Assignment No:-2

Title:- Write python code for Guessing game / Hangman

problem statement:-To python code for guessing game / Hangman.

Pre-lab :- a basic understanding of computer Programming terminologies. a basic understanding of any of the programming language will help in understanding the python programming and data Science Concepts

Theory :-

- This script / Program is an interactive guessing game, which will ask the user to guess a number between 1 and 99
- we are using the random module with the randint function to get a random number. the script also contains a while loop which make the script run until the user guess the right number. if you read my previous post about conditional statements in python. you will also recognize the if, elif and else statement.
- The script program is an interactive Hangman game which will ask the user to guess a
- This is a python script of the classic game "Hangman". The word to guess is represented by row of dashes. if the player guess a letter which exists in the word, the script writes it in all its correct possitions.

- The player has to turns to guess the word you can easily customize the game by changing the Variables.

Decisions :-

- The if-else is used to make Choice in python code this is a compound statement, the simplest form is

```
if condition:
    action-1
else:
    action-2
```

- The indentation is required. Not that the else and it's action are optional. the actions action-1 and action-2 may consist of many statements. then they must all be indented the same amount - the condition is an expression which evaluates to true or false. of Course, if the Condition evaluated to True then action-1 is exicuted, otherwise action-2 is executed. In either Case execution continues withe statement after the if-else

for example, the code.

```
x = 1
if x > 0:
print " friday is wonderful"
else:
    print " monday sucks"
print " Have a good weekend"
```

output:-

Friday is wonderfay
Have a good weekend.

- The last print statement is not part of the if-else statement (because it isn't indented). so if we change the first ~~time~~ line to say x=0. then the output would be.

Monday Sucks.
Have a good weekend

- More complex decision may have several alternatives depending on several conditions. for this the elif is used.
- It means "else if" and one can have any number of elif becauses between. the if and the else. the usage of elif is best illustrated by an example.

```
if x>=0 and x<10:
    digits=1
elif x>=10 and x<100:
    digits=2
elif x>=100 and x<1000:
    digits=3
elif x>=1000 and x<10000:
    digits=4
else:
    digits=0        # more than 4.
```

Loops :-

- python provides two looping commands:
  1) for
  2) while
  - these are compound commands

1) for loop -

Syntax of for loop is :-

```
for item in list:
    action
```

- As usual the action consists of one or more statements, all at the same indentation level, these statements are also known as the body of the loop
- the item is a variable name, and list is a list.
- Execution of the for loop works by setting the variable successively to each item in the list, and then executing the body each time.

example:

```
for i in [2,4,6,0]:
    print i
```

output :-

2 4 6 0

2) while loop :-

Syntax of a while loop is :-

- while condition:
    action.

- of cource the action may consists of one or more statement of all at the same indentation level.
- The statements in the action are known as the body of the loop. Execution of the loop works as follows:
- first the condition is evaluated. if true, the body is executed and the condition evaluates to false the ~~first time~~ again. and this repeats untill the
- if the ~~bod~~ condition evaluates to false example

```
n=0
while n<10:
  print n,
  n=n+3
```

This produces the following output.

0 3 6 9

- The body of while loop never executed if the condition evaluates to false the first time.
- if the body does not change the subsequente evaluations of the conditions, an infinite loop may occur.

for example:

```
while True:
  print "Hello".
```

it will print Hello endlessly. to interrupt of an infinite loop, use CTRL-C.

else in loops :-

- A loop may have an optional else which is executed when the loop finishes.

ex.

```
for n in [10,9,8,7,6,5,4,3,2,1]:
    print n
    else:
        print "blastoff"
```

result in the output

10 9 8 7 6 5 4 3 2 1 blastoff

and the loop

```
n=10
while n>0:
    print n,
    n=n-1
else:
    print "blastoff"
```

break, continue and pass statement :-

- The break statement, like in c breaks out the smallest enclosing for or while loop. the continue statement also borrowed from C, continues with the next iteration of the loop.
- the pass statement does nothing. it can be used when a statement is required syntactically but the ~~statemen~~ program requires no action.
- here is an example of the use of a break statement and an else clause in a loop.

```
for n in range(2,10):
    for x in range(2,n):
        if n%x==0:
            print n,'equals', x,'*',n/x
            break
    else:
    # loop fell through without finding a factor
    print n,'is a prime number.'
```

Lists:-

- A list is a finite sequence of items and one could use the range function to Create list of integers.
- In python, lists are not required to be homogeneous i.e. the items in the list could be different types.

for ex:-

```
a=[2,"Jack",45,"23 wentworth Ave"]
```

- it is a perfectly valid te list consisting of two integers and two string.
- one can refer to the entire list using the identifier a or to the i-th item in the list using a[i].

```
>>>a=[2,"Jack",45,"23 wentworth Ave"]
>>>a
[2,"Jack",45,"23 wentworth Ave"]
>>>a[0]
2
```

```
    2
>>>a[1]
'Jack'
>>>a[2]
  45
>>>a[3]
'23 wentworth Ave'
```

