EGM722 – Programming for GIS and Remote Sensing

Week 1, Part 3: Controlling Flow

Recall: Expressions and Statements

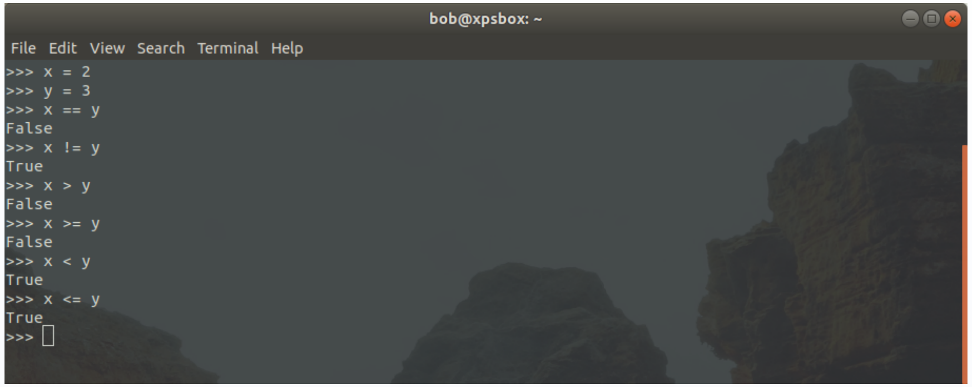
- An expression is a combination of objects, variables, and operators:
  - 42
  - x
  - x+42
- A statement is a unit of code the interpreter can execute
  - Assignment statements (e.g., x = 42)
  - return, pass statements

Boolean expressions

- A Boolean expression evaluates to true or false:
  - meaning_of_life == 42
  - 1 == 0
- The == operator is a comparison operator
  - True if left and right side are equivalent
  - False if not
- In this context, True and False are values of type bool

Comparison operators

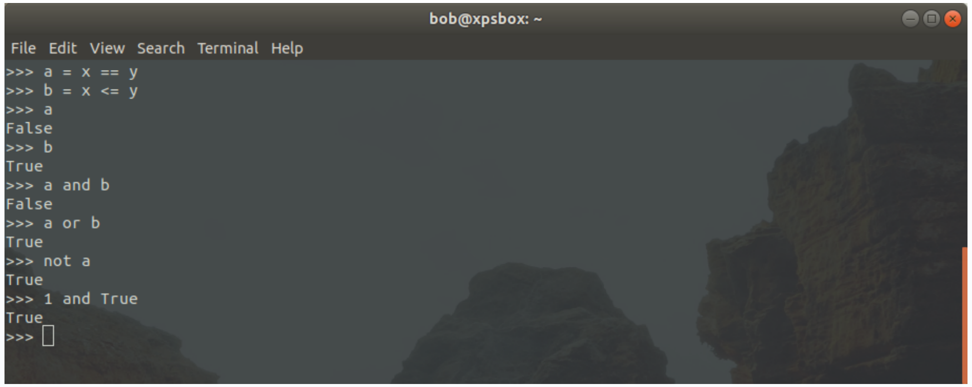
- x == y: x equal to y
- x != y: x not equal to
- x > y
- x >= y
- x < y
- x <= y
- Remember: = is for assignment only!



```
>>> x = 2
>>> y = 3
>>> x == y
False
>>> x != y
True
>>> x > y
False
>>> x >= y
False
>>> x < y
True
>>> x <= y
True
>>> 
```

Logical operators

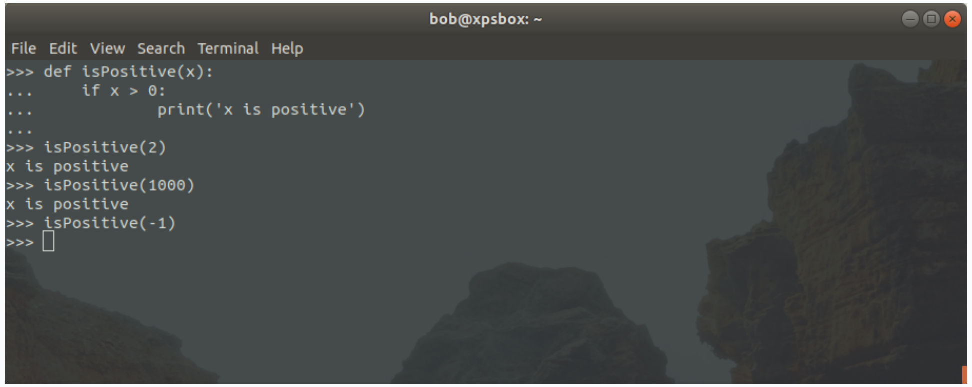
- Operators that compare two expressions
  - a and b: True if both a and b are True
  - a or b: True if either a or b are True
  - not a: True if a is False
- a,b should be boolean expressions, but nonzero values are interpreted as True



```
>>> a = x == y
>>> b = x <= y
>>> a
False
>>> b
True
>>> a and b
False
>>> a or b
True
>>> not a
True
>>> 1 and True
True
>>> 
```

