FACULTAD
DE
CIENCIAS EXACTAS Y
NATURALES

Segunda relatoría.
Informática III.

UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MANIZALES

Relatoría segundo corte

Pinzón Solano Brigida Janeth

Fecha de presentación *17/09/2022*

Ciclo For

Clase 09/09/2022

Se definió el ciclo For, y se complemento con ejercicios, reforzando los conocimientos de lógica y desarrollando la manera de indexar

El ciclo for se definió como una variable creada dentro del mismo ciclo que recorriera una determinada condición, mientras sucede, se realizan condiciones del mismo, los ciclos se pueden anidar entre sí para crear condiciones específicas.

Teniendo la siguiente estructura:

for <variable> in <iterable>:
  <sentencia1>
  <sentencia2>

Definiendo para cada concepto:

variable: Para este ítem, se define como una variable que determina el for, no debe ser definido anteriormente, si no que se nombra en el momento de crear el ciclo.

iterable: Es el elemento que se va a recorrer o iterar, dentro de las posibilidades se puede encontrar los siguientes ejemplos:

Lista, tupla,rangos, rangos y cadenas para la siguiente imagen.

```
#----Ciclo for para listas----------
print("Las alturas son: " )
Alturas=[10,20,60,1,80]
for  altura in Alturas:
  print(altura, end=" ")
print(" ")
#----Ciclo for para tuplas----------
pesos=(70,60,55,62,45)
print("Los pesos son: ")
for peso in pesos:
  print(peso,end=" ")
print(" ")
#----Ciclo for para rangos----------
secuencia=range(1,11,1)
print("La secuencia es")
for secu in secuencia:
  print(secu,end=" ")
print(" ")
#----Ciclo for para strings----------
mensaje='Hola mundo cruel'
print("Cada letra en el mensaje es: ")
for letra in mensaje:
  print(letra,end=" ")
```

Figura No 1. Ejemplos de ciclo for.

sentencia: Para este ítem se debe ingresar la acción que quiere ser ejecutada respecto a cada iteración realizada, se pueden anidar diferentes ciclos, y es a libertad del programador, como por ejemplo:

```
#Determine si un numero n es primo
numero=98
contador=0
for division in range(2,numero):
  if(numero%division)==0:
    contador=contador+1
if contador==0:
  print('Es primo')
else:
  print('No es primo')
```

Figura No 2. Ejemplos de ciclo for anidado.

Donde la sección en rojo denota la acción que se debe tomar para cada iteración

Iteraciones y métodos

Clase 14/09/2022

Se estudió la iteraciones en cadenas de caracteres, donde se caracterizó los métodos para ayudar a recibir y cambiar información de la misma, tales como upper, center, tittle, etc. Se realizaron diferentes ejercicios de indexación de cadenas con sus formas, tales como obtener la cadena al revés u obtener pedazos de la cadena.

La forma de realizar cada método de modificación tiene la siguiente estructura; donde se debe tener en cuenta que después del objeto a indexar, se debe agregar un punto y en seguida de la función de modificación así:

Por ejemplo para cadena de strings: cadena.funcion()

```
#Solo sirve con cadenas, y no alteran la cadena original
cadena1="Anita lava la tina"
#Vuelve todo mayusculas
cadena1.upper()
#Vuelve todo minuscula
cadena1.lower()
#Vuelve en formato Titulo
cadena1.title()
#Pregunta si la cadena es numerica donde retorna un booleano
cadena1.isalnum()
#Pregunta si la cadena es alfabetica donde retorna un booleano
cadena1.isalpha()
#Reemplaza un lugar de la cadema
cadena1.replace("Anita","Cristian")
#Centra la cadena
cadena1.center(10)
```

Figura No 3. Ejemplos de manejo de cadenas

Se debe tener en cuenta el tipo de respuesta que devuelve el código como por ejemplo un booleano, y también que es modificable solo en el momento de la última ejecución, sin embargo no altera la cadena original, se puede interpretar como una manera de obtener ciertos datos de manera instantánea.