- The numbering of list items always begins at 0 in python so four items in the above list are indexed by 0, 1, 2, 3.
- list items may be assigned a new value this of course changes the list.

for example:-

```
>>>a
[2,"Jack",45,'23 wentworth Ave']
>>>a[0]=2002
>>>a
[2002, 'Jack', 45, '23 wentworth Ave']
```

length of a list:-

- Every list has a length, the number of items in the list obtained using the len function:

```
>>>x=[9,4,900,-45]
>>>len(x)
4
```

- Empty list of length 0 this is created as follow

```
>>>x = [ ]
>>>len(x)
0
```

Sublists (slicing) :-

-sublists are obtained by slicing.

-if the x is an existing list. then x[start:end] is the sublist consisting of all items in the original list at index positions i such that

start ≤ i < end

-the indexing items always starts at 0 in python

ex:

```
x = range(0, 20, 2)
>> x
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
>>>x[2:5]
[4, 6, 8]
>>>x[0:5]
[0, 2, 4, 6, 8]
```

-one may clearly use a negative increment to effectively reverse a list.

```
>>>list[18::-1]
[17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

-Two existing list may be can catenated together to make a longer list, using + operator:

```
>>>[2,3,6,10] + [4,0,0,5,0]
 [2,3,6,10,4,0,0,50]
```

List Methods:-

① append:-

-if x is the name of an existing list, we can append an item to the end of the list using.

```
x.append(item)
```

example:-

```
>>>x=[3,6,8,9]
>>>x.append(999) >>>x
[3,6,8,9,999]
```

② Insert:-

- it allows an element to be inserted in the list at a specified position:

```
>>>x=['a','c','3','d','7']
>>>x.insert(0,100)
>>>x
[100,'a','c','3','d','7']
>>>x.insert(3,'Junk')
>>>x
[100,'a','c','Junk','3','d','7']
```

⑤ remove :-
-delete the first occurence of some item in the list (if possible) using remove as follows:

```
>>>x.remove('a')
>>>x
[100,'c','Junk','3','d','7']
```

- To delete the item at index position i use x.pop(i)

```
>>>x.pop(0)
  100
>>>x
['c','Junk','3','d','7']
```

- by defaut x.pop() pops the last item in list

```
>>>x.pop()
'7'
>>>x
['c','Junk','3','d']
```

Strings :-
- A string in python is a sequence of characters
- In some sense strings are similar to the lists however there are important difference. one major difference is that python strings are immutable.
ex

```
x='gobbletygook'
>>>x[2]
'b'
>>>x[5]
'e'
```

- String items are indexed starting at 0.
- slicing for strings works exactly the same as for list.
- the length function len is same as for list and the concatenation also.
- but the list methods append, insert, delete & pop are not available for strings, because strings are immutable.

Algorithm: Guessing game.

```
import random
n=random.randint(1,99)
guess=int(raw_input("Enter an integer from 1 to 99"))
while n!="guess":
    print
    if guess<n:
        print "guess is low"
        guess=int(raw_input("Enter an integer from
                                1 to 99"))
    elif guess>n:
        print "guess is high"
        guess=int(raw_input("Enter an integer from
                                1 to 99:"))
    else:
        print "you guessed it!"
        break
        Print
```

Algorithm 2 Hangman:

```
# importing the time module
import time
# welcoming the user:

name = raw_input("what is your name?")
print("Hello, " + name, "Time to play hangman!"
print " "
# wait for 1 second
time.sleep(1)
print "start guessing..."
time.sleep(0.5)
# here we set the secret:
word = "secret"

# Creates an variable with an empty value.
guess = ''
# determine the number of turns.
turns = 10
# creates a while loop
# check if the turns are more than zero.
while turn > 0:
# make a Counter that starts with zero
failed = 0
# for every character in secret_word
for char in word:
# see if the character is in the players guess
if char in guess:
```

```
#print then out the character
  print Char,
 else:
 # if not found, print a dash
    print "_"
 # and increase the failed counter with one
   failed+=1
#if failed is equal to zero
 #print you won
  if failed == 0:
  print "you won"
# exis the script
    break
 print
# ask the user go guess a character
guess= row_input ("guess a character:")
# set the players guess to guesses:
    guesses +=guess
# if the guess is not found in the secret word
   if guess not in word:
#turns counter decreases with 1(now 9)
       turns-=1
  #print wrong
     print "wrong"
  #how many turns are left
    print "you have", +turns, 'more guesses'
  #If the turs are equal to zero
    If turns==0:
  #print "you loose"
    PRint "you loose"
```

Conclusion:- Thus we have studied different Condition statements, loops and implement the hangman game using this statement.