Conditional statements

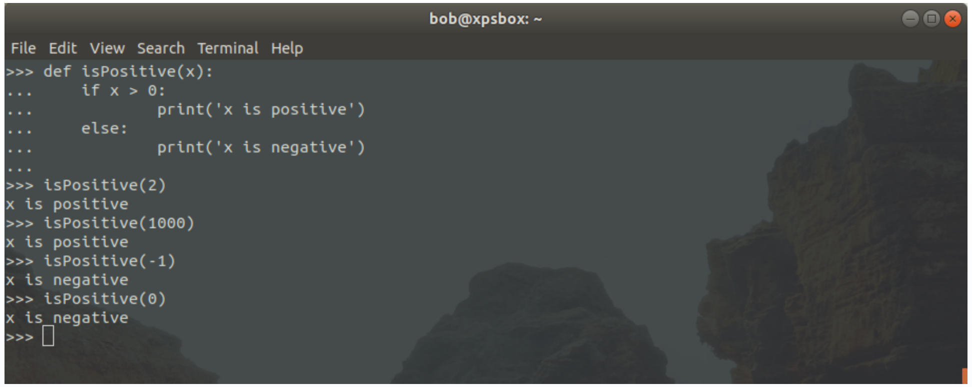
- A conditional statement:
  - executes code if statement is True
- Header: the first line
  - Ends with ‘:’
  - Can be > 1 line
- Body: indented
  - Must have at least one statement
  - Placeholder: pass



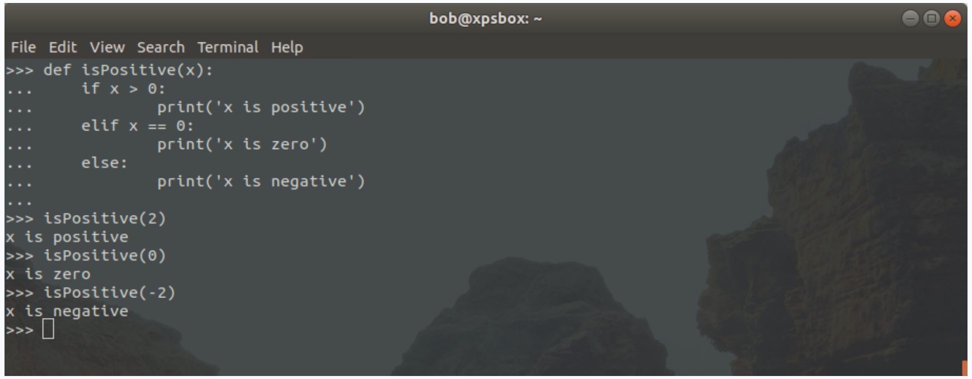
```
>>> def isPositive(x):
...     if x > 0:
...             print('x is positive')
...
>>> isPositive(2)
x is positive
>>> isPositive(1000)
x is positive
>>> isPositive(-1)
>>> 
```

Alternative execution

- Alternative execution:
  - Do different things based on conditions
  - if … else
- Multiple choices: else if (elif)
  - Only one branch is possible
  - No limit on number of elif statements
  - (optional) else clause comes at end
  - Only first True condition is executed



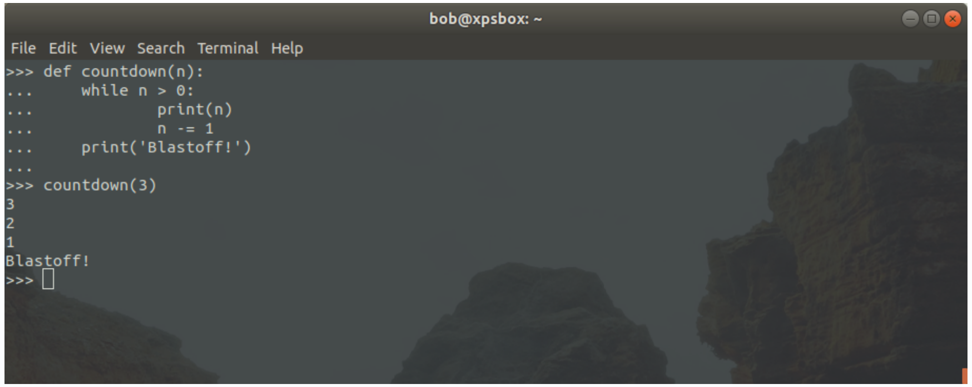
```
>>> def isPositive(x):
...     if x > 0:
...             print('x is positive')
...     else:
...             print('x is negative')
...
>>> isPositive(2)
x is positive
>>> isPositive(1000)
x is positive
>>> isPositive(-1)
x is negative
>>> isPositive(0)
x is negative
>>> 
```



