Universidad del Rosario | Educación Continua

INTRODUCCIÓN A LA PROGRAMACIÓN EN PYTHON

- Miguel Orjuela

Personas con *propósito* que ayudan a transformar vidas

Universidad del Rosario | Educación Continua

Introducción a la programación en Python

2019

Sesión # 2

Control de flujo

Miguel Angel Orjuela Rocha

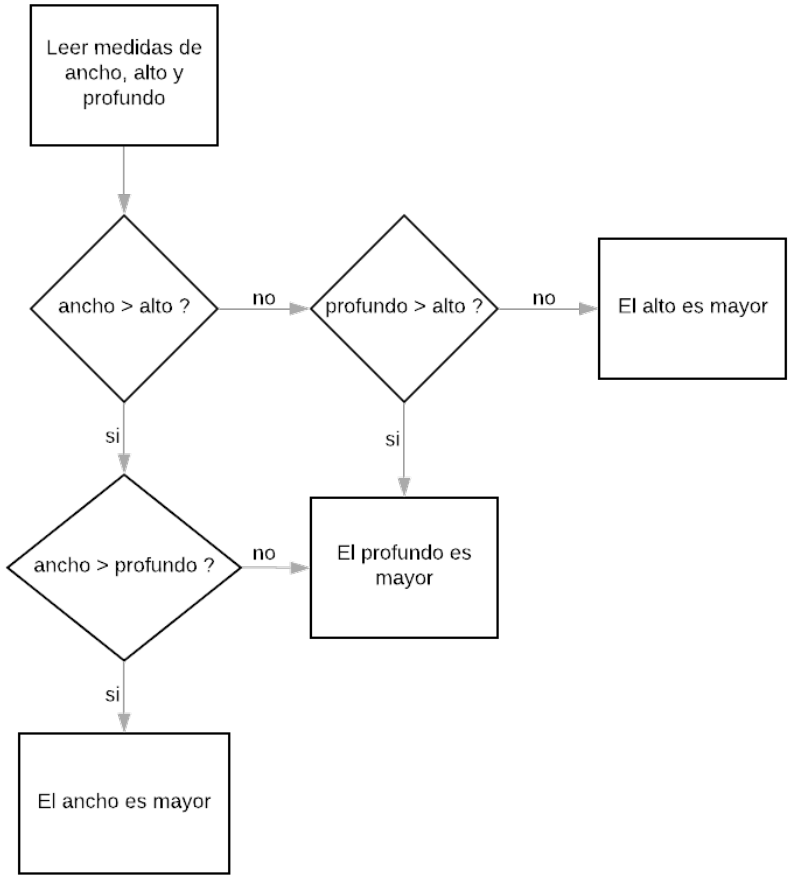
Ingeniero de Sistemas y Computación

Contenido

- if, elif, y else
- Ciclos for
- Ciclos while
- pass
- range
- Expresiones ternarias

Control de flujo

Python tiene varias palabras clave para implementar la lógica condicional, ciclos, y otros conceptos de control de flujo que vemos en otros lenguajes de programación



if, elif, y else

if, elif, y else

La declaración if revisa una condición

Si es cierta, ejecuta el código en el bloque que sigue

```
x = 3;

if x < 0:
    print('x es negativo')
```

if, elif, y else

Una declaración if puede tener uno o más bloques elif

```
if x < 0:
    print('x es negativo')
elif x == 0:
    print('x es cero’)
elif 0 < x < 5:
    print('x es positivo menor que 5')
else:
    print('x es posivito mayor que 5')
```

Si alguna expresión es cierta, las siguientes elif no se ejecutan

También puede tener un bloque else por si todas las condiciones evalúan a falso

if, elif, y else

Si la condición es compuesta (and/or), las condiciones se evalúan de izquierda a derecha y pueden hacer cortocircuito

```
a = 5; b = 7;
c = 8; d = 4;

if a < b or c > d:
    print('Se cumple la condición')
```

c > d nunca se evalúa

Se pueden encadenar comparaciones

```
4 > 3 > 2 > 1
```

Ciclos for

Ciclos for

Los ciclos for permiten recorrer una colección (como una lista o tupla)

La síntaxis es:

```
for valor in coleccion:
    # hacer algo con valor
```

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

for valor in coleccion:
    print('Imprimendo el valor {0:d}'.format(valor))
```

Ciclos for

Se puede avanzar a la siguiente iteración con la palabra continue

```
coleccion = [1,2,None,3,None,4]
total = 0
for valor in coleccion:
    if valor is None:
        continue
    total += valor
print('El total es {0:d}'.format(total))
```

Ciclos for

Un ciclo for puede ser terminando con la palabra break

```
coleccion = [1,2,9,3,8,6,9,4]
total_hasta_8 = 0
for valor in coleccion:
    if valor == 8:
        break
    total_hasta_8 += valor
print('El total hasta el valor 8 es {0:d}'.format(total_hasta_8))
```

El break solo rompe el ciclo del for en el que está adentro, for exteriores siguen corriendo

```
for i in range(4):
    for j in range(4):
        if j > i:
            break
        print((i,j))
```

Ciclos while

El ciclo while se ejecuta hasta que la condición del ciclo evalúa a falso

```
x = 256
total = 0
while x > 0:
    if total > 500:
        break
    total += x
    x = x//2
print(total)
```

Ciclos for

pass indica que no se hace una operación

```
if x > 0:
    print('Negativo')
elif x == 0:
    # Poner que son iguales
    pass
else:
    print('Positivo')
```

Ciclos for

La función range retorna un iterador que tiene un secuencia de enteros igualmente espaciados

```
range(10)
range(0,10)
list(range(0,10))
list(range(0,20,2))
list(range(5,0,-1))
```

Prácticos para recorrer secuencias

```
len(seq)
```

```
seq = [1,2,3,4]
for i in range(len(seq)):
    print(seq[i]*2)
```

Ciclos for

Ejemplo: sumar los múltiplos de 3 y los de 5 desde 0 hasta 10000

```
suma = 0
for i in range(10000):
    # % es el operador módulo
    if i % 3 == 0 or i % 5 == 0:
        suma += i
print(suma)
```

Expresiones ternarias

Expresiones ternarias

Combina un bloque if-else que produce un valor en una expresión

```
if condicion:
    valor = expresion_cierta
else:
    valor = expresion_falsa
```

```
valor = expresion_cierta if condicion else expresion_falsa
```

Ejemplo:

```
'No negativo' if x >= 0 else 'Negativo'
```

Resumen

- Las condiciones se declaran con if, elif y else
- Si un if-else sencillo se puede escribir como expresión ternaria
- El ciclo while se ejecuta hasta que su condición evalúa a falso
- Los ciclos for permiten recorrer una colección
- La función range crea secuencias

A continuación

Estructuras de datos básicas