Algunos ejemplos de las funciones más conocidas para trabajar cadenas son:

Formato de cadena: capitalize(), upper(), lower(), title(), center(spaces), strip().
Operaciones: count(subcadena), find(subcadena), replace(old, new).
Tipo de cadena: isalnum(), isalpha(), isdigit, isdecimal,
Método de indexación:[índice]
Slicing: [indice1:indice2:salto]

Slicing: El método de slicing sirve para obtener pedazos o "rebanadas" de cadenas, al igual que la indexación, este no altera la cadena original, sin embargo, si se desea guardar la misma, se puede almacenar en una variable especial para este.
Se tienen los siguientes ejemplos para dos cadenas:

```
cadena2="Mensaje para informatica 3"
cadena3="Holamundo"
#Sintaxis para extraer caracteres
cadena3[0:9:3]
cadena3[-3:-8:-1]
cadena3[-1]
 #Ejercicios extraer toda la cadena al reves
cadena2[-1:-27:-1]
#Extraer cada tercer elemento empezando en el indice
cadena2[2::3]
#Extraer los dos elementos en la mitad de la cadena
cadena2[12:14]
```

Figura No 4. Ejemplos de slicing en cadenas

Clase 30/09/2022
Para listas :
Operaciones: append(value), insert(index, value), pop(index), remove(value), count(value), sort(), reverse(), clear(), copy()
Indexado: [índice]
Slicing: [indice 1:indice 2:salto]
Donde se muestran los siguientes ejemplos con su respectiva explicación de funcionalidad.

```python
lista1 = [1,2,3,4,5,6,7,8]
#Creando la lista 2
lista2 = list(range(20)) + ["a", "b", "c", 100]
#Creando la lista 2
lista3 = ["cruel", "mundo", "hola", 1, 100, 200, 500]

# Para Agregar al final de lista1,
lista1.append(1)
print(lista1)

#Para Agregar al comienzo de lista1
lista1.insert(0, 4)
print(lista1)

#Para eliminar los 3 ultimos elementos de lista1
#-----------------------------------------------
#elimina el ultimo elemento no es necesario colocar el indice
#elimina cada elemento, con el metodo .pop()
lista1.pop(-1)
#Este metodo si modifica la lista
print(lista1)

#Sume todos los elementos de lista 2 que pueden operarse algebraicamente
#Con el metodo remove, se quitan los elementos deseados.
lista2.remove("a")
resultado = sum(lista2)
print(lista2, resultado)

#La funcion sort organiza la lista de forma ascendente o descendente
print("*Organice lista2 de forma ascendente")
#para hacerlo descendente hay que utilizar el parametro reverse
lista2.sort(reverse = True)
print(lista2)
```

Figura No 5. Ejemplos de modificaciones en listas.

Clase 05/10/2022

Los diccionarios son un tipo de estructura que guarda un par-valor, donde se puede acceder al mismo de diferentes maneras, estos se denotan con la variable llamada, y la información dentro del mismo se encierra en corchetes así:

variable={valor1:valor2}

Las diferentes maneras más comunes de trabajar un diccionario son:

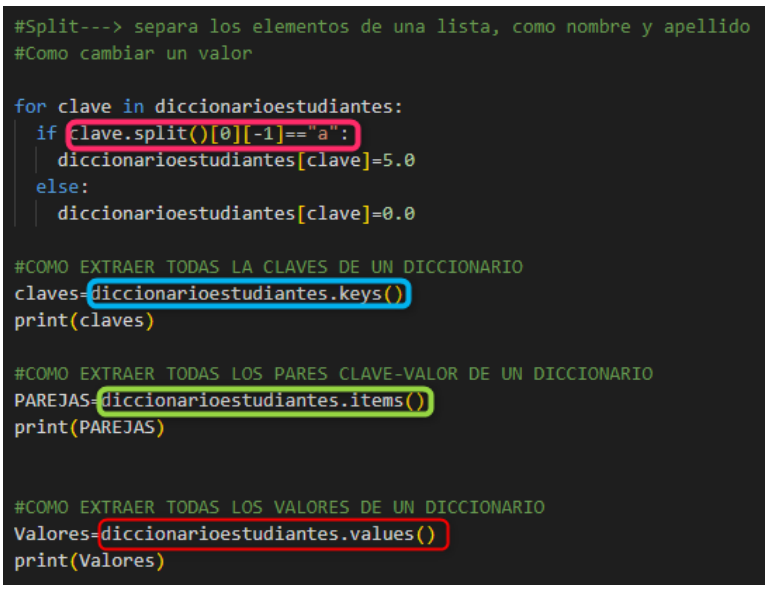
Extraccion: items(), keys(), values(), get(key)

