Sesion 33

Emplea las tuplas y diccionarios de Python en diversos programas usando funciones, donde los datos requieren un tratamiento especial.

1. Introducción:

El tema propuesto muestra a los alumnos los tipos de listas que puede implementar, para mejorar el registro y acceso de datos, estos se implementarán con funciones para hacer más relevante el tema.

MODULO 4: ¿Cómo definir y utilizar funciones?, ¿Cómo pasar argumentos y las distintas formas de hacerlo?, El alcance de los nombres, Tuplas y diccionarios, Procesamiento de datos.

2. Tuplas y Diccionarios

a. Secuencia y mutabilidad.

Un tipo de secuencia es un tipo de dato en Python el cual es capaz de almacenar más de un valor (o ninguno si la secuencia está vacía), los cuales pueden ser secuencialmente (de ahí el nombre) leídos, elemento por elemento (a través de un índice o accediendo por medio de instrucciones de alguna sentencia), para ellos se hace uso de una sentencia de bucle.

Un ejemplo de tipo de dato secuencia son las listas, que hemos trabajado en los capítulos anteriores.

La mutabilidad, es una propiedad de cualquier tipo de dato en Python que describe su disponibilidad para poder cambiar de valor libremente durante la ejecución de un programa. Existen dos tipos de datos en Python: mutables e inmutables.

Se indica que un dato es mutable cuando se puede alterar e inmutable cuando estos valores una vez asignados ya no pueden modificarse.

b. ¿Qué es una Tupla?, uso.

Es un tipo de secuencia inmutable, por lo que una vez que se crea es imposible modificar, se caracteriza por la forma como se crea, debe usarse paréntesis y usar comas como separación de valores.

Las tuplas pueden usar diversas funciones e instrucciones asociadas, para poder manipular sus valores internos, lo que no se puede hacer es modificarlos.

- La función len() acepta tuplas, y regresa el número de elementos contenidos dentro.
- El operador + puede unir tuplas (ya se ha mostrado esto antes).
- El operador * puede multiplicar las tuplas, así como las listas.
- Los operadores in y not in funcionan de la misma manera que en las listas.

Observar la siguiente secuencia de operaciones que se ejecutan sobre la tupla inicial LTup:

```
LTup=(12,"Hola",4,6.3,"A")
print(LTup)
Lt1=LTup+(999,"Zegel")
print(Lt1)
Lt2=LTup*3
print(Lt2)
if(4 in LTup):
    print("Si se encuentra")
else:
    print("NO se encuentra")
```

Al ejecutar se muestra lo siguiente:

```
(12, 'Hola', 4, 6.3, 'A')
(12, 'Hola', 4, 6.3, 'A', 999, 'Zegel')
(12, 'Hola', 4, 6.3, 'A', 12, 'Hola', 4, 6.3, 'A', 12, 'Hola', 4, 6.3, 'A')
Si se encuentra
```

Veamos ahora este otro grupo de instrucciones:

```
Tbebidas = 'agua', 'café', 'batido'
print(Tbebidas)
a, b, c = Tbebidas
print(a)
print(b," ",c)
print(Tbebidas[1])
print()
for i in range(len(Tbebidas)):
    print(Tbebidas[i])
print()
for valor in Tbebidas:
    print(valor)
```

La carga de valores es de otra manera, se está observando diversas formas de acceso a los valores, muchos similares a los trabajados en las listas, por medio de un índice.

Al ejecutar se muestra lo siguiente:

```
('agua', 'café', 'batido')
agua
café   batido
café

agua
café
batido

agua
café
batido
```

c. ¿Qué es un diccionario?, uso y manejo del método keys().

El diccionario es otro tipo de estructura de datos de Python. No es una secuencia, pero puede adaptarse fácilmente a un procesamiento secuencial, además es mutable.

El diccionario tiene una característica y es que cada valor que lo compone debe tener un Identificador, los identificadores de cada valor no pueden repetirse.

Observaciones respecto a los diccionarios:

- Cada clave debe de ser única.
- Una clave puede ser un tipo de dato de cualquier tipo de datos.

- Un diccionario no es una lista, un diccionario almacena pares de valores esa es la diferencia.
- La función len() aplica también para los diccionarios, regresa la cantidad de pares (clave-valor) en el diccionario.

