El futuro digital es de todos

MinTIC

Booleanos

Operadores

Universidad Tecnológica de Pereira

MisiónTIC 2022

Contenido

- Valores de verdad y proposiciones
- Álgebra Booleana
- Operaciones lógicas
- Tabla de verdad
  - Qué son las tablas de la verdad
- Leyes fundamentales
- Leyes de De Morgan

Contenido

- Ejecución de condiciones
  - Expresiones booleanas
- Operadores lógicos
- Ejecución de condicionales
- Ejecución alternativas
- Condicionales encadenados
- Condicionales anidadas

Contenido

- Try y Except
- Evaluación de las expresiones lógicas

Contenido

Valores de verdad y proposiciones

Una proposición es una expresión completa que tiene un valor de verdad, es decir que es verdadera o falsa.

Por ejemplo, la frase ‘Hoy está haciendo frio’ es una proposición porque expresa una idea que puede ser verdadera o falsa. Evidentemente nosotros no podemos saber si, en el momento en que el lector vea esto, estará haciendo frio o no, pero el hecho de que no lo sepamos no hace que la frase deje de ser una proposición. Por el contrario, expresiones como ‘amarillo’, ‘amanecer’ o ‘bailar’ no son frases completas, no se puede decir que sean verdaderas o falsas y por ende no pueden ser proposiciones. En español se suelen usar también los términos sentencia, afirmación o juicio para referirse a una proposición.

Contenido

Valores de verdad y proposiciones

Cuando se usan proposiciones dentro del contexto de lógica matemática, se espera que tengan un valor de verdad que, como dijimos, puede ser verdadero o falso.

Dependiendo del momento en que se evalúe, una proposición podría cambiar de valor (piense en la frase ‘Hoy está haciendo frio’), pero no es posible que una proposición tenga dos valores diferentes al mismo tiempo.

Contenido

Valores de verdad y proposiciones

Finalmente, debemos recalcar que los únicos dos valores de verdad son verdadero y falso, independientemente de cómo se representen.

Por ejemplo, es usual que estos valores se representen con expresiones como V y F, true y false, T y F o incluso números como 1 y 0. Sin embargo, en todos estos casos estamos hablando de los valores de verdad y no de expresiones de tipo numérico o de cadenas de caracteres.

Contenido

Álgebra Booleana

•El álgebra Booleana es la rama del álgebra que trabaja con proposiciones y no con valores numéricos (como el álgebra elemental).
•El álgebra Booleana se basa en los valores verdadero y falso y las operaciones ∧ (conjunción), ∨ (disyunción) y ¬ (negación).

Contenido

Operaciones lógicas

Son tres las operaciones básicas del álgebra Booleana. El resto de operaciones, como la **implicación** o la **equivalencia**, son operaciones secundarias que se pueden construir a partir de la conjunción, disyunción y negación.

Contenido

Operaciones lógicas

Conjunción. La conjunción (∧, and, y) es una operación Booleana binaria que tiene valor verdadero sólo cuando ambos operandos tienen valor verdadero. Esto implica que cuando un operando es verdadero y el otro es falso, o cuando ambos operandos son falsos, el resultado de una conjunción es un valor falso.

Contenido

Operaciones lógicas

Disyunción. La disyunción (V, or, o) es una operación Booleana binaria que tiene valor verdadero cuando por lo menos uno de los operandos tiene valor verdadero. Esto implica que una disyunción tiene valor falso sólo cuando ambos operandos tienen valor falso.

Contenido

Operaciones lógicas

Negación. La negación (¬, not, no) es la operación Booleana que toma un valor de verdad y lo convierte en el otro valor. Es decir que la negación de un valor verdadero es un valor falso, y la negación de un valor falso es un valor verdadero. La negación es una operación unaria, que se aplica sobre un solo operando.

contador > 0 and valor == 100
contador > 0 o valor == 100
contador > 0 not valor == 100

Ejemplos

```
Var = 1
print(var > 0)
print(not (var <= 0))

print(var != 0)
print(not (var == 0))
```

Contenido

Tabla de verdad

Las tablas de verdad o tabla de valores de verdad, es una tabla que muestra el valor de verdad de una proposición compuesta, para cada combinación de valores de verdad que se pueda asignar a sus componentes.

La tabla de los "valores de verdad", es usada en el ámbito de la lógica, para obtener la verdad (V) o falsedad (F), valores de verdad, de una expresión o de una proposición.

Las tablas de verdad son, por una parte, uno de los métodos más sencillos y conocidos de la lógica formal, pero al mismo tiempo también uno de los más poderosos y claros. Entender bien las tablas de verdad es, en gran medida, entender bien a la lógica formal misma.

