Python language: Basics

The FOSSEE Group

Department of Aerospace Engineering
IIT Bombay

SciPy.in 2010, Tutorials

Outline

Outline

Primitive Data types

- Numbers: float, int, complex
- Strings
- Booleans

Outline

Numbers

- **int**
  whole number, no matter what the size!

```
In []: a = 13

In []: b = 99999999999999999999
```

- **float**

```
In []: p = 3.141592
```

- **complex**

```
In []: c = 3+4j
```

Outline

Booleans

```
In [ ]: t = True

In [ ]: F = not t

In [ ]: F or t
Out[ ]: True

In [ ]: F and t
Out[ ]: False
```

( ) for precedence

```
In [ ]: a = False
In [ ]: b = True
In [ ]: c = True

In [ ]: (a and b) or c
Out[ ]: True

In [ ]: a and (b or c)
Out[ ]: False
```

Outline

Strings

Anything within “quotes” is a string!

```
' This is a string '
" This too! "
""" This one too! """
''' And one more! '''
```

Strings

Why so many?

```
' "Do or do not.  No try." said Yoda.'
" ' is a mighty lonely quote."
```

The triple quoted ones can span multiple lines!

```
""" The quick brown
fox jumped over
    the lazy dingbat.
"""
```

Strings

```
In [ ]: w = "hello"

In [ ]: print w[0], w[1], w[-1]

In [ ]: len(w)
Out[ ]: 5
```

Strings . . .

Strings are immutable

```
In [ ]: w[0] = 'H'

---------------------------------------------------------------
TypeError  Traceback (most recent call la

<ipython console> in <module>()

TypeError: 'str' object does not
           support item assignment
```

Strings . . .

Strings are immutable

```
In []: w[0] = 'H'

---------------------------------------------
TypeError  Traceback (most recent call la

<ipython console> in <module>()

TypeError: 'str' object does not
           support item assignment
```

Outline

Arithmetic operators

```
In [ ]: 1786 % 12
Out[ ]: 10

In [ ]: 45 % 2
Out[ ]: 1

In [ ]: 864675 % 10
Out[ ]: 5

In [ ]: 3124 * 126789
Out[ ]: 396088836

In [ ]: big = 12345678912345678890 ** 3

In [ ]: verybig = big * big * big * big
```

Arithmetic operators

```
In [ ]: 17 / 2
Out[ ]: 8

In [ ]: 17 / 2.0
Out[ ]: 8.5

In [ ]: 17.0 / 2
Out[ ]: 8.5

In [ ]: 17.0 / 8.5
Out[ ]: 2.0
```

Arithmetic operators

```
In [ ]: c = 3+4j

In [ ]: abs(c)
Out[ ]: 5.0

In [ ]: c.imag
Out[ ]: 4.0

In [ ]: c.real
Out[ ]: 3.0
```

Arithmetic operators

```
In [ ]: a = 7546

In [ ]: a += 1
In [ ]: a
Out[ ]: 7547

In [ ]: a -= 5
In [ ]: a

In [ ]: a *= 2

In [ ]: a /= 5
```

String operations

```
In [ ]: s = 'Hello'

In [ ]: p = 'World'

In [ ]: s + p
Out[ ]: 'HelloWorld'

In [ ]: s * 4
Out[ ]: 'HelloHelloHelloHello'
```

String operations . . .

```
In []: s * s
---------------------------------------------------------------------------
TypeError  Traceback (most recent call la

<ipython console> in <module>()

TypeError: can't multiply sequence by
                    non-int of type 'str'
```

String operations . . .

```
In []: s * s

---------------------------------------------
TypeError  Traceback (most recent call la

<ipython console> in <module>()

TypeError: can't multiply sequence by
                    non-int of type 'str'
```

String methods

```
In [ ]: a = 'Hello World'
In [ ]: a.startswith('Hell')
Out[ ]: True

In [ ]: a.endswith('ld')
Out[ ]: True

In [ ]: a.upper()
Out[ ]: 'HELLO WORLD'

In [ ]: a.lower()
Out[ ]: 'hello world'
```

Strings: split & join

```
In [ ]: chars = 'a b c'
In [ ]: chars.split()
Out[ ]: ['a', 'b', 'c']
In [ ]: ' '.join(['a', 'b', 'c'])
Out[ ]: 'a b c'

In [ ]: alpha = ', '.join(['a', 'b', 'c'])
In [ ]: alpha
Out[ ]: 'a, b, c'
In [ ]: alpha.split(', ')
Out[ ]: ['a', 'b', 'c']
```