Observar cómo se carga y se accede a un diccionario:

```
dic={'nombre':'Mirtha','edad':20,'cursos':['Python','Java','VBasic']}
print(dic['nombre'])
print(dic['edad'])
print(dic['cursos'])
```

Debe considerar que se usan llaves, y antes de cada valor debe indicar la clave a continuación y el valor asignado.

```
Mirtha
20
['Python', 'Java', 'VBasic']
```

Para acceder a los valores que están dentro de la lista, debe realizar lo siguiente:

```
print(dic['cursos'][0])
print(dic['cursos'][1])
print(dic['cursos'][2])
```

Al ejecutar tenemos:

```
Python
Java
VBasic
```

Para acceder a los valores por medio de la sentencia for, podemos usar el método keys(), observemos los dos ejemplos mostrados, ambos finalmente recorren el diccionario diseñado.

```
dic={'nombre':'Mirtha','edad':20,'cursos':['Python','Java','VBasic']}

for key in dic.keys():
  print(key, ":", dic[key])

print()

for key in dic:
  print(key, ":", dic[key])
```

Puede resultar extraño, porque sin usar el método Keys funciona de igual manera:

```
nombre : Mirtha
edad : 20
cursos : ['Python', 'Java', 'VBasic']
```

```
nombre : Mirtha
edad : 20
cursos : ['Python', 'Java', 'VBasic']
```

d. Manejo de los métodos item() y values().

Usando el método ítem(), podemos acceder a los índices y valores al mismo tiempo, observar:

```
dic={'nombre':'Mirtha','edad':20,'cursos':['Python','Java','VBasic']}

for Id, Valor in dic.items():
    print(Id, "->", Valor)
```

Al ejecutar:

```
nombre -> Mirtha
edad -> 20
cursos -> ['Python', 'Java', 'VBasic']
```

Usando el método values(), obtenemos los valores del diccionario de manera directa, observar:

```
dic={'nombre':'Mirtha','edad':20,'cursos':['Python','Java','VBasic']}

for valor in dic.values():
    print(valor)
```

Al ejecutar:

```
Mirtha
20
['Python', 'Java', 'VBasic']
```

e. Operaciones básicas: modificar, agregar y eliminar valores

Observar la siguiente secuencia de operaciones sobre la tupla Tpl y veamos que ocurre:

```
Tpl={1:"Ron Pomalca",2:"Vino Casillero",3:"Pisco Vargas"}
print(Tpl)

Tpl[2]="Wisky Jaguer"
print(Tpl)

Tpl[4]="Racumin XL"
print(Tpl)

del Tpl[3]
print(Tpl)

Tpl.popitem()
print(Tpl)
```

Al ejecutar:

```
{1: 'Ron Pomalca', 2: 'Vino Casillero', 3: 'Pisco Vargas'}
{1: 'Ron Pomalca', 2: 'Wisky Jaguer', 3: 'Pisco Vargas'}
{1: 'Ron Pomalca', 2: 'Wisky Jaguer', 3: 'Pisco Vargas', 4: 'Racumin XL'}
{1: 'Ron Pomalca', 2: 'Wisky Jaguer', 4: 'Racumin XL'}
{1: 'Ron Pomalca', 2: 'Wisky Jaguer'}
```

De esta manera están observando diversas operaciones que puede realizar con los diccionarios, ahora en la parte práctica veremos su aplicación.

3. Desarrollo de ejercicios

Los ejercicios que veremos a continuación son clásicos, están desarrollados en un entorno de funciones para su mejor observación.

1. Problema Prg1

Se tiene una tupla con 20 números que se encuentren entre 10 y 30, desarrolle un algoritmo que solicite ingresar un número, luego indique cuantas veces se repite dentro de la tupla.

Solución:

Se debe considerar que para poder ingresar datos desde cero a un tupla no existe, por lo que vamos a usar dos métodos para poder trabajar con tuplas y usar a la vez los métodos de las listas, estos son tuple() y list().

```
## Zona de Funciones
def cargatupla():
    import random
    List=[]
    for i in range(20):
        List.append(random.randint(10,30))
    return tuple(List)

def buscanum(N,Tp):
    Lst=list(Tp)
    ct=0
    for valor in Lst:
        if (N==valor):
            ct=ct+1
    return ct

## Programa MAIN
Tupnum=cargatupla()
print(Tupnum)
print("Ingrese el número a buscar:")
Num=int(input())
print ("Hay ",buscanum(Num,Tupnum)," repeticion/es")
```

Al ejecutar se muestra lo siguiente:

```
(20, 20, 30, 21, 24, 19, 20, 18, 16, 25, 19, 26, 15,
25, 22, 28, 28, 23, 23, 26)
Ingrese el número a buscar:
20
Hay  3  repeticion/es
```

2. Problema Prg2

Desarrolle un algoritmo que permita simular el ingreso a un sistema, para ello debe digitar su usuario y contraseña, solo podrá tener un número de tres intentos para lograr acceder, en caso de equivocarse en la 3era vez el sistema se cierra. Usar diccionarios para la solución.