Contenido

Qué son las tablas de la verdad

Fundamentalmente, una tabla de verdad es un dispositivo para demostrar ciertas propiedades lógicas y semánticas de enunciados del lenguaje natural o de fórmulas del lenguaje del cálculo proposicional:

- Si son tautológicas, contradictorias o contingentes
- Cuáles son sus condiciones de verdad
- Cuál es su rol inferencial, es decir, cuáles son sus conclusiones lógicas y de qué otras proposiciones se siguen lógicamente.

Contenido

Qué son las tablas de la verdad

Tabla And

Argumento A	Argumento B	A y B
False	False	False
False	True	False
True	False	False
True	True	True

Tabla Or

Argumento A	Argumento B	A or B
False	False	False
False	True	True
True	False	True
True	True	True

Tabla Not

Argumento	not Argumento
False	True
True	False

Contenido

Leyes fundamentales

- **Conmutatividad**: Tanto la conjunción como la disyunción son conmutativas, así que A ∨ B es equivalente a B ∨ A. También es cierto que A ∧ B es equivalente a B ∧ A.
- **Asociatividad**: Tanto la conjunción como la disyunción son asociativas, así que A ∨ (B ∨ C) es equivalente a (A ∨ B) ∨ C. Además, A ∧ (B ∧ C) es equivalente a (A ∧ B) ∧ C.
- **Distribución**: en el álgebra Booleana, la conjunción distribuye sobre la disyunción y viceversa.

Contenido

Leyes de De Morgan

La negación de una conjunción es la separación de las negaciones.
La negación de una disyunción es la conjunción de las negaciones.

```
not (p and q) == (not p) or (not q)
not (p or q) == (not p) and (not q)
```

Contenido

Ejecución de condicionales

Expresiones booleanas

Una expresión booleana es una expresión que es verdadera o falsa.

En el siguiente ejemplo se utiliza el operador ==, que compara dos operandos y produce True si son iguales o falsos si son diferentes.

Contenido

Ejecución de condicionales

```
var1 = 5
var2 = 5
print(var1 > var2)
print(var1 < var2)
print(var1 != var2)
print(var1 == var2)
print(type(True))
print(type(False))
```

```
x != y # x no es igual a y
x > y # x es mayor que y
x < y # x es menos que y
x >= y # x es mayor o igual que y
x <= y # x es menor o igual que y
x is y # x es lo mismo que y
x is not y # x no es lo mismo que y
```

Contenido

Ejecución de condicionales

Aunque estas operaciones probablemente le resulten familiares, los símbolos de Python son diferentes de los símbolos matemáticos para las mismas operaciones.

Un error común es utilizar un solo signo igual (=) en lugar de un doble signo igual (==).

Recuerde que = es un operador de asignación y == es un operador de comparación.

Contenido

Operadores lógicos

Hay tres operadores lógicos: y (and), o (or), y no (not). La semántica (significado) de estos operadores es similar a su significado en inglés. Por ejemplo,

x > 0 and x < 10

Contenido

Operadores lógicos

Hay tres operadores lógicos: y (and), o (or), y no (not). La semántica (significado) de estos operadores es similar a su significado en inglés. Por ejemplo,

x > 0 and x < 10, n%2 == 0 or n%3 == 0

Contenido

Ejecución de condicionales

Para escribir programas útiles, casi siempre necesitamos la capacidad de comprobar las condiciones y cambiar el comportamiento del programa en consecuencia.

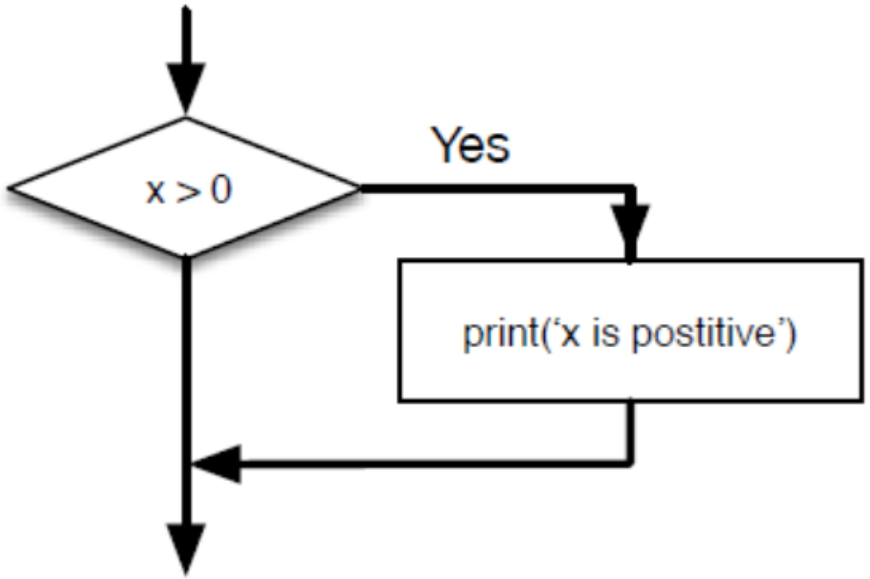
Las **declaraciones condicionales** nos dan esta habilidad. La forma más simple es la declaración de "si" o "if":

