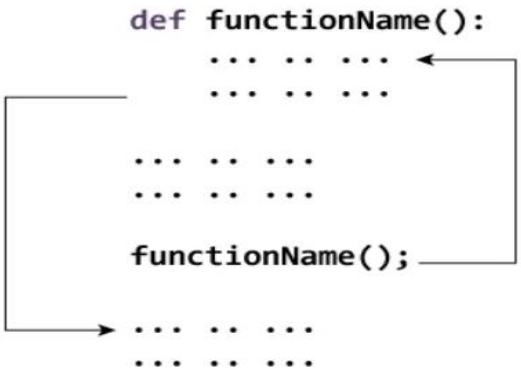
Function:

A group of related statements that performs a specific task, it breaks our program into smaller and modular chunks. It makes a large program more organize and manageable



Syntax:

```
def function_name(parameters):
        """docstring"""
        statement(s)
```

- Keyword def makes the start of the function header.
- Function name to uniquely identify the function. It must be similar to defining identifiers.
- 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.

Example:

```
def hello():      ## Function definition
        print("Hello There")   ##Body of function
hello()           ##Function Calling
```

Output: Hello There

Function with argument:

```
def greet(name):
   print("Hello, " + name + ". Good morning!")
greet('Rahul')
```

Output: Hello, Rahul. Good morning!

Note: the function definition should always be present before the function call. Otherwise, we will get an error. For example,

```
greet('Paul')
def greet(name):
   print("Hello, " + name + ". Good morning!")
```

Output: NameError: name 'greet' is not defined

Return Statement:
The return statement is used to exit a function and go back to the place from where it was called.

Syntax of Return:
return [expression_list]

```
def absolute_value(num):
   """This function returns the absolute
   value of the entered number"""

   if num >= 0:
      return num
   else:
      return -num


print(absolute_value(2))

print(absolute_value(-4))
```

Output:

2

4

Scope and Lifetime of a variable:

Scope of a variable is the portion of a program where the variable is recognized. Parameters and variables defined inside a function are not visible from outside the function have local scope.

Lifetime:

1. The lifetime of a variable is the period throughout which the variable exists in the memory. The lifetime of variables inside a function is if the function executes.
2. They are destroyed once we return from the function. Hence, a function does not remember the value of a variable from its previous calls.

```
def my_func():
    x = 10
    print("Value inside function:",x)

x = 20
my_func()
print("Value outside function:",x)
```

Output:
Value inside function: 10
Value outside function: 20

Some of the build-in functions:

abs()	returns absolute value of a number
all()	returns true when all elements in iterable is true
any()	Checks if any Element of an Iterable is True
ascii()	Returns String Containing Printable Representation
bin()	converts integer to binary string
bool()	Converts a Value to Boolean
bytearray()	returns array of given byte size
bytes()	returns immutable bytes object
callable()	Checks if the Object is Callable
chr()	Returns a Character (a string) from an Integer
classmethod()	returns class method for given function
compile()	Returns a Python code object
complex()	Creates a Complex Number
delattr()	Deletes Attribute From the Object
dict()	Creates a Dictionary

dir()	Tries to Return Attributes of Object
divmod()	Returns a Tuple of Quotient and Remainder
enumerate()	Returns an Enumerate Object
eval()	Runs Python Code Within Program
exec()	Executes Dynamically Created Program
filter()	constructs iterator from elements which are true
float()	returns floating point number from number, string
format()	returns formatted representation of a value
frozenset()	returns immutable frozenset object
getattr()	returns value of named attribute of an object
globals()	returns dictionary of current global symbol table
hasattr()	returns whether object has named attribute
hash()	returns hash value of an object
help()	Invokes the built-in Help System
hex()	Converts to Integer to Hexadecimal
id()	Returns Identify of an Object
input()	reads and returns a line of string
int()	returns integer from a number or string
isinstance()	Checks if a Object is an Instance of Class
issubclass()	Checks if a Class is Subclass of another Class
iter()	returns an iterator
len()	Returns Length of an Object
list()	creates a list in Python
locals()	Returns dictionary of a current local symbol table
map()	Applies Function and Returns a List
max()	returns the largest item
memoryview()	returns memory view of an argument
min()	returns the smallest value
next()	Retrieves next item from the iterator
oct()	returns the octal representation of an integer
open()	Returns a file object
ord()	returns an integer of the Unicode character
pow()	returns the power of a number
object()	creates a featureless object
print()	Prints the Given Object
property()	returns the property attribute
range()	return sequence of integers between start and stop
repr()	returns a printable representation of the object
reversed()	returns the reversed iterator of a sequence
round()	rounds a number to specified decimals
set()	constructs and returns a set

setattr()	sets the value of an attribute of an object
slice()	returns a slice object
sorted()	returns a sorted list from the given iterable
staticmethod()	transforms a method into a static method
str()	returns the string version of the object
sum()	Adds items of an Iterable
super()	Returns a proxy object of the base class
tuple()	Returns a tuple
type()	Returns the type of the object
vars()	Returns the __dict__ attribute
zip()	Returns an iterator of tuples
__import__()	Function called by the import statement

