CS 133 - Introduction to Computational and Data Science

Instructor: Renzhi Cao
Computer Science Department
Pacific Lutheran University
Spring 2017

Previous class

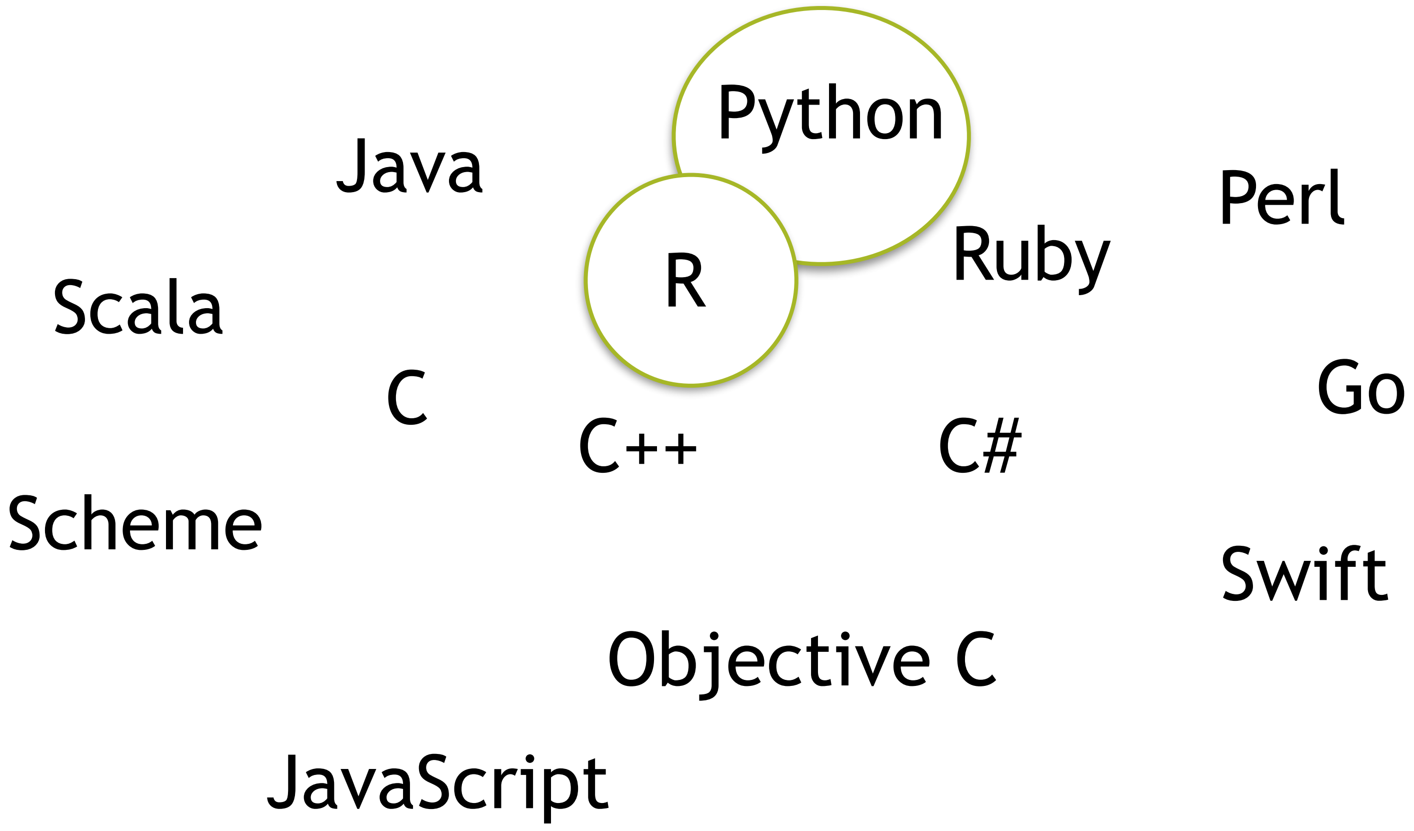
- We have learned the path and file system.
- Today we are going to learn how to use Atom and get our first Python program!

