Day - 12

**Scope & Number Guessing Game**

Scope, Namespace, Local and Global variables

**12.1 Scope and Namespace**

|  |  |
| --- | --- |
| * Local Scope: variables, names, functions, blocks defined inside a function. Cant be accessible outside the parent function. * Global Scope: variables, names, functions, blocks defined outside a function. Accessible to anywhere. * NameSpace: The general idea of Scope is called namespace to reduce name-collision. | D:\100_days_comer_soln\image.png |

|  |  |  |
| --- | --- | --- |
| Local Scope | Global Scope | Namespace |
| * exist within a function * only valid within the walls of this function | * valid anywhere within the python file | * valid in certain scope |

**12.2 No block scope in python**

* There is no block scope in python unlike **Java**, **C++**. Variables within if, else, for, while "code blocks with colon and indentation", they don't count as creating a ***local scope/ fence***. Scope remains same in all Kind of while, for, if ***CTRL statements***.
* If ***CTRL*** statements ***while***, ***for***, ***if*** , blocks are inside a function then functions-scope will applied to those ***CRTL-statements***.

|  |  |
| --- | --- |
| #*There is no block-scope in py*  #*Scope remains same in all Kind of while, for, if CTRL statemnt*  t = 10  **if** t **<** 34 :      p = 45  #*variable p can be accessed outside "if" block*  **print**(p) | * Indenting is needed to use local variables.   **def** **out\_scope**():      local\_var = 123  **print**(local\_var)    #*follwing throws error*  #*print(local\_var)* |

**12.3 Modify a Global Variable from Local-SCOPE**

* Because of Namespace, no name-collision arise in any scope. i.e. *identical* *names* for variables doesn’t *collide* in different *scope*.
* Variables with identical names act as independent separate variables in different scope. For example: in following test\_var inside function and outside function are completely different:

#*Global variables can be accessed and modified within function by keyword "global"*

#*BUT not a good idea. Instead "return" stmnt should use*

glbl\_var = 100

**def** **crazy\_scope**():

    #*Following looks for the specified global variable outside current scope*

**global** glbl\_var

    glbl\_var = 300

**print**(f"Original global variable : {glbl\_var}")

**crazy\_scope**()

**print**(f"after function call global variable : {glbl\_var}")

#*Identical named local-global*

test\_var = 700

**def** **scope\_demo2**():

    test\_var = 100

**print**("Inside funtion", test\_var)

**scope\_demo2**()

**print**("Outside funtion", test\_var)

**12.4 Constant Conversion for Global variable**

#*Global variables are good for constants. Can be used in "local scope"*

#*Conventionally Upper-case letters are used for constants.  Eg:*

PI = 3.14159

HTTP = "https://google.com"

**def** **local\_demo**():

    t= PI/2.0

**return** f"Pi : {PI} and 1/2 Pi: {t}"

**print**(**local\_demo**())

* **ASCII art generator**

Any Text to ASCII art: <https://patorjk.com/software/taag/#p=display&f=Big&t=Piru%20%20%20%20lina>

* Exercise 12.1 Number guessing game.

Practice version

#*-----------------------------<My code>----------------------------*

logo = """

  \_   \_                 \_                  \_\_\_\_\_                     \_                \_\_\_\_\_

 | \ | |               | |                / \_\_\_\_|                   (\_)              / \_\_\_\_|

 |  \| |\_   \_ \_ \_\_ \_\_\_ | |\_\_   \_\_\_ \_ \_\_  | |  \_\_ \_   \_  \_\_\_  \_\_\_ \_\_\_ \_ \_ \_\_   \_\_ \_  | |  \_\_  \_\_ \_ \_ \_\_ \_\_\_   \_\_\_

 | . ` | | | | '\_ ` \_ \| '\_ \ / \_ \ '\_\_| | | |\_ | | | |/ \_ \/ \_\_/ \_\_| | '\_ \ / \_` | | | |\_ |/ \_` | '\_ ` \_ \ / \_ **\**

 | |\  | |\_| | | | | | | |\_) |  \_\_/ |    | |\_\_| | |\_| |  \_\_/\\_\_ \\_\_ \ | | | | (\_| | | |\_\_| | (\_| | | | | | |  \_\_/

 |\_| \\_|\\_\_,\_|\_| |\_| |\_|\_.\_\_/ \\_\_\_|\_|     \\_\_\_\_\_|\\_\_,\_|\\_\_\_||\_\_\_/\_\_\_/\_|\_| |\_|\\_\_, |  \\_\_\_\_\_|\\_\_,\_|\_| |\_| |\_|\\_\_\_|

                                                                              \_\_/ |

                                                                             |\_\_\_/

"""

**import** random

**def** **compare**(usrNnum, rndNum):

**if** usrNnum **==** rndNum:

**return** "correct"

**elif** (rndNum - 10) **<** usrNnum **<** (rndNum + 10):

**if** (rndNum - 5) **<** usrNnum **<** (rndNum + 5):

**if** usrNnum **<** rndNum :

**return** f"Guess again a little bit higher."

**elif** usrNnum **>** rndNum :

**return** f"Guess again a little bit lower."

**else**:

**if** usrNnum **<** rndNum :

**return** f"Guess a bit bigger number."

**elif** usrNnum **>** rndNum :

**return** f"Guess a bit smaller number."

**else**:

**if** usrNnum **<** rndNum :

**return** f"too small"

**elif** usrNnum **>** rndNum :

**return** f"too big"

game\_over = **False**

**while** **not** game\_over:

**print**(logo)

    rand\_num = random**.randint**(1, 100)

