→ "Local Vs Global Scope in Python" :→

Python Global variables :→ These are the variables which are not defined inside any function and have a global scope.

Python local variables :→ These are the variables which are defined inside a function and their scope is limited to that function only.

Note :→ Local variables are accessible only inside the function in which it was initialized whereas the global variables are accessible throughout the program and inside every function.

Python local variables :→ Local variables in python are those which are initialized inside a function and belong only to that particular function. It cannot be accessed anywhere outside the function.

Creating local variables in python :→

```
def f():
    s = "I love python"
    print(s)

f()
```

O/p: I love python

Let's see can a local variable be used outside a function?

```
def f():
    s = "I love python"
    print("Inside function", s)

f()
print(s)
```

O/p

Inside function: I love python
NameError: name 's' is not defined.

Python Global Variables :- These are those which are defined outside any function and which are accessible throughout the program, i.e., inside and outside of every function. Let's see how to create a python global variable.

Creating a global variable in python.

```
def f():
    print("Inside function", s)

s = "I love python"
f()
print("Outside function", s)
```

O/P: Inside function I love python
Outside function I love python

Note :- The variable 's' is defined as the global variable and is used both inside the function as well as outside the function.

Note :- if a variable with the same name is defined inside the scope of the function as well, then it will print the value given inside the function only and not the global value.

Eg:-
```
def f():
    s = "Hello"
    print(s)

s = "I love python"
f()
print(s)
```

Hello
I love python.

Now let's see what happens, if we try to change the value of a global variable inside the function?

Eg:→
```
def f():
    s += 'GLP'
    print("Inside function", s)

s = "I love python"
f()
```

O/p:→ Cannot access local variable 's' where it is not associated with a value

Explanation:→ When we use `s += 'GLP'` within the function, Python assumes 's' is a local variable. Since you have not assigned any value to 's' within the function before using it, python raises the UnboundLocalError.

To make above program works, we need to use the "global" keyword in python.

We only need to use the global keyword in a function if we want to do assignments or change the global variable. global is not needed for printing and accessing.

Any variable which is changed or created inside of a function is local if it has not been declared as a global variable. To tell python, that we want to use the global variable, we have to use the keyword "global", as shown:→

Using python global keyword:→

[P.T.O]

```
def f():
    global s
    s += "GLP"
    print(s)
    s = "My name is Khan"
    print(s)

s = "I Love Python"
f()
print(s)
```

Output:

```
I love Python GLP
My name is Khan
My name is Khan
```

Eg2:->

```
a = 1
def f():
    print('inside f() : ', a)
def g():
    a = 2
    print('inside g() : ', a)
def h():
    global a
    a = 3
    print('inside h() : ', a)
print('global : ', a)
f()
print('global : ', a)
g()
print('global: ', a)
h()
print('global : ', a)
```

Output:

```
global : 1
inside f() : 1
global : 1
inside g() : 2
global : 1
inside h() : 3
global : 3
```

[P.T.O]

"Difference b/w Local Variable Vs. Global Variables" :⇒

Comparison basis	"Global Variables"	'Local Variables'
1> Definition	declared outside the function	Declared inside the function
2> Lifetime	They are created when the execution of the program begins & are lost when the program is ended.	They are created when the function starts its execution & are lost when the function ends
3> Data Sharing	offers data sharing	It doesn't offers data sharing
4> Scope	Can be accessed throughout the code.	Can be accessed only inside the function
5> Storage	A fixed location selected by the compiler.	They are kept on the stack.
6> Value	Once the value changes, it is reflected throughout the code.	Once changed, the variable doesn't affect other functions of the program

Global Variables :⇒

Scope: Accessible throughout the entire program or script, including all functions.

Declaration: Defined outside any function or class.

Lifetime: Exists for the duration of the program's execution.

P.T.O

```
global_var = 10
def my_function():
    print(global_var)
```

Local variables:

Scope: Accessible only within the function or block where they are defined.

Declaration: Defined inside a function or block.

Lifetime: Exists only during the function's execution.

Example:→

```
def my_function():
    local_var = 5
    print(local_var)
my_function()
print(local_var)
```

Let's see how can we change a Global variable from inside a function in Python:→

To modify a global variable inside a function, use the global keyword. This tells python that you are referring to the global variable, & not creating a new local one.

Example:→

```
global_var = 10
def change_global():
    global global_var
    global_var = 20
change_global()
print(global_var)
```

Output

20

→ "Import Module in python" :→

In python, modules allow us to organize code into reusable files, making it easy to import and use functions, classes & variables from other scripts.

Note :→ Importing a module in python is similar to using #include in C/C++ providing access to pre-written code and built-in libraries.

→"Importing a module in python" :→ Most common way to use a module is by importing it with the import statement. This allows access to all the functions & variables defined in the module.

Eg:→
```
import math
pie = math.pi
print ("The value of pi is:", pie)
```

O/p: The value of pi is: 3.1415926---

→ "Importing specific functions" :→

Instead of importing the entire module, we can import only the functions or variables we need using the from keyword. This makes the code cleaner & avoids unnecessary imports.