Atom

- Demo how to open and use Atom

Introduction to Python

- What is Python? Why do people use Python?

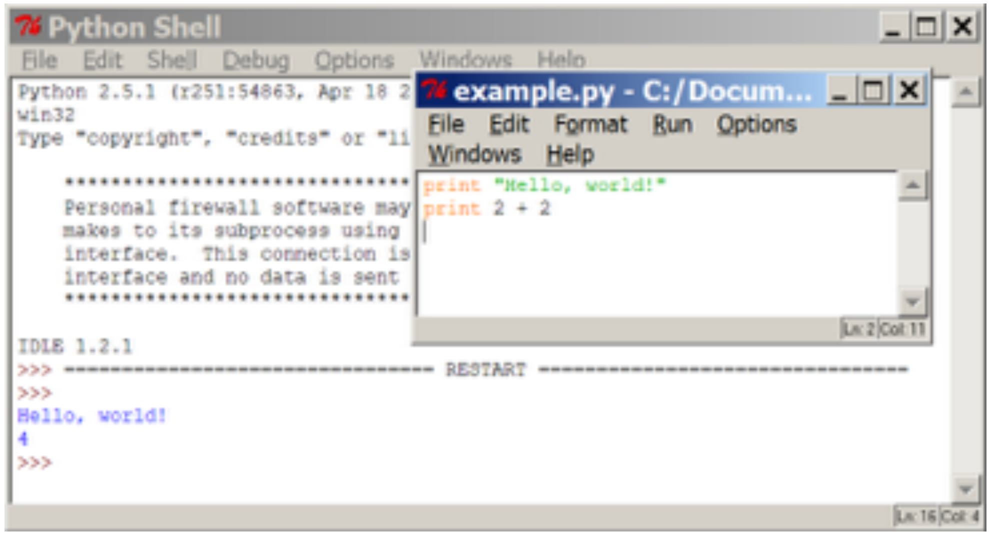


Introduction to Python

- It’s free
- It’s portable
- It’s powerful
- It’s mixable

Programming basics

- **code** or **source code**: The sequence of instructions in a program.
- **syntax**: The set of legal structures and commands that can be used in a particular programming language.
- **output**: The messages printed to the user by a program.
- **console**: The text box onto which output is printed.
  - Some source code editors pop up the console as an external window, and others contain their own console window.

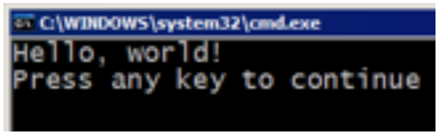


```
Python 2.5.1 (r251:54863, Apr 18 2
win32
Type "copyright", "credits" or "li

    ******************************
    Personal firewall software may
    makes to its subprocess using
    interface.  This connection is
    interface and no data is sent
    ******************************

IDLE 1.2.1
>>> ================================ RESTART ================================
>>>
Hello, world!
4
>>>
```

```
print "Hello, world!"
print 2 + 2
```



```
Hello, world!
Press any key to continue
```