**print**("random number is", rand\_num)

    ask = **input**("\n\t Want to play 'Hard' or 'Easy', input 'h'/'e' : ")**.lower**()

**if** ask **==** 'h':

        count = 5

**elif** ask **==** 'e':

        count = 10

**else**:

**print**("Invalid char. Using easy mode")

        count = 10

    i = 0

**while** (i **<** count):

        user\_guess = **int**(input("Enter a Number between 1 to 100: "))

**if** **compare**(user\_guess, rand\_num) **!=** "correct":

**print**("Chances left : ", count-1-i)

**print**(**compare**(user\_guess, rand\_num))

            i += 1

**elif** **compare**(user\_guess, rand\_num) **==** "correct":

**print**("Well done!!!! you guessed the correct number!!!")

            i = count

**if** **input**("Wanna play again ? y/n :") **==** 'n':

        game\_over = **True**

#*python number\_gussing.py*

Instructor version

|  |
| --- |
| [ASCII generator](http://patorjk.com/software/taag/#p=display&f=Big&t=Number%20Guessing%20Game) |

#*-----------------------------<My code>----------------------------*

logo = """

  \_   \_                 \_                  \_\_\_\_\_                     \_                \_\_\_\_\_

 | \ | |               | |                / \_\_\_\_|                   (\_)              / \_\_\_\_|

 |  \| |\_   \_ \_ \_\_ \_\_\_ | |\_\_   \_\_\_ \_ \_\_  | |  \_\_ \_   \_  \_\_\_  \_\_\_ \_\_\_ \_ \_ \_\_   \_\_ \_  | |  \_\_  \_\_ \_ \_ \_\_ \_\_\_   \_\_\_

 | . ` | | | | '\_ ` \_ \| '\_ \ / \_ \ '\_\_| | | |\_ | | | |/ \_ \/ \_\_/ \_\_| | '\_ \ / \_` | | | |\_ |/ \_` | '\_ ` \_ \ / \_ **\**

 | |\  | |\_| | | | | | | |\_) |  \_\_/ |    | |\_\_| | |\_| |  \_\_/\\_\_ \\_\_ \ | | | | (\_| | | |\_\_| | (\_| | | | | | |  \_\_/

 |\_| \\_|\\_\_,\_|\_| |\_| |\_|\_.\_\_/ \\_\_\_|\_|     \\_\_\_\_\_|\\_\_,\_|\\_\_\_||\_\_\_/\_\_\_/\_|\_| |\_|\\_\_, |  \\_\_\_\_\_|\\_\_,\_|\_| |\_| |\_|\\_\_\_|

                                                                              \_\_/ |

                                                                             |\_\_\_/

"""

**from** random **import** randint

#*generate random number*

EASY\_LEVEL = 10

HARD\_LEVEL = 5

**def** **check\_answer**(guess, answer, turns):

  """ chekcs answer against guess, returns the number of turns remaining """

**if** guess **==** answer:

**print**(f"Congrats. Answer was {answer}")

**if** **input**("restart? y or n: ") **==** "y":

**game**()

**return**

**elif** guess **>** answer:

**print**("Too high")

**return** turns -1 #*before, I used += which is bound to variable*

**elif** guess **<** answer:

**print**("Too low")

**return** turns -1

#*set difficulty and attempts*

**def** **difficulty**():

  level = **input**("Choose a difficulty. easy or hard?: ")

  #*in this stage, creating global constants*

     #*if I create variable'turns' using =, inside function,  it's no use as it's local,*

     #*instead, use "return" so I can use it whenever I call this function:  set global var and assign the function to get output.*

**if** level **==** "easy":

**return** EASY\_LEVEL

**print**("10 attempts")

**else**:

**return** HARD\_LEVEL

**print**("5 attempts")

**def** **game**():

  answer = **randint**(1, 100)

**print**("welcome")

**print**(answer)

  #*loop user guess - compare - result if they get it wrong. if get it right, finish.*

  turns = **difficulty**()

  #*setting local variable turns,output of difficult() is EASY\_LEVEL OR HARD\_LEVEL  which contains 10, 5 each and it's number of turns.*

  guess = 0

  #*when creating while loop, guess is defined after while, so make an empty one first. it will be only used once.*

**while** guess **!=** answer **and** turns **!=** 0:

    #*get user guess*

**print**(f"you have {turns} attempts remaining to guess the number.")

    guess = **int**(input("Guess a number btw 1 to 100: "))

    #*compare guess with answer #show result*

    #*track thenumber of turns, reduce by 1 if they get it wrong*

    turns = **check\_answer**(guess, answer, turns) #*check\_answer returns output: turns -1 . Also prints direction*

**if** turns **==** 0:

**print**("you've run out of guess the number")

**if** **input**("restart? y or n: ") **==** "y":

**game**()

**return** #*🚨 to exit and end function*

**game**()

#*python number\_gussing\_instructor.py*

Remarks:

**return** turns -1 is used because, turns -= 1 is bound to variable.