Functions

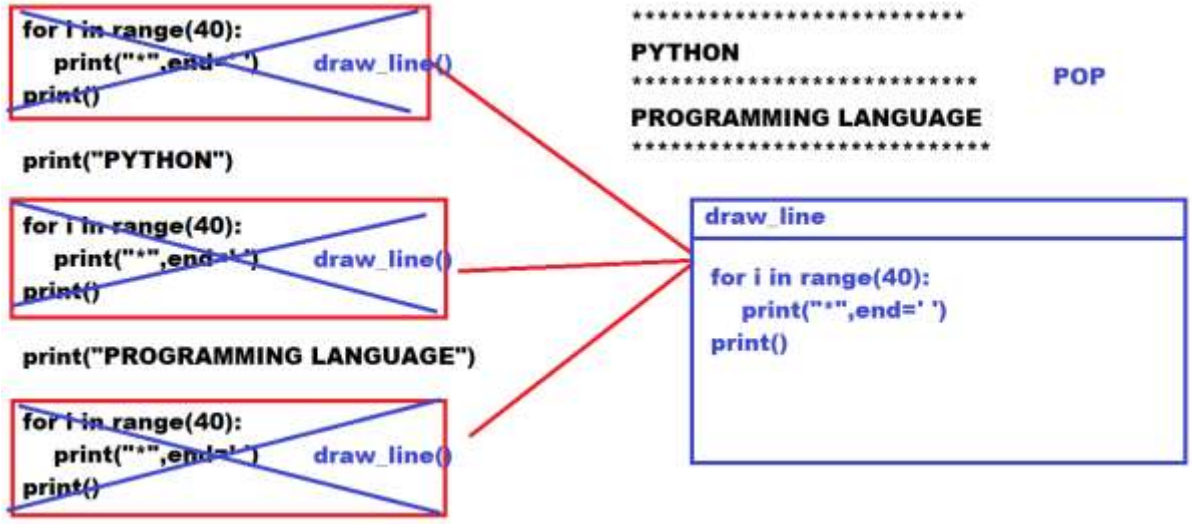
Python is a multi paradigm programming language. A programming paradigm defines set of rules and regulations for writing programs.

Types of programming paradigms

1. Procedural Oriented Programming (POP)
2. Modular Oriented Programming (MOP)
3. Functional Oriented Programming (FOP)
4. Object Oriented Programming (OOP)

Procedural Oriented Programming

In procedural oriented programming, code is organized by dividing according their operations into small pieces called sub routines.
These sub routines can be procedures or functions.
Functions are building blocks of procedural oriented programming.



What is function?

A function is small program inside program.
A function is self contained block, which contains set of instructions to perform operations.
A function is named block.

Advantage of functions

1. **Reusability**: Functions allows writing code once and using many times.
2. **Efficiency**: functions decrease the size of the program, hence it increases efficiency.
3. **Modularity:** dividing programming instructions according to their operations into small pieces.
4. **Readability:** Easy to maintain and understand

Types of functions

Functions are two types

1. Predefined functions
2. User defined functions

Predefined functions

The functions provided by python or third party vendor are called predefined functions. These functions are called library functions.

Example: print(), input(), chr(), ord(),int(),float(),hex(),oct(),…

User defined functions

The functions written by programmer are called user defined functions. These are application specific functions.

Basic steps for working with functions

1. Define function
2. Invoke function

Define function

Defining function is nothing but writing definition of function.

"def" keyword is used for writing function or definition function

Syntax:

```
def <function-name>([parameters]):
        '''doc string'''
        statement-1
        statement-2
```

A function can be defined,

1. Function without parameters without return value
2. Function without parameters with return value
3. Function with parameters without return value
4. Function with parameters with return value

A function with parameters/arguments receives values
A function without parameters/arguments does not receives values

Example:

```
# function without parameters
def sayHello():
    print("Hello Pythonist")

def sayBye():
    print("Bye Pythonist")


# main
sayHello() # invoking
function/calling function
sayHello() # invoking
function/calling function
sayHello() # invokiing
function/calling function
#sayHello(100) Error
sayBye()
sayBye()
```

Output

```
Hello Pythonist
Hello Pythonist
Hello Pythonist
Bye Pythonist
Bye Pythonist
```

Example

```
def draw_line():
    for i in range(40):
        print("*",end=' ')
    print()



draw_line()
```

Output

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
PYTHON
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
PROGRAMMING LANGUAGE
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

print("PYTHON") draw_line() print("PROGRAMMING LANGUAGE") draw_line()	

Local Variables

A variable created inside function or a variable declared inside function is called local variable. Local variables memory is allocated within function context (space). The scope of these variables are within function and life time of these variables are until execution of function.

Local variables memory is allocated when function is called and de-allocated after execution of function.

Whenever function is called, the execution control switched from calling place to called function and after execution of function it returns calling place.

Example:	**Output**

def fun1():
  x=100 # Local variable
  y=200 # Local Variable
  print(x)
  print(y)
 	
def fun2():
  print(x) # Error
  print(y) # Error
 	
fun1()
fun2()
	100
	200
	Traceback (most recent call last):
	  File "E:/python5pmjun/test198.py", line 13, in <module>
	  fun2()
	  File "E:/python5pmjun/test198.py", line 8, in fun2
	  print(x)
	NameError: name 'x' is not defined

Example	**Output**
def add():	Enter value of n1 :5

n1=int(input("Enter value of n1 :")) # L.V
n2=int(input("Enter value of n2 :")) # L.V
n3=n1+n2 # L.V
print(f'sum of {n1} and {n2} is {n3}')
 	
add()
	Enter value of n2 :2
	sum of 5 and 2 is 7

Global Variables