NameSpace

Scope:

For an example, you are planting a tree in a garden which is inside your house. Now the Tree can be viewed only by your family. but if you plant the same tree outside the house, then the tree can be viewed by anyone from both inside and outside the house.

In [ ]:
```
# Example

enemies = 1

def increase_enemies():
    enemies = 2
    print(f"enemies inside the function ==> {enemies}")

increase_enemies()
print(f"enemies outside the function ==> {enemies}")
```

```
enemies inside the function ==> 2
enemies outside the function ==> 1
```

when you find the output of the previous code, then you can understand the term called ***Global and Local scope.*** otherwise known as ***NameSpace.***

Local Space

Local Space is the variable which cannot be accessed at anywhere of the file. because it's scope is only in local.

example : variable inside a function given below

In [ ]:
```
def  drink_potion():
    potion_strength = 5         # Local Variable, it's scope is only inside the function
    print(potion_strength)

print(potion_strength)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 5
      2     potion_strength = 5
      3     print(potion_strength)
----> 5 print(potion_strength)

NameError: name 'potion_strength' is not defined
```

The above error is because the variable **potion_strength** is not assigned. but it is assigned inside a function. so if we need to print the **potion_strength**, we need to call the fuction, because the variable is local in the function. The Scope of the **Local Variable** is limited (i.e.,) It's scope is within the function and not outside the function anywhere.

In [ ]:
```
def  drink_potion():
    potion_strength = 5         # Local Variable, it's scope is only inside the function
    print(potion_strength)

print(drink_potion())
```

```
5
None
```

Global Space

In **Global Space** the variable is declared outside the function. so that we can use the variable either inside the funtion or outside the function. (i.e.,) anywhere in our file.

Example Below

In [ ]:
```
name = 'aswin'
def print_name():
    print(name)

print(name)             # we can access the variable using variable name
print(print_name())     # and also using functions anywhere
```

```
aswin
aswin
None
```

NOTE:

If an variable is created or assined inside the loop or if statements then it is also Global Scope. Because we can access it outside the block.

In [ ]:
```
game_level = 3
enemies = ['skeleton','alien','zombie']   # Variable Outside the block

if game_level < 5:
    new_enemy = enemies[0]                # Variable Inside the block

print(new_enemy)                          # accessing the variable which is inside the block from outside the block
```

```
skeleton
```

Modifying Global Scope

Yes, we can modify Global Scope by adding **gobal** in front of the variable.

Let's take our 1st Example:

In [ ]:
```
# Example

enemies = 1

def increase_enemies():
    global enemies
    print(f"enemies inside the function ==> {enemies}")

increase_enemies()
print(f"enemies outside the function ==> {enemies}")
```

```
enemies inside the function ==> 1
enemies outside the function ==> 1
```

Applications of *Global*

we can use ***global*** while we are using some constants

In [ ]:
```
# Example

pi = 3.14

def circle():
    global pi
    print(f"the pi value is constant ==> {pi}")

circle()
```

```
the pi value is constant ==> 3.14
```

Task

Create a number Guessing game with following comments

```
#Number Guessing Game Objectives:

# Include an ASCII art logo.(Optional)
# Allow the player to submit a guess for a number between 1 and 100.
# Check user's guess against actual answer. Print "Too high." or "Too low." depending on the user's answer.
# If they got the answer correct, show the actual answer to the player.
# Track the number of turns remaining.
# If they run out of turns, provide feedback to the player.
# Include two different difficulty levels (e.g., 10 guesses in easy mode, only 5 guesses in hard mode).
```

In [ ]:
```
# Include an ASCII art logo.(Optional)

# Step 2 ==> generating random number between 1 - 100
import random as r
actual_num = int(r.randint(1,100))
# print(f"the actual number is {actual_num}")

# Step 3 ==> Allow the player to submit a guess for a number between 1 and 100.
def user_guess():
    guessed_num = int(input("enter the guessed number between 1 to 100"))

    # Step 4 ==> Check user's guess against actual answer. Print "Too high." or "Too low." depending on the user's answer.
    if guessed_num < actual_num:
        print("Too low")
    elif guessed_num > actual_num:
        print("Too high")

    # Step 5 ==> If they got the answer correct, show the actual answer to the player.
    else:
        print(f" Yes you got the answer ==> {actual_num}")
        return True
    return False


# If they run out of turns, provide feedback to the player.

# Step 1 ==> Include two different difficulty levels (e.g., 10 guesses in easy mode, only 5 guesses in hard mode).
game_mode = input("which mode do you want to prefer ==> 'easy' or 'hard : ")
guess = 0
if game_mode == 'hard':
    guess += 5
if game_mode == 'easy':
    guess += 10
print(f"You have {guess} more lives")

while guess!=0:
    # Step 6 ==> Track the number of turns remaining.
    if user_guess():
        break
    guess -= 1
    # user_guess()
    print(f"You have {guess} more lives")

if guess == 0:
    print(f"you had run out of lives, and the actual number is {actual_num}")
```

```
You have 10 more lives
Too low
You have 9 more lives
Too low
You have 8 more lives
Too low
You have 7 more lives
Too high
You have 6 more lives
Too high
You have 5 more lives
Too low
You have 4 more lives
Too low
You have 3 more lives
Too low
You have 2 more lives
Too low
You have 1 more lives
```

```
Too low
You have 0 more lives
you had run out of lives, and the actual number is 78
```