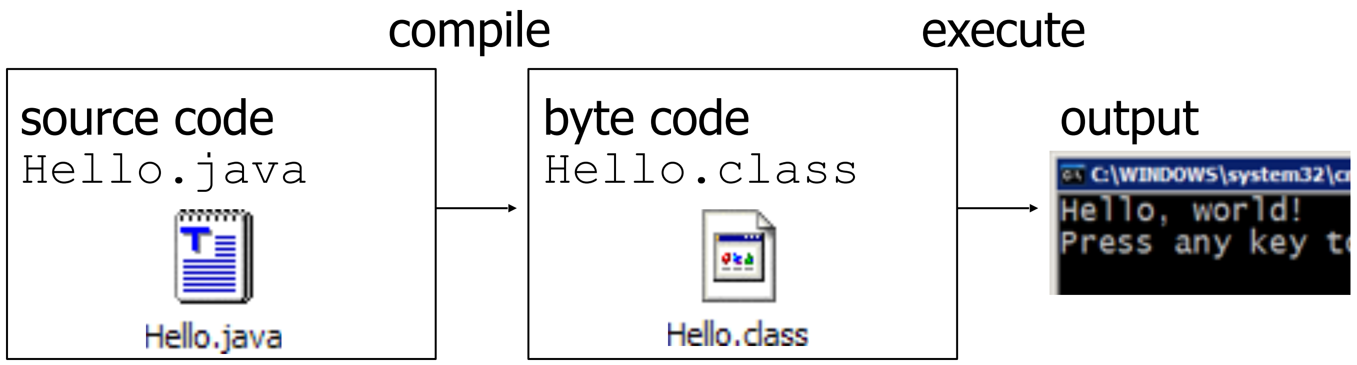
The Python Interpreter

- Python is an interpreted language
- The interpreter provides an interactive environment to play with the language
- Results of expressions are printed on the screen

```
>>> 3 + 7
10
>>> 3 < 15
True
>>> 'print me'
'print me'
>>> print 'print me'
print me
>>>
```

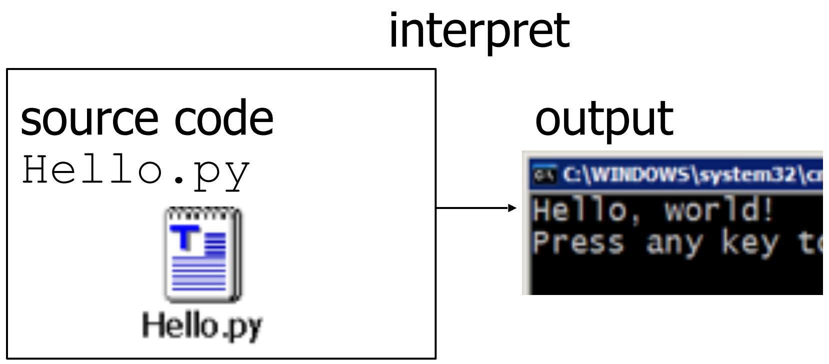
Compiling and interpreting

- Many languages require you to *compile* (translate) your program into a form that the machine understands.



```
Hello, world!
Press any key to
```

- Python is instead directly *interpreted* into machine instructions.



```
Hello, world!
Press any key t
```

How to run?

- Python scripts end with .py
- python test.py
- python2 test.py

Python 2 vs **Python 3**

- print 'Hello, World!‘ vs print(' Hello, World!')

Demo: Hello World

•Open a terminal window and type “python”

•If on Windows open a Python IDE like IDLE

•At the prompt type ‘hello world!’

```
>>> 'hello world!'
'hello world!'
```

Demo: Hello World

- In Atom, create a python script: helloworld.py.

Let’s do it together!

How to run?

- We need to “navigate” until we find our file to be able to execute it.
- To do that we need to open a command prompt window.
- Type the words cmd in the search bar and click on the command prompt.
- To find your file you can use the following commands
  - **cd name_of_directory** to change directory
  - **cd ..** To go one directory “below” (Note that there is a space between cd and the 2 dots)
  - **dir** list all files in the current directory

Handout 2

- Let’s practice it for a simple Python program!

Comments

- Documenting your code is very important
- Use # to write any message that will be ignored by Python.
- This can also be used to test your code

# This is a single line comment

- ‘’’ is used to have multiline comments

‘’’

This is a multi

Lin

E

Comment

‘’’

The print statement

- Elements separated by commas print with a space between them
- A comma at the end of the statement (print ‘hello’,) will not print a newline character

```
>>> print 'hello'
hello
>>> print 'hello', 'there'
hello there
```

variables

- In Python, like in other languages, we store values in variables. Unlike other languages, in Python the variables don’t have a “type”
- Use of single quotes ‘’ represents text. No quotes represents numbers

- >>>message = ‘Hello’
- >>>print message
- >>>message = “Hello”
- >>>print message
- >>>message = ‘”Hello’”
- >>>print message