Solución:

```
## Zona de Funciones
def cargadic():
    Dic={"JM30":"JMontes","ES45":"ESanchez","MB26":"MBarrios"}
    return Dic

## Programa MAIN
Users=cargadic()
Sw=True
c=0
while(Sw or c<3):
    print("Ingrese el usuario:")
    usr=input()
    print("Ingrese la contraseña:")
    pwd=input()
    Op=True
    for idx,valor in Users.items():
        if(idx==pwd)and(valor==usr):
            Op=False
            break
    if Op:
        c=c+1
        print("Error de usuario/contraseña")
        if(c==3):
            break
    else:
        break
if(c==3):
    print("Se terminaron los intentos...!!!")
else:
    print("Bienvenido al Sistema...!!!")
```

Al ejecutar el programa, se muestra lo siguiente:

```
Ingrese el usuario:
Epinto
Ingrese la contraseña:
ep45
Error de usuario/contraseña
Ingrese el usuario:
ESanchez
Ingrese la contraseña:
ES45
Bienvenido al Sistema...!!!
```

Si se comenten 3 errores se muestra de la siguiente manera:

```
Ingrese el usuario:
ETrelles
Ingrese la contraseña:
Et75
Error de usuario/contraseña
Ingrese el usuario:
ETrellez
Ingrese la contraseña:
Er45
Error de usuario/contraseña
Ingrese el usuario:
ETrelles
Ingrese la contraseña:
er34
Error de usuario/contraseña
Se terminaron los intentos...!!!
```

3. Problema Prg3

A continuación, veremos un código que muestra la programación de un alumno durante la semana, observe el código entregado y trate de generar el enunciado a lo mostrado.

Solución:

```
Dicdia={1:"Domingo",2:"Lunes",3:"Martes",4:"Miercoles",
        5:"Jueves",6:"Viernes",7:"Sabado"}
print(Dicdia[2])
print(Dicdia[4])
dia=int(input("Ingrese el dìa en nùmeros:"))
print(Dicdia[dia])

for id in Dicdia:
    print(id,":",Dicdia[id])
DicCur={}
DicCur["Lunes"]=["Ofimàtica","Comunicaciones"]
DicCur["Martes"]=["FProgramaciòn"]
DicCur["Miercoles"]=["FProgramaciòn","EFìsica"]
DicCur["Jueves"]=[]
DicCur["Viernes"]=["Redes"]
print(DicCur)
print("")
print(DicCur["Martes"])
```

```
dia=int(input("Ingrese el dìa en nùmeros de su clase:"))
print(Dicdia[dia])
print(DicCur[Dicdia[dia]])

DicVar={"C03":"Curso de Python",(1,2,3):True,1:5000}
print(DicVar["C03"])

val=int(input("Ingrese el nùmero:"))
if(val in (1,2,3)):
    print(DicVar[(1,2,3)])
else:
    print("No existe...")
```

Al ejecutar el programa, se muestra lo siguiente:

```
Lunes
Miercoles
Ingrese el dìa en nùmeros:4
Miercoles
1 : Domingo
2 : Lunes
3 : Martes
4 : Miercoles
5 : Jueves
6 : Viernes
7 : Sabado
{'Lunes': ['Ofimàtica', 'Comunicaciones'], 'Martes':
['FProgramaciòn'], 'Miercoles': ['FProgramaciòn', 'E
Fìsica'], 'Jueves': [], 'Viernes': ['Redes']}

['FProgramaciòn']
Ingrese el dìa en nùmeros de su clase:6
Viernes
['Redes']
Curso de Python
Ingrese el nùmero:3
True
```

4. Actividad

A) Ejercicios en general

a) Desarrollar un algoritmo que cargue en un diccionario los usuarios y contraseñas del área donde trabajan los empleados (similar a los desarrollado en los ejercicios), luego deberá indicar si existe un error en el usuario o en la contraseña ingresadas (usar mensajes), solo se permiten 3 intentos. (debe modificar el código desarrollado)

b) Ingresar 100 números en una tupla, luego deberá ubicar el mayor y menor valor de los números ingresados, deberá crear funciones para mostrar dichos valores.

c) Desarrolle un diccionario que permita capturar los signos zodiacales, asignarle a cada uno un índice, luego deberá ingresar una fecha y verificar a que signo le pertenece.

d) Crear una tupla con los meses del año, el programa deberá pedir números, si el numero esta entre 1 y la longitud máxima de la tupla, muestra el contenido de esa posición sino muestra un mensaje de error.

   El programa termina cuando el usuario introduce un cero.

5. Fuentes consultadas:

A) https://edube.org/learn/programming-essentials-in-python-part-1-spanish
B) Edison Zavaleta C. (2005). *Fundamentos de Programación*. Perú, Editorial Abaco-Lima.