Eg:→
```
from math import pi
print(pi)
```

O/p:→ 3.1415926---

[P.T.O]

→ Importing Built-in modules :→

Python provides many built-in modules that can be imported directly without installation. These modules offer ready-to-use functions for various tasks, such as random number generation, math operations & file-handling.

Eg:→
```
import random
res = random.randint(1, 10)
print("Random Number:", res)
```

O/p:→ Random Number : 9

Note :→ random is an inbuilt module in Python.
random.randint(1,10) generates a random integer between 1 and 10.

→ Importing modules with Aliases :→

To make code more readable and concise, we can assign an alias to a module using the "as" keyword. This is especially useful when working with long module names.

Eg:→
```
import math as m
result = m.sqrt(25)
print("Square root of 25:", result)
```

O/p:→ Square root of 25 : 5.0

Note :→ import math as m imports the math module & assigns it the alias m.

[P.T.O]

→ Importing Everything from a Module (*)

Instead of importing specific functions, we can import all functions and variables from a module using the "*" Symbol. This allows direct access to all module contents without prefixing them with the module name.

Eg:→
```
from math import *
print(pi)
print(factorial(6))
```

O/p: 3.14159265---.
720.

→ "Handling Import Errors in Python" :→

When importing a module that doesn't exist or isn't installed, Python raises an 'ImportError'. To prevent this, we can handle such cases using try-except blocks.

Eg:→
```
try:
    import mathematics          # Incorrect module name
    print(mathematics.pi)
except ImportError:
    print("Module not found! Please check the module name
      or install it if necessary.")
```

Note :→ try block attempts to import a module, if the module is missing or misspelled, Python raises an ImportError.
The except block catches the error & displays a user-friendly message instead of crashing the program.

→ 'Python program to implement Rock, Paper, Scissors game'.

Winning Rules are as follows:→

Rock vs paper → paper wins
Rock vs scissors → Rock wins
Paper vs scissors → scissors wins.

In this game, randint() inbuilt function is used for generating random integer values within a given range.

```
import random
print('Winning rules of the game Rock Paper Scissors are:\n'
    + "Rock vs Paper → Paper wins \n"
    + "Rock vs Scissors → Rock wins \n"
    + "Paper vs Scissors → Scissors wins \n")
While True:
    print("Enter your choice\n 1 - Rock \n 2 - Paper \n 3 -
    Scissors \n")
    choice = int(input("Enter your choice: "))
    While choice > 3 or choice < 1:
        choice = int(input('Enter a valid choice please: '))
    if choice == 1:
        choice_name = 'Rock'
    elif choice == 2:
        choice_name = 'Paper'
    else:
        choice_name = 'Scissors'
```

```
print ('User choice is:', choice_name)
print ("Now it's computer's Turn...")

Comp_choice = random.randint(1, 3)

if comp_choice == 1:
    Comp_choice_name = 'Rock'
elif comp_choice == 2:
    Comp_choice_name = 'Paper'
else:
    Comp_choice_name = 'Scissors'

print ("Computer choice is:", comp_choice_name)
print (choice_name, 'vs', comp_choice_name)

if choice == comp_choice:
    result = "Draw"
elif (choice == 1 and comp_choice == 2) or (comp_choice == 1 and
    choice == 2):
    result = 'Paper'
elif (choice == 1 and comp_choice == 3) or (comp_choice == 1 and
    choice == 3):
    result = 'Rock'
elif (choice == 2 and comp_choice == 3) or (comp_choice == 2 and
    choice == 3):
    result = 'Scissors'

if result == "Draw":
    print ("It is a tie")
elif result == choice_name:
    print("User wins!")
```

[P.T.O]

```
else:
    print("Computer wins!")

print("Do you want to play again? (Y/N)")
ans = input().lower()
if ans == 'n':
    break

print("Thanks for playing!").
```

→ "Hangman Game":

