 = int (input ("Please enter the first number: "))
num_2 = int (input ("Please enter the second number: "))

if choice == 'a':
    print (num_1, " + ", num_2, " = ", add(num_1, num_2))

elif choice == 'b':
    print (num_1, " - ", num_2, " = ", subtract(num_1, num_2))

elif choice == 'c':
    print (num1, " * ", num2, " = ", multiply(num1, num2))
elif choice == 'd':
    print (num_1, " / ", num_2, " = ", divide(num_1, num_2))
else:
    print ("This is an invalid input")
```

```
Please select the operation.
a. Add
b. Subtract
c. Multiply
d. Divide
Please enter choice (a/ b/ c/ d): a
Please enter the first number: 3
Please enter the second number: 4
3  +  4  =  7
```

Prepared by Abhi Shah

In [40]:
```
#5. Python program to check if the number is an Armstrong number or not

# take input from the user
def armstrong(num):


# initialize sum
    sum = 0

# find the sum of the cube of each digit
    temp = num
    while temp > 0:
        digit = temp % 10
        sum += digit ** 3
        temp //= 10

# display the result
    if num == sum:
        print(num,"is an Armstrong number")
    else:
        print(num,"is not an Armstrong number")
num = int(input("Enter a number: "))
armstrong(num)
```

```
Enter a number: 153
153 is an Armstrong number
```

In [42]:
```
#6. Write a Python Program to find the reverse of given number using functi
def reverse(num):
    rev_num=0
    while num!=0:
        digit = num % 10
        rev_num=rev_num*10+digit
        num //= 10
    print("Reverse no.",rev_num)
num=int(input("Enter number:"))
reverse(num)
```

```
Enter number:153
Reverse no. 351
```

In [44]:
```
#7. Write a Python Program to convert Fahrenheit to Celsius temperature
ch=int(input("Enter your choice:"))
def cel():
    f=float(input("Enter temp in Fahrenheit:"))
    c=(f-32)*(5/9)
    print("Temp in Celsius:",c)
def fah():
    c=float(input("Enter temp in Celsius:"))
    f=((9/5)*c)+32
    print("Temp in Fahrenheit:",f)
print("1. Temp in Celsius\n2. Temp in Fahrenheit")
if (ch==1):
    cel()
elif(ch==2):
    fah()
else:
    print("Enter correct choice")
```

```
Enter your choice:1
1. Temp in Celsius
2. Temp in Fahrenheit
Enter temp in Fahrenheit:33
Temp in Celsius: 0.5555555555555556
```

In [45]:
```
#7. Write a Python Program to convert Fahrenheit to Celsius temperature
ch=int(input("Enter your choice:"))
def cel():
    f=float(input("Enter temp in Fahrenheit:"))
    c=(f-32)*(5/9)
    print("Temp in Celsius:",c)
def fah():
    c=float(input("Enter temp in Celsius:"))
    f=((9/5)*c)+32
    print("Temp in Fahrenheit:",f)
print("1. Temp in Celsius\n2. Temp in Fahrenheit")
if (ch==1):
    cel()
elif(ch==2):
    fah()
else:
    print("Enter correct choice")
```

```
Enter your choice:2
1. Temp in Celsius
2. Temp in Fahrenheit
Enter temp in Celsius:95
Temp in Fahrenheit: 203.0
```

Prepared by Abhi Shah

In [46]:
```
#7. Write a Python Program to convert Fahrenheit to Celsius temperature
ch=int(input("Enter your choice:"))
def cel():
    f=float(input("Enter temp in Fahrenheit:"))
    c=(f-32)*(5/9)
    print("Temp in Celsius:",c)
def fah():
    c=float(input("Enter temp in Celsius:"))
    f=((9/5)*c)+32
    print("Temp in Fahrenheit:",f)
print("1. Temp in Celsius\n2. Temp in Fahrenheit")
if (ch==1):
    cel()
elif(ch==2):
    fah()
else:
    print("Enter correct choice")
```

```
Enter your choice:3
1. Temp in Celsius
2. Temp in Fahrenheit
Enter correct choice
```

In [47]:
```
#8. Write a Python Program to calculate fibonacci series using function
def fib(n):
    a = 0
    b = 1
    if n == 1:
        print(a)
    else:
        print(a)
        print(b)
        for i in range(2,n):
            c = a + b
            a = b
            b = c
            print(c)
fib(10)
```

```
0
1
1
2
3
5
8
13
21
34
```

Prepared by Abhi Shah

In [48]:
```
#9. Write a Python Program to find sum and average of first n natural numbe
n=int(input("Enter the number upto which you want the sum:"))
def sum(n):
    sum=0
    for i in range(1,n+1):
        sum=sum+i
        avg=sum/i
    return sum,avg
sum,avg=sum(n)
print("Sum=",sum,"Average=",avg)
```

```
Enter the number upto which you want the sum:7
Sum= 28 Average= 4.0
```

In [49]:
```
#10. Write a Python Program to check whether a number is in a given range u
n=int(input("Enter num:"))
start=int(input("Enter start:"))
stop=int(input("Enter stop:"))
def num(n,start,stop):
    if(n>start and n<stop):
        print("Yes")
    else:
        print("No")
num(n,start,stop)
```

```
Enter num:5
Enter start:2
Enter stop:10
Yes
```

In [50]:
```
#10. Write a Python Program to check whether a number is in a given range u
n=int(input("Enter num:"))
start=int(input("Enter start:"))
stop=int(input("Enter stop:"))
def num(n,start,stop):
    if(n>start and n<stop):
        print("Yes")
    else:
        print("No")
num(n,start,stop)
```

```
Enter num:1
Enter start:2
Enter stop:10
No
```

In [51]:
```
#11. Write a Python Program to demonstrate a function to print the number 1
def printnum():
    for i in range(1,9):
        print(i)
printnum()
```

```
1
2
3
4
5
6
7
8
```

Prepared by Abhi Shah