Q1. Functions with pass by value. What will be the output?

```
def evenOdd(x):
   if (x % 2 == 0):
      print("even")
   else:
      print("odd")
```

```
Output:
even
odd
```

Q2. Functions with pass by reference. Guess the output.

```
def myFun(x, y=50):
   print("x: ", x)
   print("y: ", y)


# Driver code (We call myFun() with only
# argument)
myFun(10)
```

Output:

```
('x: ', 10)
('y: ', 50)
```

Q3. Predict the output

```
def student(firstname, lastname):

   print(firstname, lastname)


# Keyword arguments
student(firstname='Python', lastname='Programming')
student(lastname=' Programming', firstname='Python')
```

Output:
Python Programming
Programming Python

4. **Predict the output**

```
def print_temperatures():
  print('temperature in Fahrenheit was:', temp_fahr)
  print('temperature in Kelvin was:', temp_kelvin)

temp_fahr = 212.0
temp_kelvin = fahr_to_kelvin(temp_fahr)

print_temperatures()
```

Output:

```
temperature in Fahrenheit was: 212.0
temperature in Kelvin was: 373.15
```

5. Predict the output

```
def change_list(list1):

    list1.append(20)
    list1.append(30)
    print("list inside function = ",list1)
    #defining the list
list1 = [10,30,40,50]
#calling the function
change_list(list1)
print("list outside function = ",list1)
```

Output:

```
list inside function =  [10, 30, 40, 50, 20, 30]
list outside function =  [10, 30, 40, 50, 20, 30]
```

6. Predict the output

```
def change_string (str):
   str = str + " How are you "
   print("printing the string inside function :",str)

string1 = "Hi I am there"

#calling the function
change_string(string1)

print("printing the string outside function :",string1)
```

Output:
printing the string inside function : Hi I am there How are you
printing the string outside function : Hi I am there

7. Predict the output

```
def myfunc():
        global x
        x = 100
myfunc()
print(x)
```

Output:
100

8. Predict the output

```
x = 400
def myfunc():
        global x
        x = 50
myfunc()
print(x)
```

Output:
50

9. Predict the output

```
def simple_interest(p,t,r):
   return (p*t*r)/100
p = float(input("Enter the principle amount? "))
r = float(input("Enter the rate of interest? "))
t = float(input("Enter the time in years? "))
print("Simple Interest: ",simple_interest(p,r,t))
```

Output:
Enter the principle amount? 1000
Enter the rate of interest? 5
Enter the time in years? 5
Simple Interest: 250.0

10. Predict the output

```
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Sachin")
```

Output:
my_function() missing 1 required positional argument: 'lname'

11. Predict the output:

```
def changeme( mylist ):
   "This changes a passed list into this function"
   mylist.append([1,2,3,4]);
   print("Values inside the function: ", mylist)
   return

# Now you can call changeme function
mylist = [10,20,30];
changeme( mylist );
print("Values outside the function: ", mylist)
```

Output:
Values inside the function: [10, 20, 30, [1, 2, 3, 4]]
Values outside the function: [10, 20, 30, [1, 2, 3, 4]]

12. Predict the output:
A) Function with arbitrary keyword args

```
def greet(*names):
    for name in names:
        print("Hello", name)
greet("Monica", "Luke", "Steve", "John")
```

Output:
Hello Monica
Hello Luke
Hello Steve
Hello John

B) Function with default argument

```
def my_function(country = "Norway"):
  print("I am from " + country)
my_function("Sweden")
my_function("India")
my_function()
```

Output:
I am from Sweden
I am from India
I am from Norway

13. Rohit is new to the English course. He needs to learn all the practice sheets done in the course. He asked her friend Anjali to provide a list of all the practice sheets and their associated marks. He needs to find average marks assigned to a problem. Question with maximum marks and lowest marks. Help Rohit by providing the python script for the same. Create a function to perform the above task

Input:
3

1 Preposition problem 2.5
2 Article Problem 3
3 Spelling Problems 5

Output:
average marks per practice sheet is 3.5 marks
Spelling Problems is problem with maximum marks i.e. 5.0 marks
Preposition problem is problem with minimum marks i.e. 2.5 marks

Description
Input:
first line represents number of practice sheets 'k'
in next 'k' lines, first value represents practice number, followed by title of problem and last value represents marks assigned for that problem.

Output:
average marks
problem title of highest mark problem
problem title of lowest mark problem

Solution:

```
def solution(N):
        assign_ID=[]
        course_title=[]
        marks=[]
        for i in range(N):
        S=list(input().split())
        last=len(S)-1
        assign_ID.append(int(S[0]))
        course_title.append(S[1:last])
        marks.append(float(S[last]))
        avg_marks=sum(marks)/len(marks)
        id_max=marks.index(max(marks))
        id_min=marks.index(min(marks))
        X=' '.join(course_title[id_max])
        Y=' '.join(course_title[id_min])
print('average marks per assignment is {} marks'.format(avg_marks))
print('{} is problem with maximum marks i.e. {} marks'.format(X,max(marks)))
print('{} is problem with minimum marks i.e. {} marks'.format(Y,min(marks)))
```