String formatting

```
In []: x, y = 1, 1.234

In []: 'x is %s, y is %s' %(x, y)
Out[]: 'x is 1, y is 1.234'
```

- *%d*, *%f* etc. available

http://docs.python.org/library/stdtypes.html

Relational and logical operators

```
In [ ]: p, z, n = 1, 0, -1
In [ ]: p == n
Out[ ]: False

In [ ]: p >= n
Out[ ]: True

In [ ]: n < z < p
Out[ ]: True

In [ ]: p + n != z
Out[ ]: False
```

Built-ins

```
In [ ]: int(17 / 2.0)
Out[ ]: 8

In [ ]: float(17 / 2)
Out[ ]: 8.0

In [ ]: str(17 / 2.0)
Out[ ]: '8.5'

In [ ]: round( 7.5 )
Out[ ]: 8.0
```

Odds and ends

- Case sensitive
- Dynamically typed $\Rightarrow$ need not specify a type

```
In [ ]: a = 1
In [ ]: a = 1.1
In [ ]: a = "Now I am a string!"
```

- Comments:

```
In [ ]: a = 1   # In-line comments
In [ ]: # A comment line.
In [ ]: a = "# Not a comment!"
```

Outline

Simple IO: Console Input

- raw_input() waits for user input.

```
In []: a = raw_input()
5

In []: a
Out[]: '5'

In []: a = raw_input('Enter a value: ')
Enter a value: 5
```

- Prompt string is optional.
- All keystrokes are strings!
- `int()` converts string to int.

Simple IO: Console output

- **`print`** is straight forward
- Put the following code snippet in a file **`hello1.py`**

```
print "Hello"
print "World"
```

```
In []: %run -i hello1.py
Hello
World
```

Simple IO: Console output . . .

Put the following code snippet in a file **hello2.py**

```
print "Hello",
print "World"
```

```
In []: %run -i hello2.py
Hello World
```

Note the distinction between **print x** and **print x,**

Outline

Control flow constructs

- `if/elif/else` : branching
- `while` : looping
- `for` : iterating
- `break, continue` : modify loop
- `pass` : syntactic filler

Outline

if...elif...else example

Type the following code in an editor & save as ladder.py

```
x = int(raw_input("Enter an integer:"))
if x < 0:
    print 'Be positive!'
elif x == 0:
    print 'Zero'
elif x == 1:
    print 'Single'
else:
    print 'More'
```

Outline

Outline

while

Example: Fibonacci series

Sum of previous two elements defines the next

```
In []: a, b = 0, 1
In []: while b < 10:
   ...:     print b,
   ...:     a, b = b, a + b
   ...:
   ...:
1 1 2 3 5 8
```

range()

range([start,] stop[, step])

- **range()** returns a list of integers
- The **start** and the **step** arguments are optional
- **stop** is not included in the list

Documentation convention

- Anything within **[]** is optional
- Nothing to do with Python.

for ...range()

Example: print squares of first 5 numbers

```
In []: for i in range(5):
  ....:     print i, i * i
  ....:
  ....:
0 0
1 1
2 4
3 9
4 16
```

for ...range()

Example: print squares of odd numbers from 3 to 9

```
In []: for i in range(3, 10, 2):
  ....:     print i, i * i
  ....:
  ....:
3 9
5 25
7 49
9 81
```

Outline

Problem 1.1: *Armstrong* numbers

Write a program that displays all three digit numbers that are equal to the sum of the cubes of their digits.
That is, print numbers *abc* that have the property
$abc = a^3 + b^3 + c^3$
For example, $153 = 1^3 + 5^3 + 3^3$

Problem 1.2: Collatz sequence

1. Start with an arbitrary (positive) integer.
2. If the number is even, divide by 2; if the number is odd, multiply by 3 and add 1.
3. Repeat the procedure with the new number.
4. It appears that for all starting values there is a cycle of 4, 2, 1 at which the procedure loops.

Write a program that accepts the starting value and prints out the Collatz sequence.

What did we learn?

- Data types: int, float, complex, boolean, string
- Operators: +, -, *, /, %, **, +=, -=, *=, /=, >, <, <=, >=, ==, !=, a < b < c
- Simple IO: `raw_input` and `print`
- Conditionals: `if elif else`
- Looping: `while` & `for`