Everything is an object

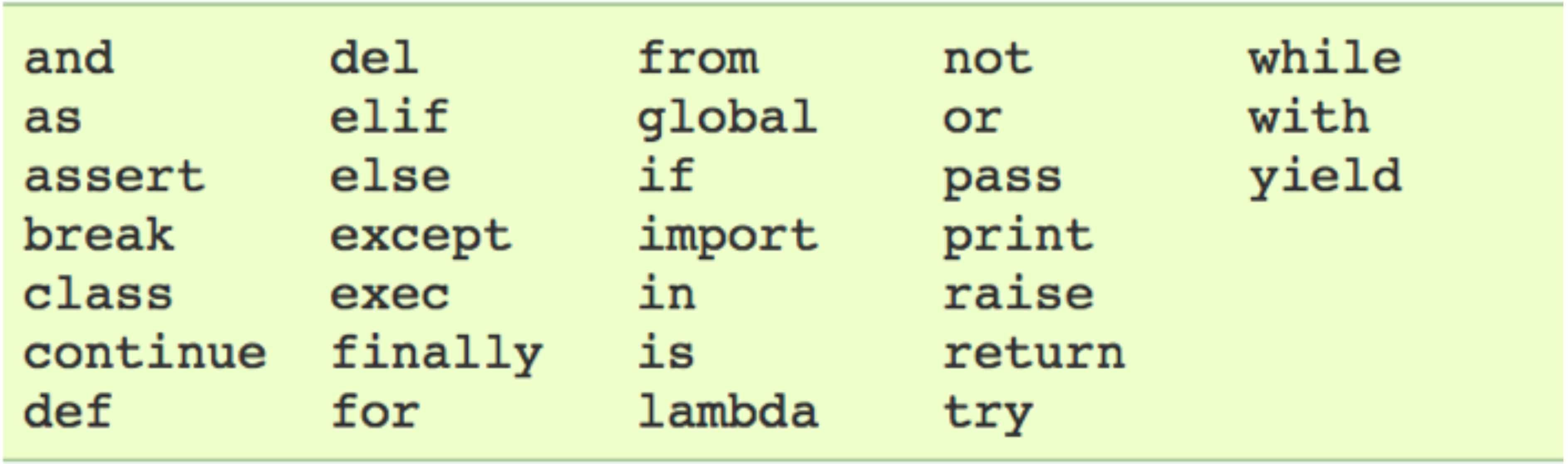
- Everything means everything, including functions and classes (more on this later!)
- Data type is a property of the object and not of the variable

```
>>> x = 7
>>> x
7
>>> x = 'hello'
>>> x
'hello'
>>>
```

variables

Rules of naming a variable:

