Python language: Basics

The FOSSEE Group

Department of Aerospace Engineering
IIT Bombay

7 February, 2010
Day 2, Session 1

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 numbers

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

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

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

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

Outline

Booleans

```
In []: t = True

In []: f = not t

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

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

5 m

( ) 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
```

10 m

Outline

FOSSEE Team (IIT Bombay)   Basic Python   11 / 36

Strings

Strings were introduced previously, let us now look at them in a little more detail.

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

In []: print w[0] + w[2] + w[-1]
Out[]: hlo

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
```

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'
```

A bit about IPython

Recall, we showed a few features of IPython, here is one more:

- IPython provides better help
- object.function?

  ```
  In []: a = 'Hello World'
  In []: a.lower?
  ```

Still with strings

- We saw split() yesterday
- join() is the opposite of split()

```
In []: ''.join(['a', 'b', 'c'])
Out[]: 'abc'
```

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
```

20 m

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 = 1234567891234567890 ** 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 []: 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`
```

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!"
```

35 m

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

1. Data types
   - Numbers
   - Booleans
   - Strings

2. Operators

3. Simple IO

4. Control flow
   - Basic Conditional flow

Control flow constructs

- **if/elif/else** : branching
- **C if X else D** : Ternary conditional operator
- **while** : looping
- **for** : iterating
- **break, continue** : modify loop
- **pass** : syntactic filler

Outline

If...elif...else example

Type out the code below in an editor.

```
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'
```

45 m

Ternary conditional operator

```
...
a = raw_input('Enter number(Q to quit):')
num = int(a) if a != 'Q' else 0
...
```

What did we learn?

- Data types: int, float, complex, boolean, string
- Operators: +, -, *, /, %, **, +=, -=, *=, /=, >, <, <=, >=, ==, !=, a < b < c
- Simple IO: `raw_input` and `print`
- Conditional structures: `if/elif/else`, `C if X else D`