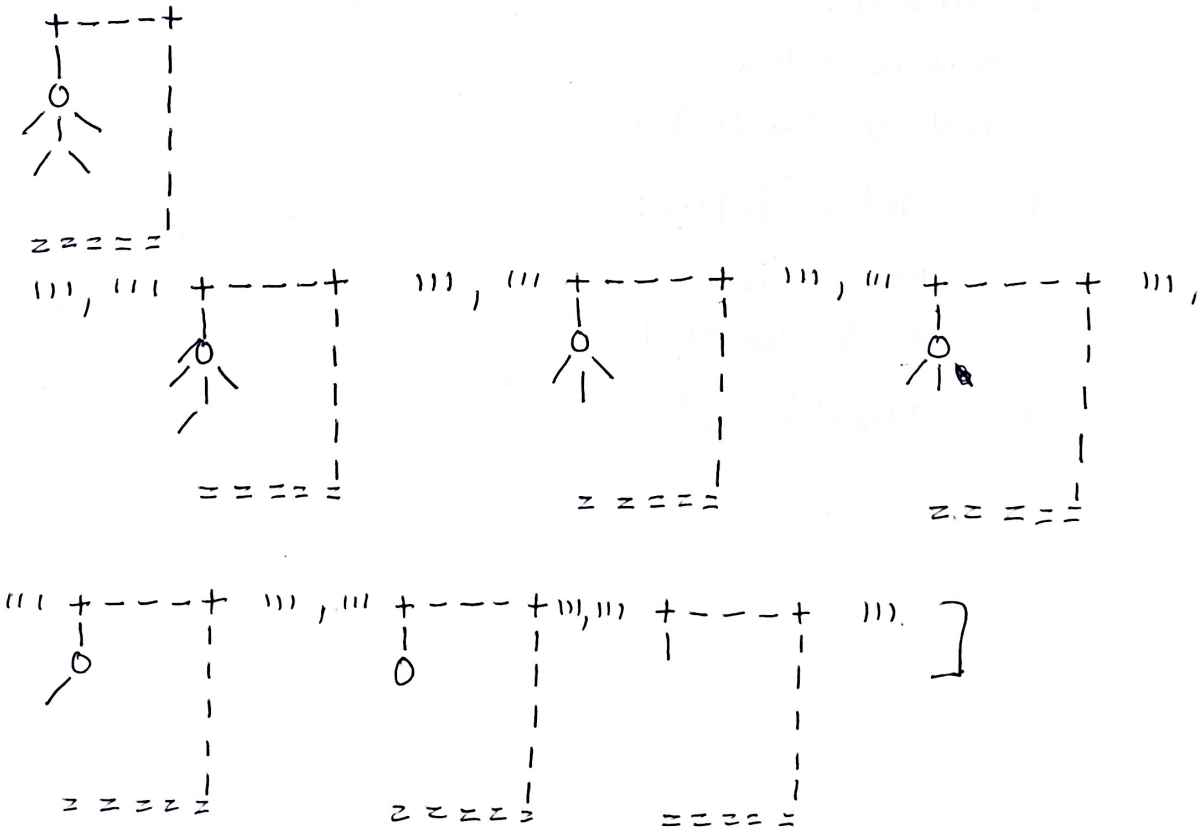
```
import random
word_list = ["apple", "beautiful", "potato"]
chosen_word = random.choice(word_list)
lives = 6
print(chosen_word)
display = []
for i in range(len(chosen_word)):
    display += '_'
print(display)
game_over = False
while not game_over:
    guessed_letter = input("Guess a letter:").lower()
    for position in range(len(chosen_word)):
        letter = chosen_word[position]
        if letter == guessed_letter:
            display[position] = guessed_letter
    print(display)
```

[P.T.O]

```
if guessed_letter not in chosen_word:
    lives -= 1
    if lives == 0:
        game_over = True
        print("You lost!!")
    if '_' not in display:
        game_over = True
        print("You won!!")
```

Let's now mimic creating a hangman too. if we make a wrong choice we need to add an animation also

```
import random
word_list = ["apple", "beautiful", "potato"]
chosen_word = random.choice(word_list)
stages = [''' +---+ | O | /|\ | / \ | ========= ''', ''' +---+ | O | /|\ | / | ========= ''', ''' +---+ | O | /|\ | | ========= ''', ''' +---+ | O | /| | | ========= ''', ''' +---+ | O | | | | ========= ''', ''' +---+ | | | | ========= ''']
```



```
lives = 6
print(chosen_word)
display = []
for i in range(len(chosen_word)):
  display += '_'
print(display)
game_over = False
while not game_over:
  guessed_letter = input("Guess a letter: ").lower()
  for position in range(len(chosen_word)):
    letter = chosen_word[position]
    if letter == guessed_letter:
      display[position] = guessed_letter
  print(display)
  if guessed_letter not in chosen_word:
    lives -= 1
    if lives == 0:
      game_over = True
      print("You lost!!")
    if '_' not in display:
      game_over = True
      print("You won!!")
  print(stages[lives])
```

→ "Password Generator in Python using Random module".

```
import random
letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q',
           'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K',
           'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']

numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

symbols = ['!', '#', '$', '%', '&', '(', ')', '*', '+']

print("Welcome to Password Generator!")
n_letters = int(input("How many letters you want in your password?\n"))
n_symbols = int(input("How many symbols you want in your password?\n"))
n_numbers = int(input("How many numbers you want in your password?\n"))

password = ""
for i in range(1, n_letters+1):
    char = random.choice(letters)
    password += char

for i in range(1, n_symbols+1):
    char = random.choice(symbols)
    password += char

for i in range(1, n_numbers+1):
    char = random.choice(numbers)
    password += char

print(password)
```

Note :- This program will generate a password wherein letters are followed by special symbols followed by numbers always.

Let's shuffle it a bit.

[P.T.O]

Let's modify it a bit.

```
password_list = []
for i in range(1, n_letters+1):
    char = random.choice(letters)
    password_list += char

for i in range(1, n_symbols+1):
    char = random.choice(symbols)
    password_list += char

for i in range(1, n_numbers+1):
    char = random.choice(numbers)
    password_list += char
print(password_list)
random.shuffle(password_list)
print(password_list)

password = ""
for char in password_list:
    password += char
print(password)
```