Python for Finance

Control Flow, data structures and first application

Andras Niedermayer

DAUPHINE
UNIVERSITÉ PARIS

Outline

Booleans

```
        In [1]: spam = True
        In [2]: span
        Out [2]: True
        In [3]: true
---------------------------------------
---------------------------------------
NameError
Traceback (most recent call last)
<ipython-input-19-74d9a83219ca> in <module>()
----> 1 true

NameError: name 'true' is not defined
```

Operators

Operator	Meaning
==	equal to
!=	unqueal to
<	less than
>	more than
<=	less or equal
>=	greater or equal

Booleans

```
In [1]: 42 == 42
Out [1]: True
In [2]: 42 == 99
Out [2]: False
In [3]: 2 != 3
Out [3]: True
In [4]: 2 != 2
Out [4]: False
```

Booleans

```
In [1]: 'hello' == 'hello'
In [2]: 'hello' == 'Hello'
In [3]: 'dog' != 'cat'
In [4]: True == True
In [5]: True != False
In [6]: 42 == 42.0
In [7]: 42 == '42'
```

Booleans

```
In [1]: (4 < 5) and (5 < 6)
In [2] (4 < 5) and (9 < 6)
In [3]: (1==2) or (2==2)
In [4]: 2+2==4 and not 2+2==5 and 2*2==2+2
```

Code blocks

In Python, code blocks are delimited by indentation and in general the beginning of a control flow statement ends with a colon (':')

```
In [1]: name = 'Mary'
In [2]: password = 'swordfish'
In [3]: if name == 'Mary':
...   :     print 'Hello Mary'
...   :     if password == 'swordfish':
...   :         print 'Access granted.'
...   :     else:
...   :         print 'Wrong password.'
```

If, then, else

If

```
In [1]: if name == 'Mary':
...   :     print 'Hello Mary'
```

Else

```
In [1]: if name == 'Mary':
...   :     print 'Hello Mary'
...   : else:
...   :     print 'Who are you?'
```

Else if (=elif)

```
In [1]: name = 'Bob'
In [2]: age = 5
In [3]: if name == 'Alice':
...   :     print 'Hi, Alice.'
...   : elif age < 12:
...   :     print 'You are not Alice, kiddo.'
```

Loops

- While loops

```
In [1]: spam = 0
In [2]: while spam < 5:
...   :     print 'Hello World, '\
...   :           'number {}.'.format(spam)
...   :     spam = spam + 1
```

- Break

```
In [1]: while True:
...   :     print 'Please type your name.'
...   :     name = raw_input()
...   :     if name == 'your name':
...   :         break
In [2]: print 'Thank you.'
```

Loops

- Continue

```
In [1]: while True:
...   :     print 'Who␣are␣you?'
...   :     name = raw_input()
...   :     if name != 'Joe':
...   :         continue
...   :     print 'Hello,␣Joe.␣What␣is␣the␣pass
...   :     password = raw_input()
...   :     if password == 'swordfish':
...   :         break
In [2]: print 'Access␣granted.'
```

For loops

For loops

```
In [1]: for spam in range(5):
...   :     print 'Hello World, '\
...   :           'number {}.'.format(spam)
```

Range

```
In [1]: help(range)
```

range(...) range(stop) $->$ list of integers range(start, stop[, step]) $->$ list of integers

Return a list containing an arithmetic progression of integers. range(i, j) returns [i, i+1, i+2, ..., j-1]; start (!) defaults to 0. When step is given, it specifies the increment (or decrement). For example, range(4) returns [0, 1, 2, 3]. The end point is omitted! These are exactly the valid indices for a list of 4 elements.

Excercise

8. Write code that prints 'Hello' if the variable spam is equal to 1, 'Howdy' if it is equal to 2 and 'Greetings' in all other cases.
9. what is the difference between break and continue?
10. What is the difference between range(10), range(0, 10) and range(0,10,1)?
11. Write a program that calculates the sum of the first 20 integers using a for loop.
12. Write a program that calculates the sum of the first 20 integers using a *while* loop.
13. Write a program that calculates the sum of the first 20 integers using a closed form solution for the sum.

Solutions

- Excercise 8:

```
In [1]: if spam == 1:
...   :   val = 'Hello'
...   : elif spam == 2:
...   :     val = 'Howdy'
...   : else:
...   :     val = 'Greetings'
...   :
In [2]: print val
```

- Excercise 9: Break quits a loop, whereas continue returns to the beginning of the loop and continues running the code in the loop.
- Excercise 10: There is no difference! range(10) has the value stop=0 and uses the default values start=0 and step=1. range(0,10) and range(0,10,1) specify these parameters explicitly

Solutions

- Excercise 11:

```
In [1]: mysum = 0
In [2]: for i in range(1, 21):
...   :     mysum = mysum + i
...   :
In [3]: print mysum
```

Shorthand for `some = some + 1`: `some += 1`

- Excercise 12:

```
In [1]: mysum = 0
In [2]: i = 1
In [2]: while i < 21:
...   :     mysum += i
...   :     i += 1
...   :
In [3]: print mysum
```

Solutions

- Excercise 13:

```
In [1]: mysum = (20*21)/2
```

- Remark: An alternative way is to use Python's built=in 'sum' function:

```
In [1]: mysum = sum(range(1, 21))
```

Outline

Functions/Subroutines

Wikipedia:
"In computer programming, a subroutine/function is a sequence of program instructions that perform a specific task, packaged as a unit. This unit can then be used in programs wherever that particular task should be performed."

Functions

- Defining a function

```
In [1]: def myfunction():
...   :     print 'Hello World!'
...   :
```

- Using a function

```
In [1]: myfunction()
```