Contenido

Ejecución de condicionales

```
x = 10

if x > 0:
    print('x es positivo')
```



Contenido

Ejecución alternativas

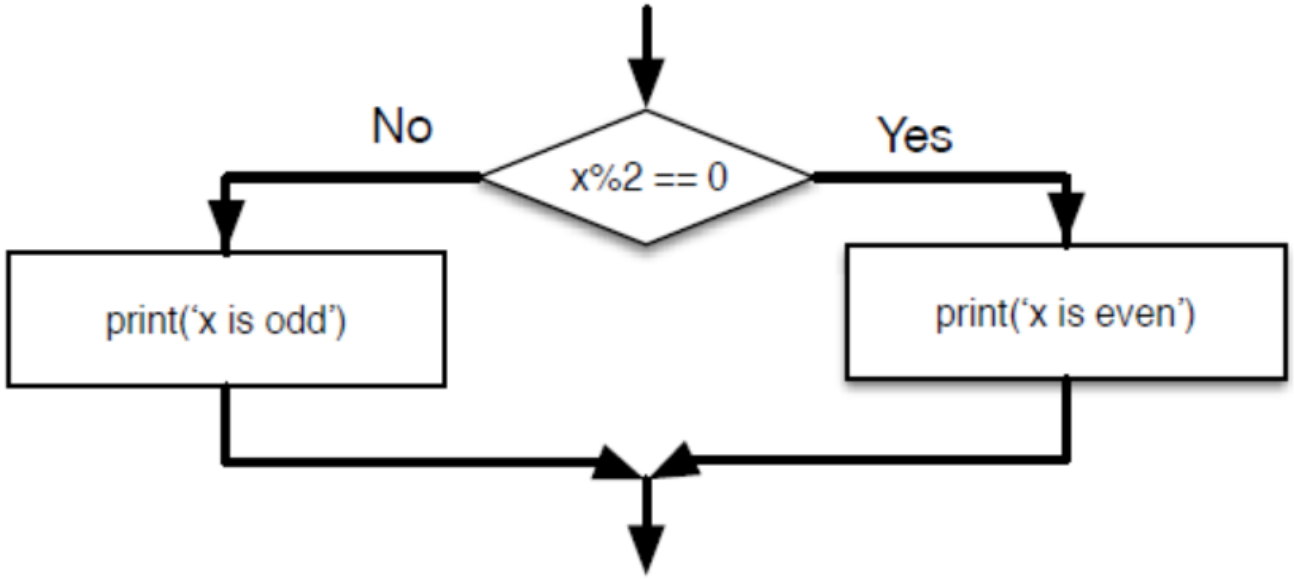
Una segunda forma de la declaración de if es una ejecución alternativa, en la que hay dos posibilidades y la condición determina cuál de ellas se ejecuta. La sintaxis tiene este aspecto:

```
x = 10
if x % 2 == 0:
    print('x es par')
else:
    print('x es inpar')
```

Si el residuo cuando x se divide por 2 es 0, entonces sabemos que x es par, y el programa muestra un mensaje a tal efecto. Si la condición es falsa, se ejecuta el segundo conjunto de declaraciones.

Contenido

Ejecución alternativas



Contenido

Condicionales encadenados

```
x = 10
y = 20
if x < y:
    print('"x" es menor que "y"')
elif x > y:
    print('"x" es mayor que "y"')
else:
    print('"x" y "y" son iguales')
```

