Python language: Basics

The FOSSEE Group

Department of Aerospace Engineering
IIT Bombay

Mumbai, India

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
  In []: c = complex(3, 4)
  ```

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
```

Finding the type

```
In []: a = 1.0
In []: type(a)
Out[]: float
In []: type(1)
Out[]: int

In []: type(1+1j)
Out[]: complex

In []: type('hello')
Out[]: str
```

Outline

1. Data types
   - Numbers
   - Booleans
   - Strings

2. Operators

3. Simple IO

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[]: 8.5     # 8 on Python 2.x

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

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

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

Arithmetic operators: floor division

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

In []: 17 // 2.0
Out[]: 8.0

In []: 17.0 // 2.0
Out[]: 8.0

In []: 17.0 // 8.6
Out[]: 1.0
```

Arithmetic operators

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

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

In []: c.imag
Out[]: 4.0
In []: c.real
Out[]: 3.0

In []: c.conjugate()
(3-4j)
```

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'
```

String methods

```
In []: a = '   Hello World   '
In []: b = a.strip()
In []: b
Out[]: 'Hello World'

In []: b.index('ll')
Out[]: 2
In []: b.replace('Hello', 'Goodbye')
Out[]: 'Goodbye 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
```

The `assert` statement

- You will see it in tests and your exam!

```
In []: assert p != n
In []: assert p == n
---------------------------------------------------
AssertionError          Traceback (most recent call l
----> 1 assert p == n

AssertionError:
```

- No error if condition is True
- Raises error if False

assert examples

```
In []: assert p == n, "Oops condition failed"
-----------------------------------------------------
AssertionError          Traceback (most recent call l
----> 1 assert p == n

AssertionError: Oops condition failed
```

- Can supply an optional message

String containership

```
In []: fruits = 'apple, banana, pear'
In []: 'apple' in fruits
Out[]: True

In []: 'potato' in fruits
Out[]: False
```

- Use tab complete to list other string methods
- Use **?** to find more information

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!"
  ```

Exercise 1

Given a 2 digit integer **x**, find the digits of the number.

- For example, let us say **x = 38**
- Find a way to get **a = 3** and **b = 8** using **x**?

Possible Solution

```
In []: a = x//10
In []: b = x%10
In []: a*10 + b == x
```

Another Solution

```
In []: sx = str(x)
In []: a = int(sx[0])
In []: b = int(sx[1])
In []: a*10 + b == x
```

Exercise 2

Given an arbitrary integer, count the number of digits it has.

Possible solution

```
In []: x = 12345678
In []: len(str(x))
```

Sneaky solution!

Outline

Simple IO: Console Input

- **input()** waits for user input.
  ```
  In []: a = input()
  5

  In []: a
  Out[]: '5'   # Python 3.x!

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

- Prompt string is optional.

- **input** produces strings (Python 3.x)

- **int()** converts string to int.

Simple IO: Python 2.x vs 3.x

- **print** is familiar to us
- Changed from Python 2.x to 3.x
- We use the Python 3 convention here

- If on Python 2.x do this:
  In []: from __future__ import print_function
- Safe to use in Python 3.x also

Simple IO: Console output

- Put the following code snippet in a file **hello1.py**

```
from __future__ import print_function
print("Hello", "World")
print("Goodbye", "World")
```

Now run it like so:

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

Simple IO: Console output . . .

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

```python
from __future__ import print_function
print("Hello", end=' ')
print("World")
```

Now run it like so:

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

Mini Exercise

- Read docs for **print**
- Remember: use **print?**

Simple IO: Console output . . .

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

```python
from __future__ import print_function
print("Hello", end=' ')
print("World")
```

Now run it like so:

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

Mini Exercise

- Read docs for **print**
- Remember: use **print?**

What did we learn?

- Data types: int, float, complex, boolean, string
- Use **type** builtin function to find the type
- Operators: +, -, *, /, %, **, +=, -=, *=, /=, >, <, <=, >=, ==, !=, a < b < c
- Simple IO: **input** and **print**

Homework

- Explore the various string methods
- Read the documentation for the string methods
- Explore the different builtins seen so far also