Universidad Autónoma de Baja California
Facultad de Ingeniería Arquitectura y Diseño

Ingeniero en computación

Materia: Lenguaje de programación Python

Alumno: Marcos Ruíz González

Matrícula: 361603

Maestro: Pedro Núñez Yépiz

Actividad No. 8

Tema - Unidad: Listas y modulo random

Ensenada Baja California a 11 de octubre del 2023

1. INTRODUCCIÓN

Una lista es similar a una tupla con la diferencia fundamental de que permite modificar los datos una vez creados.

mi_lista = ['cadena de texto', 15, 2.8, 'otro dato', 25]
A las listas se accede igual que a las tuplas, por su número de índice:
print mi_lista[1] # Salida: 15
print mi_lista[1:4] # Devuelve: [15, 2.8, 'otro dato']
print mi_lista[-2] # Salida: otro dato
Las listas NO son inmutables: permiten modificar los datos una vez creados:
mi_lista[2] = 3.8 # el tercer elemento ahora es 3.8
Las listas, a diferencia de las tuplas, permiten agregar nuevos valores:
mi_lista.append('Nuevo Dato')

2. COMPETENCIA

Emplear funciones en lenguaje interpretado, por medio de la discusión de su sintaxis y la experimentación en un ambiente interactivo de ejecución, para redactar funciones que puedan ser ejecutadas repetidas veces sin la reescritura de sus líneas de código, con determinación y creatividad.

3. FUNDAMENTOS

Las listas en Python son un tipo de dato que permite almacenar datos de cualquier tipo. Son mutables y dinámicas, lo cual es la principal diferencia con los sets y las tuplas.

Las listas en Python son uno de los tipos o estructuras de datos más versátiles del lenguaje, ya que permiten almacenar un conjunto arbitrario de datos. Es decir, podemos guardar en ellas prácticamente lo que sea. Si vienes de otros lenguajes de programación, se podría decir que son similares a los arrays.

```
lista = [1, 2, 3, 4]
```

También se puede crear usando `list` y pasando un objeto iterable.

```
lista = list("1234")
```

Una lista sea crea con `[]` separando sus elementos con comas `,`. Una gran ventaja es que pueden almacenar tipos de datos distintos.

```
lista = [1, "Hola", 3.67, [1, 2, 3]]
```

Algunas propiedades de las listas:

- Son **ordenadas**, mantienen el orden en el que han sido definidas
- Pueden ser formadas por tipos **arbitrarios**
- Pueden ser **indexadas** con `[i]`.
- Se pueden **anidar**, es decir, meter una dentro de la otra.
- Son **mutables**, ya que sus elementos pueden ser modificados.
- Son **dinámicas**, ya que se pueden añadir o eliminar elementos.

4. PROCEDIMIENTO

1.- Programa en Python que genere un número entre el 1 y 10 **(no visible)** preguntarle al usuario que número cree que generó la computadora, el usuario tendrá 3 oportunidades de adivinar.
Decir si adivino o si falló en sus 3 intentos.
NOTA: 100% VALIDADO (usar función para validar números)
NOTA2: El usuario podrá jugar cuantas veces lo desee, al final del juego desplegar cantidad de ganados y perdidos
2.- El **juego Busca Número** muestra una lista de 10 números, sin mostrar su contenido,
Al usuario se le muestra un número que se generó aleatoriamente y el usuario tendrá 3 intentos de adivinar en qué índice del arreglo se encuentra.
El usuario recibirá un mensaje que diga **GANASTE, PERDISTE, TIENES UN NUEVO INTENTO**
El usuario podrá jugar cuantas veces lo desee.
NOTA: La lista se deberá llenar con números aleatorios del 1 al 10 no repetidos.
(Hacer una función que regrese la lista con los 10 números sin repetir)
NOTA: REALIZA 3 VERSIONES DIFERENTES DEL LLENADO DE LA LISTA ALEATORIA
VERSIÓN A) Usar ciclos para validar los repetidos, y sólo random para generar los números dentro del rango
VERSIÓN B) Usa funciones de la librería Random para llenar la lista con los números sin repetir
VERSIÓN C) Usa funciones de la librería Random para llenar la lista con los números sin repetir

5. RESULTADOS Y CONCLUSIONES

En esta practica se realizaron ejercicios con el fin de practica el aprendizaje obtenido sobre las listas en Python, así como el reforzamiento del uso del modulo RANDOM. A su vez también se refuerza el conocimiento de la correcta estructuración del código, el uso de ciclos y condiciones para la elaboración de diversas funciones.

6. ANEXOS

```
[1] import random
```

```
[64] def validnum(ri, rf, msge):
        while True:
            num=int(input(f"{msge}"))
            if num>=ri and num<=rf:
               break
            else:
                continue
        return num
```

```
wins = 0
lose = 0

while True:
    opor=0
    numrand = random.randint(1,10)
    print(f"{numrand}")
    while opor<3:
        num = validnum(1, 10, f"Adivina el numero secreto entre 1 y 10\n")
        if num == numrand:
            print("FELICIDADES GANASTE")
            wins += 1
            break
        opor+=1
    else:
        print("LO SIENTO PERDISTE")
        lose += 1
    opc=int(input("¿Jugar de nuevo? \n 1.-SI  2.-NO"))
    if opc==1:
        continue
    elif opc==2:
        break
print(f"TERMINO EL JUEGO \nGANADOS: {wins} \nPERDIDOS: {lose}")
```

```
opc = int(input("Elige la version del juego que quieres \n1.- Version A \n2.- Version B \n3.- Version C\n"))
if opc == 1:
    VersionA()
elif opc == 2:
    VersionB()
elif opc == 3:
    VersionC()
```

VERSIÓN A

```
[57] def VersionA():
        seguir = 1
        while seguir!=0:
            lista = []
            numsec = random.randint(1,10)
            i = 0
            while i < 10:
                num = random.randint(1,10)
                if num not in lista:
                    lista.append(num)
                    i+=1
                else:
                    continue
            i = 0
            while i<3:
                num = int (input(f"Dime en que posicion crees que se encuentra el numero: {numsec} Tienes 3 intentos \n"))
                if numsec in lista:
                    indice = lista.index(numsec)
                    if num == indice:
                        print("FELICIDADES ADIVINASTE LA POSICION")
                        break
                    elif i<2:
                        print("Cometiste un error vuelve a intentarlo")
                        i+=1
                    elif i==2:
                        print("Cometiste 3 errores has perdido")
                        i+=1
            print (f"{num} se encuentra en el indice: {indice}")
            seguir = int(input("Jugar de nuevo? 1.- SI 0.- NO \n"))
```

VERSIÓN B

```
[58] def VersionB():
```