Python Basics

Nakul Gopalan
ngopalan@cs.brown.edu

With help from Cam Allen-Lloyd

Introduction to the idea

- Readable, easy to learn programming language.
- Created by Guido van Rossum
- Named after the BBC show "Monty Python's Flying Circus".
- The official definition of the programming language as given by the Python Software Foundation is:
  “Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.”

Python philosophy

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability counts
- On the interpreter try: import this

Hello World!

- Python is open source and free to download.
- For this lecture I am hosting an interpreter on my webpage.
- Python 2.7 come pre-installed in most Unix OS.
- The interpreter can be called from terminal by typing `python`.
- Printing “Hello World!”: `print "Hello World!"`

Simple script

- Scripts can also be saved and then run from the terminal.
- Consider the script random_ints.py:

```
import random
for i in range(5):

    print(random.randint(10,99))
```

Indentation

- Python does not have begin or end statements for code blocks
- Uses colons (:) and indentation instead. Try Running the following code block:

```
x = 0

while x < 10:

    if x % 2 == 0:

        print x

    x += 1

print 'done.'
```

Dynamic Typing

```
var = 5

print var

print type(var)

var = "spam"

print var

print type(var)
```

- In Python a variable name is bound only to an object and not to a data type in any way.
- Basic data types in python are immutable.

Sequence Types: String

```
mystring = 'ham and eggs'

print mystring[0:4]

print mystring.find('and')

print mystring.split(' ')
```

Sequence Types: Lists

```
mylist = []
mylist.append(1)
mylist.append(2)
mylist.append("three")

print mylist

newlist = [1,1,2,3,5,8,13]
newlist[4] = 3000
print newlist[4]
print newlist[-1]
newlist.pop()
print newlist
```

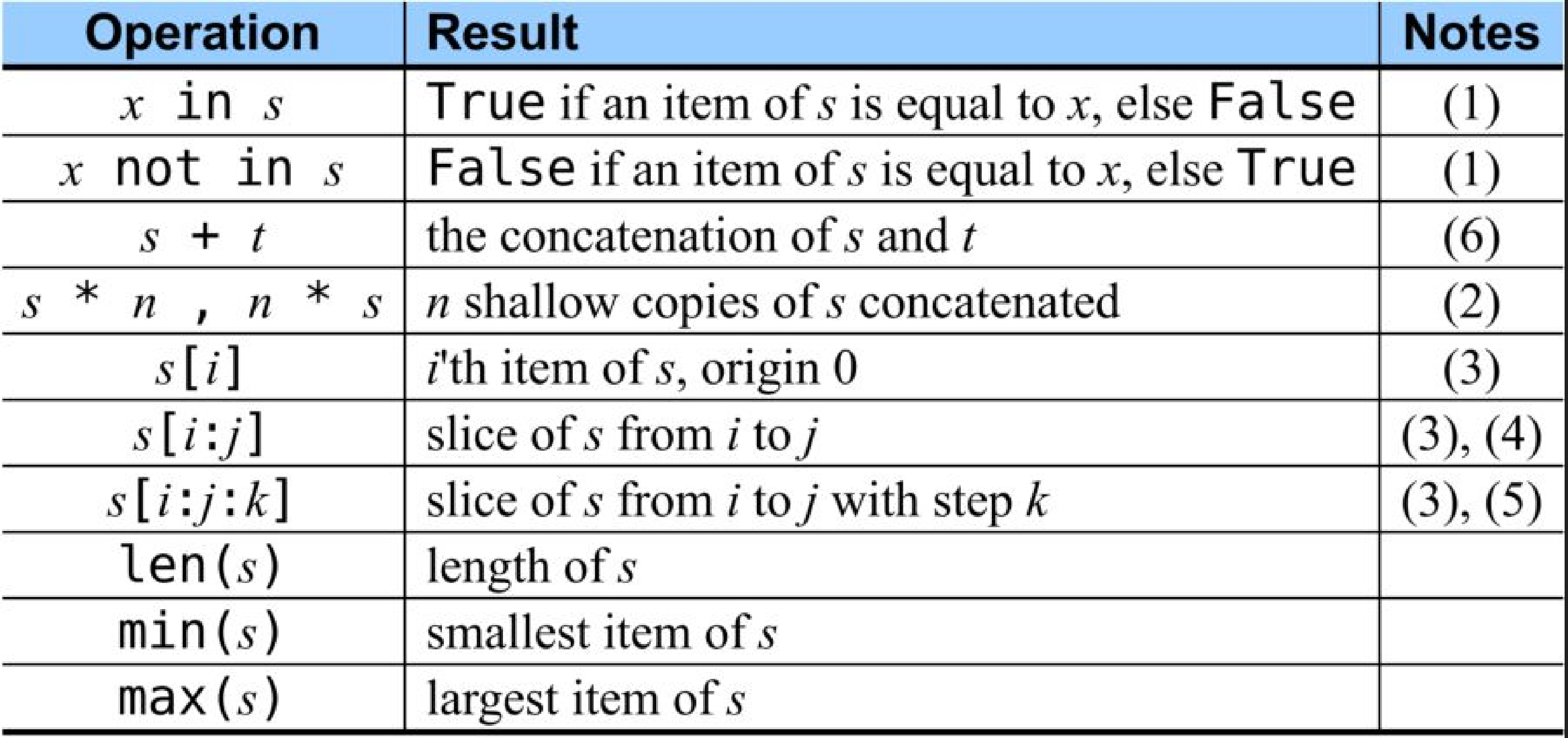
Lists are mutable objects.

