**REPORT ON USER DEFINED FUNCTIONS IN PYTHON**

**Introduction:**

In Python, the functions are the significant building blocks of code that allow modularity and reuse. Functions ensure to tie up the related codes together and allow the programmers to re-use the code that helps them to organize and manage the code efficiently. This report presents a broad overview of user-defined functions in python - their creation and usage, and their benefits.

**What is Function:**

A function contains the block of code and performs its job to fulfill the particular task. Functions in python help to encapsulate code logic into reusable chunks of code, and it gets executed at the time when a function is invoked/called. It can take input values, and it also can return some output values.

**Why Use Functions?**Functions promote: **•Reusability of Code:** one time writing code and reuse that many times.  
**• Modularity:** Break complex problems into simpler sub- problems **• Readability:** code is more readable and easier to organize

**Defining and Calling Functions**

**Defining a Function**

In Python, when a function is declared, it uses the keyword def, followed by a function name and a parenthesis. Parameters could also be added in the parentheses. When a function is declared, there is a colon at the end of that line, and the block of code inside the function is further intended. Its syntax goes like this:

def function\_name(paramaters):  
 #code block  
 return value

**Example:**

def greet(name):  
""" Function to greet a person by name"""  
return f " Hello, {name}!"

**Explanation:**

• ' def ' Means that it is the beginning of a function.  
• 'greet' is the function name  
• 'name:' This is a parameter.  
• 'return' The return statement to send a value back to the caller.

**Calling a Function**Once a function has been defined, it may be called by writing its name followed by parentheses. If the function takes parameters, they are given within the parentheses.  
**Example:**

message=greet(“Alice” )

print(message) # output: Hello, Alice!

**Explanation:**• greet("Alice") calls the function with "Alice" as the argument.  
• The function returns "Hello, Alice!", which is then printed.

**Function Parameters and Arguments**

**Types of parameters**

* **Positional parameters:** Those assigned based on their order in the function call.

def add(x, y):  
 return x + y

* **Keyword Parameters:** The parameters assigned based on the names used in the function call.

def describe\_person(name, age):  
 return f "{name} is {age} years old"

describe\_person(name= "Bob", age=30)

* **Variable-Length Parameters:** These allow for passing a variable amount of arguments using \*args for non-keyword arguments and using \*\*kwargs for keyword arguments.

def print\_scores(\*scores):  
 for score in scores:

print(score)  
print\_scores(90, 85, 88)  
  
def describe\_items(\*\*items):

for item, description in items.items():  
 print( f "{item}:{description}")  
describe\_items(apple= "A red fruit" banana="A yellow fruit" )

**Returning Values**

Functions can return values using the return statement. If a function doesn't have a return statement it will return None by default.

**Example:**

def multiply (a, b):  
 return a\*b

**Explanation:**

* return a\*b returns the product of a and b.

**Scope And Lifetime Of Variables**  
**Local and Global Scope**

* **Local Variables:** This is defined inside the function and accessible only to that function.

def local\_var\_example():  
 local\_var=10  
 return local\_var

* **Global Variables:** It is defined outside any function and thus are accessible from any function in the module.

global\_var = 20  
def global\_var\_example():  
 return global\_var

**The global Keyword**

To modify a global variable within a function, the global keyword is employed.

**Example:**

counter = 0  
  
def increment\_counter():  
 global counter  
 counter +=1

**Advanced Function Concepts**

**Nested Functions**Functions can be defined inside other functions. The inner function is only available within the other function.

Example:

def other-function( ):"  
 def inner\_function( ):  
 return "Hello from the inner function!  
 return inner\_function( )

**Lambda Functions**Lambda function are small anonymous functions defined using the keyword lambda. They can have any number of parameters but one expression.

**Example :**

square= lambda x: x\*x  
print(square(5)) # output: 25

**Explanation:**  
lambda x: x\*x defines a function that returns the square of x.

**Conclusion**User-defined functions in Python have the facility for coding in modular, reusable, and well-organized manners. Besides, defining code functions will facilitate developers to increase readability, handle complexities, and maintain code with much ease. Knowing just how to define and use functions, including parameters, scope, and the use of lambda functions, will provide substantial resources to developers for writing efficient and effective Python code.