Flujo de control

Condicionales if...else

Al realizar el tratamiento de los datos, existen ocasiones donde nosotros deseamos realizar una operación o cálculo solo cuando se satisfacen ciertas condiciones. En Python se usa `if...elif...else` para establecer flujos de control usando operadores lógicos. Ejemplo:

```
vr = 5
if vr >= 5:
        print "La variable vr es igual o mayor a 5"
```

En el ejemplo anterior podemos identificar que utilizamos una estructura condicional donde vamos a establecer una condición, si ésta condición es verdadera se realizará una operación o tarea; si la condición no se cumple, salimos de la estructura condicional.

Flujo de control

Condicionales if...else

En el ejemplo la variable `vr` tiene un valor de 5, si la variable es mayor o igual que 5, Python deberá mostrarnos un mensaje, si la condición no se cumple, no se visualizará ningún mensaje.

Debemos identificar que al usar las estructuras condicionales, siempre se establecerá una condición o expresión lógica. En algunos casos se requiere que al no cumplirse una condición inicial, deseamos que se realice una tarea en particular, a pesar de este hecho. En estos casos usamos la palabra `else` en nuestra estructura condicional. Analice el siguiente ejemplo (nótese la sangría en las líneas de programación).

Flujo de control

Condicionales if...else

```
num = -10
if num > 0:
        print "La variable num contiene un numero positivo"
else:
        print "La variable num contiene un numero negativo"
```

Del ejemplo anterior podemos observar que se establece una condición donde se pregunta si el valor de la variable `num` es positivo. Si la condición se cumple, se despliega un mensaje expresando que la variable tiene un número positivo; por otra parte, si la condición no se cumple ENTONCES se llevará a cabo otra tarea, en este caso se visualizará un mensaje donde se notifica que la variable posee un número negativo.

Flujo de control

Condicionales if...else

Lleva a cabo el siguiente ejercicio:

```
num = 10
if num > 0:
        print "La variable num contiene un numero positivo"
elif num == 0:
    print "La variable tiene un valor de 0"
else:
        print "La variable num contiene un numero negativo"
```

Qué ocurre, si cambiamos el valor de la variable num a cero?

Ciclo while

Operadores y expresiones lógicas

Los ciclos `while` se usan para repetir varias veces líneas de nuestros programas. Mediante una expresión lógica se controla el número de veces en las que se efectuará el ciclo. Ejemplo:

```
c = 6
c == 40     # La variable c es igual a 40?
c != 40     # La variable c es DIFERENTE de 40?
c >= 5      # La variable es MAYOR o IGUAL que 5?
c <= 5      # La variable es MENOR o IGUAL que 5?
c > 3       # La variable es MAYOR que 3?
c < 3       # La variable es MENOR que 3?
```

Ciclo while

Operadores y expresiones lógicas

Cuando hacemos el análisis de nuestros datos, en algunas ocasiones es necesario filtrar la información por medio de expresiones lógicas, donde se evalua si éstas son verdaderas o falsas y a partir de ello, continuar con el análisis de los datos (cicloWhile.py). Ejemplo:

```
ti = 2000; dt = 1; tf = 2017
while ti <= tf:
        print('Year: %4d')% ti
        ti = ti+dt
print('Ultimo valor de ti = %4d', % ti)
print('La expresion ti <= tf es:',(ti<=tf) )
```

Ciclo for

Al procesar nuestros datos, puede presentarse el caso donde sea necesario analizar cada uno de los datos que almacenamos en una lista o arreglo.

El ciclo (o búcle) for lo usamos para iterar sobre una secuencia (lista, tuple o string). Su sintaxis es:

```
for var in secuencia:
        bloque de instrucciones
```

donde `var` es la variable que toma el valor del ítem en la `secuencia` en cada iteración.

Ciclo for

Ejemplos

- Visualización de elementos de una lista:

```
a = [ 23, 45, 67, 78]
for c in a:
        print c
```

- Cálculo del factorial de 7:

```
fac = 1
for c in (1,2,3,4,5,6,7):
        fac = fac*c
print fac
```

Ciclo for

función range

La función `range` la usamos para generar una secuencia de números.
La función `range(n)` genera enteros desde el 0,1,2,...,n-1
Si usamos `range(inicio,final,incremento)` se obtiene una secuencia que inicia con el valor inicio, el siguiente valor es inicio+incremento y de esta manera continua la secuencia hasta alcanzar el valor final.
PERO NO SE INCLUYE EL VALOR FINAL. Ejemplos:

```
range(5)
range(2,5,3)
range(1,11,2)
```

Ciclo for

Creando listas usando el ciclo for

El generar una lista de manera manual suele ser una tarea tediosa si existen muchos valores secuenciales. Usando el ciclo `for` y la función append podemos generar una lista de manera sencilla y rápida. Ejemplo:

```
T = [-20,-15,-10,-5,0,5,10,15,20] # Generando lista de manera
    manual
TF = []   # Generando una lista usando for y range
for x in range(-20,21,5)
        TF.append(x)
```