Sequence Types: Tuples

```
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');

tup1[0] = 100;



tup3 = tup1 + tup2;
print tup3

print len tup3

for x in tup3: print x
```

- Tuples are immutable objects.
- However their constituent elements can be altered.

Operations on Sequence Types



Operation	Result	Notes
x in s	True if an item of s is equal to x, else False	(1)
x not in s	False if an item of s is equal to x, else True	(1)
s + t	the concatenation of s and t	(6)
s * n , n * s	n shallow copies of s concatenated	(2)
s[i]	i'th item of s, origin 0	(3)
s[i:j]	slice of s from i to j	(3), (4)
s[i:j:k]	slice of s from i to j with step k	(3), (5)
len(s)	length of s	
min(s)	smallest item of s	
max(s)	largest item of s	

From the Python Reference manual

Data Type: Dictionaries

```
numbers = {'one': 1, 'two': 2, 'three': 3, 'four': 4 }

print numbers['one']

del numbers['one']

print numbers

print numbers.keys()
```

Name binding

```
a = [1, 2]
b = a
print b, a
b.append(3)
print a

a = 1
b = a
print b, a
b=b+1
print a
print b
```

- Python has name binding, that is, names bind to objects.

Control Flow Statements: Indentation important!!!

```
age = 22

if age < 13:
    print 'kid'
elif age < 18:
    print 'teen'
else:
    print 'adult'
```

```
for i in range(5):
    pass

for i in [0, 1, 2, 3, 4]:
    if i > 5:
        break
else:
    print 'Did not break'
```

```
x = 1024

while x > 1:
    x = x / 2
    if (x % 10) != 2:
        continue
    print x
```

Functions

```
def fib(n=10):
    print "Prints a Fibonacci series up to %d: " %n
    a, b = 1, 1
    while a < n:
        print a,
        a, b = b, a+b
    print "."
    return None

fib(500)
fib()
```

Classes

```
import math

class Vector2:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def len(self):
        return math.sqrt(self.x**2 + self.y**2)

    def __str__(self):
        return "From str method of Vector2: x is %d, y is %d" % (self.x, self.y)

    __DoNotTouch = 10


v = Vector2(3, 4)
print v
print "({},{}):".format(v.x, v.y), "len = {}".format(v.len())
```

Inheritance

```
class shape:
    def __init__(self, b, h):
        self.base = b
        self.height = h
    def __str__(self):
        return str( (self.base, self.height) )

class rectangle(shape):
    def area(self):
        return self.base * self.height

class triangle(shape):
    def area(self):
        return self.base * self.height / 2



rect = rectangle(4.0, 3.0)
tri  = triangle(4.0, 3.0)
print " rect: {}   area: {}".format(rect, rect.area())
print " tri:  {}   area: {}".format(tri, tri.area())
```

Numpy (Numeric Python) Basics

```
import numpy as np

a = np.array([1, 4, 5, 8], float)
print a

m = np.array([[1, 2, 3], [4, 5, 6]], float)
print m

m = np.array(range(8), float).reshape((2, 4))
print m, m.shape
print m.transpose(), m.transpose().shape
```

Numpy basics

Other functions include:

- a.fill(x) to fill array with x
- a.tolist() to convert to list
- np.concatenate((a, b, c)) to concatenate arrays
- np.ones((2,3), dtype=float) to create arrays with ones.
- np.identity(4, dtype=float) to create identity matrix

Numpy Array Math

```
import numpy as np

a = np.array([1,2,3], float)
b = np.array([5,2,6], float)

for x in a: print x

print "add: ", a+b
print "multiply: ", a*b
print "divide: ", b/a
print "square root: ", np.sqrt(a)

print "matrix multiplication: ", np.dot(a, b)

#print "logic values: ", b>a
#np.poly([-1, 1, 1, 10])

#print np.mean(a)
#print np.var(a)
```

Matplotlib

```
import matplotlib.pyplot as pl
import numpy as np
x = np.linspace(0,2*np.pi,50)
y = np.cos(x)
pl.plot(x,y)
pl.title("my plot")
pl.xlabel(' x')
pl.ylabel('Cos')

pl.show()

#plt.savefig('sine.png')
```

Sources

- Content is based on slides by Zhenyu Zhou, Richard Guo and Cam Allen-Lloyd
- python.org - Official Python website
- Berkeley Python/UNIX tutorial - Available on course webpage
- learnpython.org - Basic tutorials, examples
- A Byte of Python - Beginner’s tutorial
- Oliver Fromme - Python Information and Examples
- Trinket.io for their online interpreter and codebase
- Numpy tutorial M. Scott Shell, UCSB Engineering