Contenido

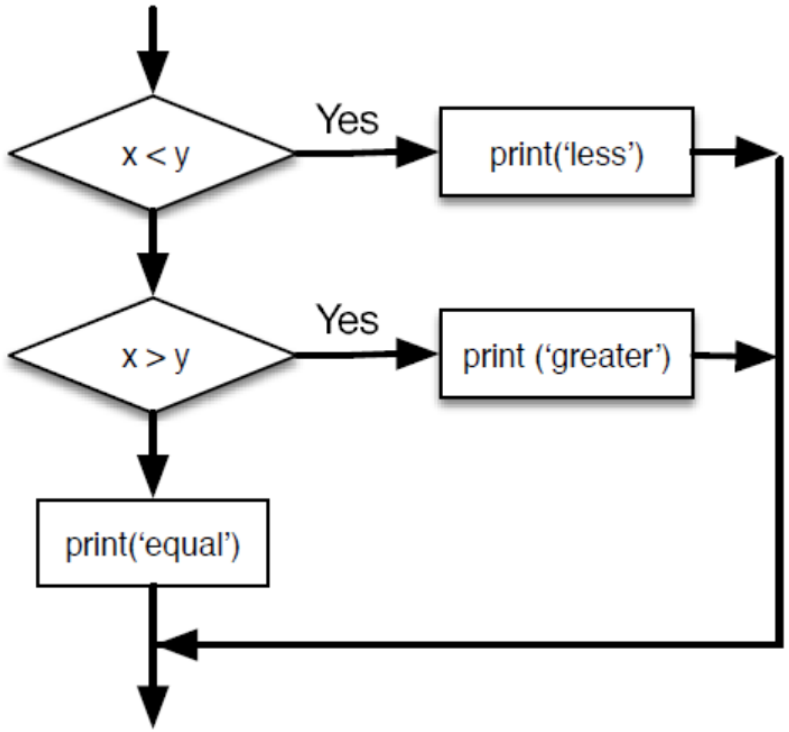
Condicionales encadenados

elif es una abreviatura de "else if".

De nuevo, se ejecutará exactamente una rama. No hay límite en el número de declaraciones de elif. Si hay una cláusula "else if", tiene que ser al final, pero no tiene que haber una.

Contenido

Condicionales encadenados



Contenido

Condicionales encadenados

```
letra = 'a'
if letra == 'a':
    print('Mal resultado')
elif letra == 'b':
    print('Buen resultado ')
elif letra == 'c':
    print('Cerca, pero no es correcto')
```

Contenido

Condicionales anidadas

Un condicional también puede anidarse dentro de otro. Podríamos haber escrito el ejemplo de las tres ramas así:

```
x = 9
y = 32
if x == y:
    print('x e y son iguales')
else:
    if x < y:
        print('x es menos que y')
    else:
        print('x es mayor que y')
```

Contenido

Condicionales anidadas

El condicional exterior contiene dos ramas.

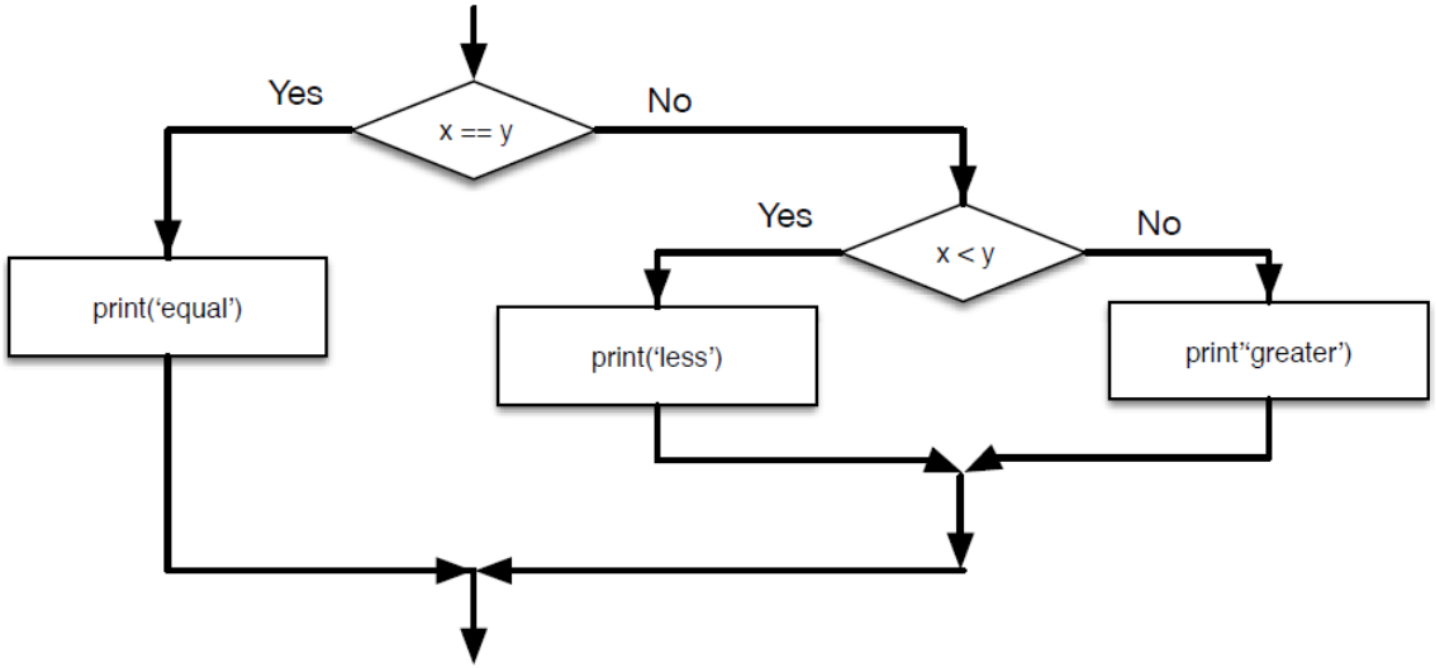
- La primera rama contiene una declaración simple.
- La segunda rama contiene otra declaración if, que tiene dos ramas propias.

