Making Decisions

Dr. Baldassano

chrisb@princeton.edu

Yu’s Elite Education

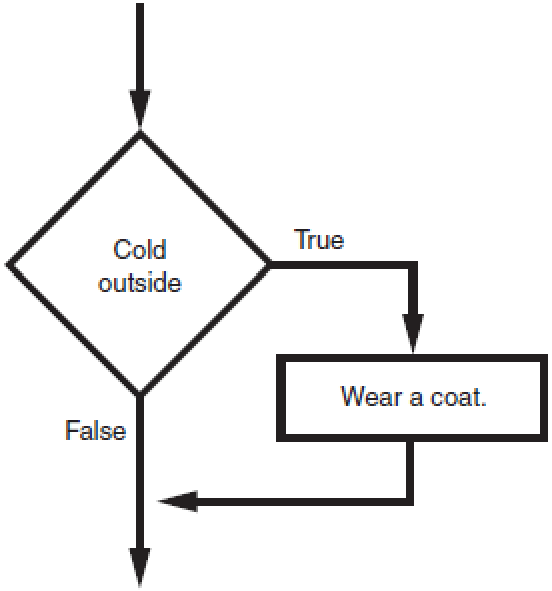
Last week recap

- Functions
  - Void
  - Value-returning
- Importing modules
- Writing functions

Making decisions

- For most programs, we want our actions to depend on the input or the data

Figure 3-1 A simple decision structure



The if statement

- Python syntax:

```
if condition:
      Statement
      Statement
```

- First line is keyword `if` followed by condition
  - The condition can be true or false
  - If it is true the block statements are executed, otherwise block statements are skipped

Example

```
temp = int(input(‘Enter temperature: ’))
if temp < 60:
   print(‘Bring a jacket!’)
```

Boolean Expressions

- The condition of an if statement is a “Boolean expression” that should have a value of either True or False
- Examples:
  - Function that returns True or False:

    ```
    if IsPrime(x):
    ```

  - Relational operator:

    ```
    if x > y:
    ```

Relational Operators

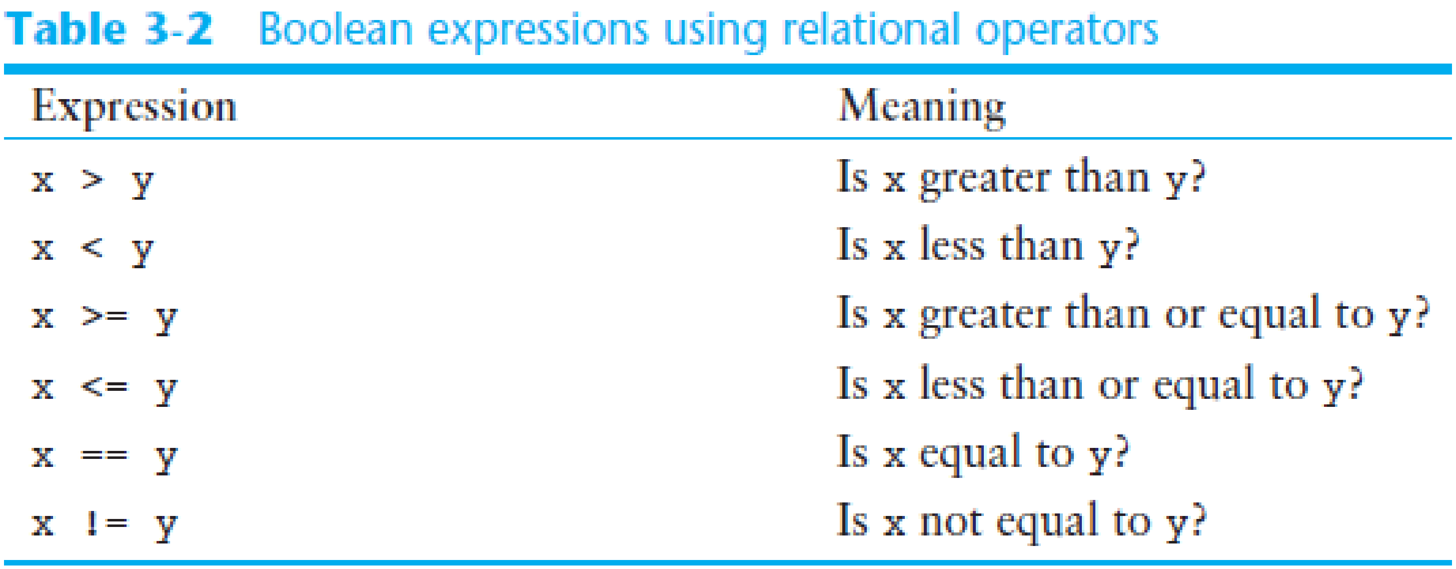


Table 3-2 Boolean expressions using relational operators

Expression	Meaning
`x > y`	Is x greater than y?
`x < y`	Is x less than y?
`x >= y`	Is x greater than or equal to y?
`x <= y`	Is x less than or equal to y?
`x == y`	Is x equal to y?
`x != y`	Is x not equal to y?