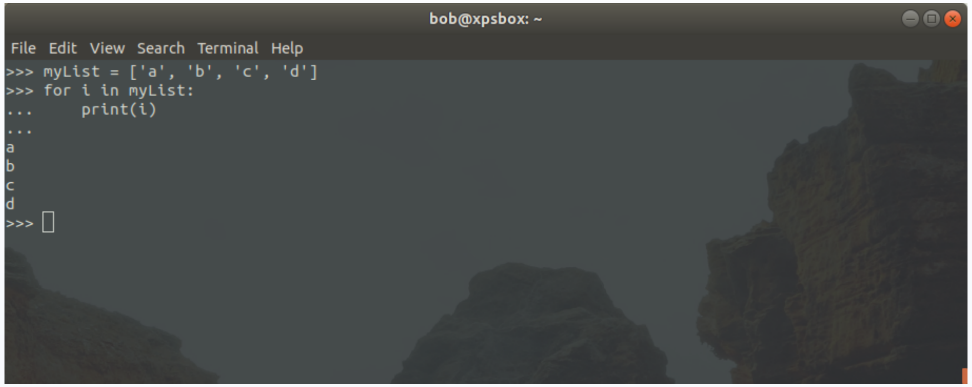
```
>>> def isPositive(x):
...     if x > 0:
...             print('x is positive')
...     elif x == 0:
...             print('x is zero')
...     else:
...             print('x is negative')
...
>>> isPositive(2)
x is positive
>>> isPositive(0)
x is zero
>>> isPositive(-2)
x is negative
>>> 
```

Loops

- What if we want to repeat instructions?
- Python uses 2 main kinds of loops: while, for
- while loops:
  - while statement is true, do (something)
  - When statement is no longer true, stop
  - Body of loop should change at least one variable (otherwise, infinite loops)
- for loops:
  - Iterate a predetermined number of times
  - Any iteratable (e.g., lists, tuples) can be used
  - Can also use range() to set a number



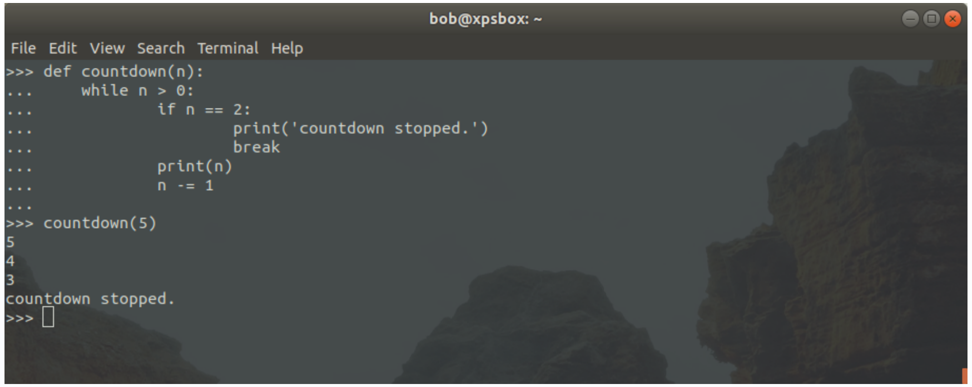
```
>>> def countdown(n):
...     while n > 0:
...             print(n)
...             n -= 1
...     print('Blastoff!')
...
>>> countdown(3)
3
2
1
Blastoff!
>>> 
```



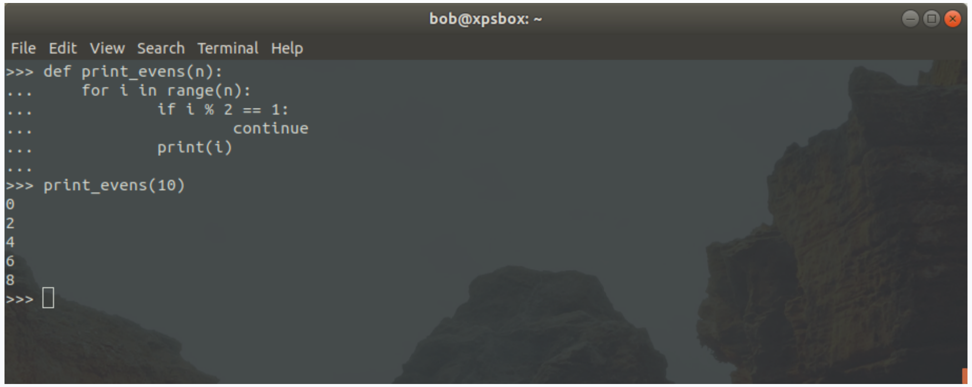
```
>>> myList = ['a', 'b', 'c', 'd']
>>> for i in myList:
...     print(i)
...
a
b
c
d
>>> 
```

Break and continue

- What happens if we want to stop going through a loop?
  - break: stop execution of loop
  - continue: stop execution of this iteration only



```
>>> def countdown(n):
...     while n > 0:
...             if n == 2:
...                     print('countdown stopped.')
...                     break
...             print(n)
...             n -= 1
...
>>> countdown(5)
5
4
3
countdown stopped.
>>> 
```



```
>>> def print_evens(n):
...     for i in range(n):
...             if i % 2 == 1:
...                     continue
...             print(i)
...
>>> print_evens(10)
0
2
4
6
8
>>> 
```

Summary

- Often, we want to control the flow of our programs:
  - Skip/repeat instructions
  - Choose which instructions to run
- Control flow using:
  - Comparison and logical operators
  - Conditional statements (if… elif… else)
  - Loops