Eliminar: pop(key)

Almacenamiento: clear(), copy()

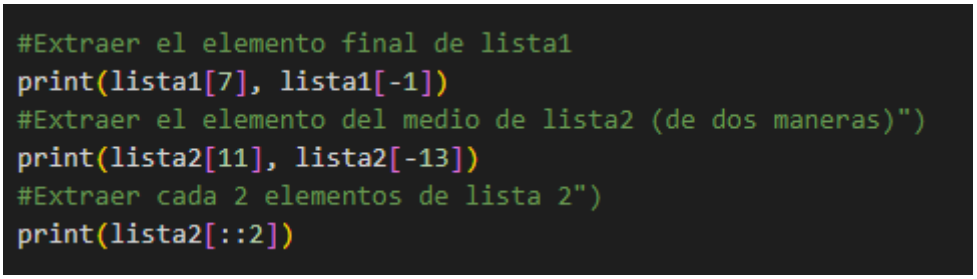
Indexado: [key]

Donde se muestran los siguientes ejemplos con su respectiva explicación de funcionalidad.



```
#Split---> separa los elementos de una lista, como nombre y apellido
#Como cambiar un valor

for clave in diccionarioestudiantes:
  if clave.split()[0][-1]=="a":
    diccionarioestudiantes[clave]=5.0
  else:
    diccionarioestudiantes[clave]=0.0

#COMO EXTRAER TODAS LA CLAVES DE UN DICCIONARIO
claves=diccionarioestudiantes.keys()
print(claves)

#COMO EXTRAER TODAS LOS PARES CLAVE-VALOR DE UN DICCIONARIO
PAREJAS=diccionarioestudiantes.items()
print(PAREJAS)


#COMO EXTRAER TODAS LOS VALORES DE UN DICCIONARIO
Valores=diccionarioestudiantes.values()
print(Valores)
```

Figura No 6. Ejemplos de manejo de diccionarios.



```
#Extraer el elemento final de lista1
print(lista1[7], lista1[-1])
#Extraer el elemento del medio de lista2 (de dos maneras)")
print(lista2[11], lista2[-13])
#Extraer cada 2 elementos de lista 2")
print(lista2[::2])
```

Figura No 7. Slicing en diccionarios.

Para tuplas:
Operaciones: index(value), count(value)
Indexado: [indice]
Slicing: [indice1:indice2:salto]

Funciones

Clase 07/10/2022

Funciones:
Las funciones van a ser bloques de código que contienen una serie de declaraciones para cumplir con un fin específico
La función no se ejecuta si no es llamada, donde reciben parámetros de entrada y dichos parámetros pueden participar en las declaraciones posteriores,nos va a retornar una salida, una de las ventajas es que estas funciones permiten tener documentación y simplificar el código.
La estructura es:

```
def primera(a):
   print('¡Hola', a, '!')
primera('Janeth')
```

Donde:
`def:` Crea la función, este es fijo
`primera` : Se llama el nombre de su función, y es a conveniencia del programador
Un ejemplo básico es:

```python
def num_mayor(a,b):
    if a>b:
        mayor=a
    else:
        mayor=b
    return mayor
```

Figura No 8.Ejemplo de funciones.

Todos los ejemplos se encuentran en el repositorio:
[1]https://github.com/miTitansito/Informatica-III