Be Careful of = and ==

- Don’t confuse = and ==
- Assignment operator (=)
  - Assigns a variable name on the left to data on the right
  - Usually on a line by itself
- Equality operator (==)
  - More like equals sign in math – tests whether the things on the left and right are equal, takes values True or False
  - Order doesn’t matter

Practice in IDLE

Two possible paths

- What if we want to output one message for temperatures less than 60 and another for temperatures more than 60?

```
if temp < 60:
   print(‘Cold’)
if temp >= 60:
   print(‘Hot’)
```

If else

- Can simplify using the `else` keyword

```
if temp < 60:
    print(‘Cold’)
else:
    print(‘Hot’)
```

Quiz

- Which code prints out whether a number x is equal to 0?

A.
```
if x = 0:
   print(‘Zero’)
else:
   print(‘Positive’)
```

B.
```
if x == 0:
   print(‘Zero’)
else:
print(‘Positive’)
```

C.
```
if x == 0
   print(‘Zero’)
else
   print(‘Positive’)
```

D.
```
if 0 == x:
   print(‘Zero’)
else:
   print(‘Positive’)
```

Lots of paths

- Want to output a student’s letter grade

```
grade = int(input('Grade: '))
if grade >= 90:
   print('A')
if grade >= 80:
   print('B')
if grade >= 70:
   print('C')
if grade >= 60:
   print('D')
```

Lots of paths

```
if grade >= 90:
   print('A')
else:
    if grade >= 80:
       print('B')
    else:
        if grade >= 70:
           print('C')
        else:
            if grade >= 60:
               print('D')
```

if - elif

- Can use the `elif` keyword as an abbreviation for “else if”
- Makes code more readable by getting rid of nesting

if - elif

```
if grade >= 90:
   print('A')
elif grade >= 80:
    print('B')
elif grade >= 70:
    print('C')
elif grade >= 60:
    print('D')
```

if - elif

- Can also add a final “else” statement

```
if year == 2015:
   print('This year')
elif year == 2014:
    print('Last year')
else:
    print('A while back')
```

Logical operators

- What if we want to make a more complicated decision?
- “If you are under 5 or over 80, you cannot ride the rollercoaster”
- “If a number is NOT prime, factor it”
- We can put together multiple conditions using *logical operators*

Logical Operators

- not: reverses the boolean value of what comes after it
  - `if not IsPrime(x):`
- and: true only if both sides are true
  - `if x > 5 and x < 10:`
- or: true if either side is true
  - `if x < 4 or x > 15:`

Truth Tables

	A is True	A is False
not A	False	True

	A True B True	A True B False	A False B True	A False B False
A and B	True	False	False	False
A or B	True	True	True	False

Boolean Practice

- All values of x between 0 and 10 (including 0 and 10)
  - `x >= 0 and x <= 10`
- For string day, is true on Mondays and Wednesdays
  - `day == "Monday" or day == “Wednesday“`
- x is a positive even number less than 5
  - `x == 2 or x == 4`

Quiz

- Which of these has valid syntax? (Two correct answers)

A.

```
if not not x == 0:
   print(‘Zero’)
```

B.

```
if 0 < x < 4:
   print(‘Small’)
```

C.

```
if x == 10 and:
   print(‘Ten’)
```

D.

```
if x == 5 or == 10:
   print(‘Match’)
```

Careful of boolean conversion

- If a variable is used as a boolean expression by itself, it will be interpreted as False if 0, True otherwise
- What does this statement do?

```
if x or y > 5:
```

Shortcurcuit evaluation

- If left side of “and” is false, whole expression must be false
  - `False and (?)` must be false
- If left side of “or” is true, whole expression must be true
  - `True or (?)` must be true
- In this case other side is never evaluated - can be useful for avoiding running a function

Shortcurcuit evaluation

- Imagine we have two functions, IsEven(x) which is fast and IsPrime(x) which is slow
- Can check for prime number as

```
if not IsEven(x) and IsPrime(x):
```

- Will only call IsPrime on odd numbers

Using parentheses

- May need to add parentheses to group expressions

```
if not (x <= 5 or y == 10):
```

DeMorgan’s Law

- It is possible to “multiply” a statement by the not operator
- The not operator distributes to each expression, `and`s are exchanged for `or`s

Algebra: `a*(x+y)  = a*x + a*y`

Boolean:

```
not (x and y) = not x or  not y
not (x or y)  = not x and not y
```

DeMorgan’s Examples

- not (day == ‘Mon’ or day == ‘Tues’)
  - not day == ‘Mon’ and not day == ‘Tues’
- not (x < 0 or x > 10)
  - (not x < 0) and (not x > 10)
  - x >= 0 and x <= 10
- not (x == 10 and y == 5)
  - (not x == 10) or (not y == 5)
  - x != 10 or y != 5

Assignment: Blackjack

- Goal: want a hand of cards worth as close to 21 as possible, without going over
- Value of a hand is the sum of the card values, where:
  - 2-10 worth their number
  - J, Q, K worth 10
  - A worth 11 or 1 – count as 11 unless that would make hand go over 21
- Hands over 21 bust

Blackjack examples

- 5, 2, 4
  - Value: 11
- 2, Q, 4
  - Value: 16
- Q, Q, K
  - Value: Bust
- A, 3, 4
  - Value: 18
- J, Q, A
  - Value: 21

Assignment

- Given three string variables, card1, card2, card3
- Calculate value of blackjack hand

card1 = ‘5’

card2 = ‘9’

card3 = ‘A’

Output: 15