Esas dos ramas son ambas declaraciones simples, aunque también podrían haber sido declaraciones condicionales.

Aunque la **indentación** de las declaraciones hace que la estructura sea aparente, los condicionales anidados se vuelven difíciles de leer muy rápidamente. En general, es una buena idea evitarlos cuando se pueda.

Contenido

Condicionales anidadas



Contenido

Condicionales anidadas

Los operadores lógicos suelen ser una forma de simplificar las declaraciones condicionales anidadas. Por ejemplo, podemos reescribir el siguiente código utilizando un solo condicional:

```
if 0 < x:
    if x < 10:
        print('x es un número positivo de un solo dígito.')
```

```
if 0 < x and x < 10:
    print('es un número positivo de un solo dígito')
```

Contenido

Try y Except

La declaración print se ejecuta sólo si pasamos los dos condicionales, para que podamos obtener el mismo efecto con el operador y:

```
"""
Para explicar Try y Except utilizaremos de ejemplo una función que
convierte la temperatura en grados Fahrenheit a una temperatura en
grados Celsius:
"""
temperatura_fahr = input('Ingrese la temperatura en grados Fahrenheit
: ')
fahr = float(temperatura_fahr)
cel = (fahr - 32.0) * 5.0 / 9.0
print(cel)
```

Contenido

Try y Except

Hay una estructura de ejecución condicional incorporada en Python para manejar estos tipos de errores esperados e inesperados llamada "try / except". La idea de try y except es que se sabe que alguna secuencia de instrucción(es) puede tener un problema y se quiere añadir algunas declaraciones para ser ejecutadas si se produce un error. Estas declaraciones adicionales (el bloque except) se ignoran si no hay ningún error. Puedes pensar en la característica try y except de Python como una "póliza de seguro" en una secuencia de declaraciones.

Contenido

Try y Except

Podemos reescribir nuestro convertidor de temperatura de la siguiente manera:

```
temperatura_fahr = input('Introduzca una temperatura en Fahrenheit:')
try:
    fahr = float(temperatura_fahr)
    cel = (fahr - 32.0) * 5.0 / 9.0
    print(cel)
except:
    print('No ingreso ningún número, gracias')
```

Contenido

Evaluación de las expresiones lógicas

Cuando Python está procesando una expresión lógica como x >= 2 and (x/y) > 2, se evalúa la expresión de izquierda a derecha. Debido a la definición de y, si x es menos de 2, la expresión x >= 2 es Falsa y por lo tanto toda la expresión es Falsa sin importar si (x/y) > 2 evalúa a Verdadero o Falso.

Cuando Python detecta que no hay nada que ganar evaluando el resto de una expresión lógica, detiene su evaluación y no hace los cálculos en el resto de la expresión lógica. Cuando la evaluación de una expresión lógica se detiene porque el valor global ya se conoce, se llama cortocircuito de la evaluación.

Mientras que esto puede parecer un buen punto, el comportamiento de cortocircuito lleva a una inteligente técnica llamada el patrón del guardián. Considere la siguiente secuencia de códigos en el Intérprete de Python:

Contenido

Evaluación de las expresiones lógicas

```
x = 6
y = 2
print(x >= 2 and (x/y) > 2)

x = 1
y = 0
print(x >= 2 and (x/y) > 2)

x = 6
y = 0
print(x >= 2 and (x/y) > 2)
```

Contenido

Evaluación de las expresiones lógicas

Podemos construir la expresión lógica para colocar estratégicamente una evaluación del guardia justo antes de la evaluación que podría causar un error de la siguiente manera:

```
x = 1
y = 0
print(x >= 2 and y != 0 and (x/y) > 2)
```