- Don’t start with numbers
- Don’t use @ or -
- Don’t use reserved words



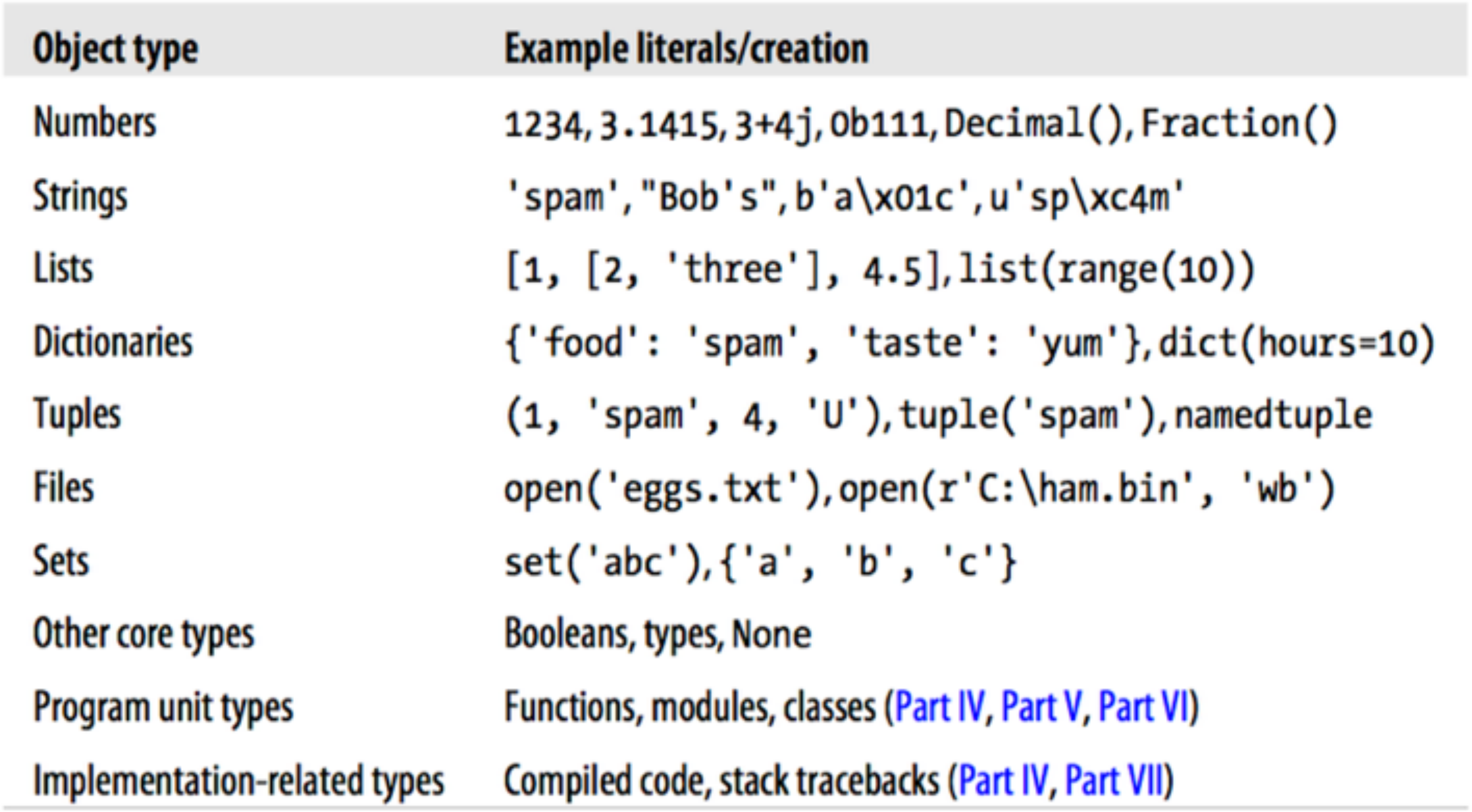
and	del	from	not	while
as	elif	global	or	with
assert	else	if	pass	yield
break	except	import	print	
class	exec	in	raise	
continue	finally	is	return	
def	for	lambda	try	

Practice

Can I use the following variable names?

- 1ab
- ab@a
- aAAA3
- ABDA2
- AND
- for
- For
- a_12A
- b-32D

Object Type Summary



Object type	Example literals/creation
Numbers	`1234`, `3.1415`, `3+4j`, `0b111`, `Decimal()`, `Fraction()`
Strings	`'spam'`, `"Bob's"`, `b'a\x01c'`, `u'sp\xc4m'`
Lists	`[1, [2, 'three'], 4.5]`, `list(range(10))`
Dictionaries	`{'food': 'spam', 'taste': 'yum'}`, `dict(hours=10)`
Tuples	`(1, 'spam', 4, 'U')`, `tuple('spam')`, `namedtuple`
Files	`open('eggs.txt')`, `open(r'C:\ham.bin', 'wb')`
Sets	`set('abc')`, `{'a', 'b', 'c'}`
Other core types	Booleans, types, None
Program unit types	Functions, modules, classes (Part IV, Part V, Part VI)
Implementation-related types	Compiled code, stack tracebacks (Part IV, Part VII)

Numbers: Integers

- Integer – the equivalent of a C long
- Long Integer – an unbounded integer value.

```
>>> 132224
132224
>>> 132323 ** 2
17509376329L
>>>
```

Numbers: Floating Point

- int(x) converts x to an integer
- float(x) converts x to a floating point
- The interpreter shows
  a lot of digits

```
>>> 1.23232
1.2323200000000001
>>> print 1.23232
1.23232
>>> 1.3E7
13000000.0
>>> int(2.0)
2
>>> float(2)
2.0
```

Numbers: Floating Point

- int(10.39)
- int(100.9999)
- int(1001.00001)
- float(87)
- float(eight)

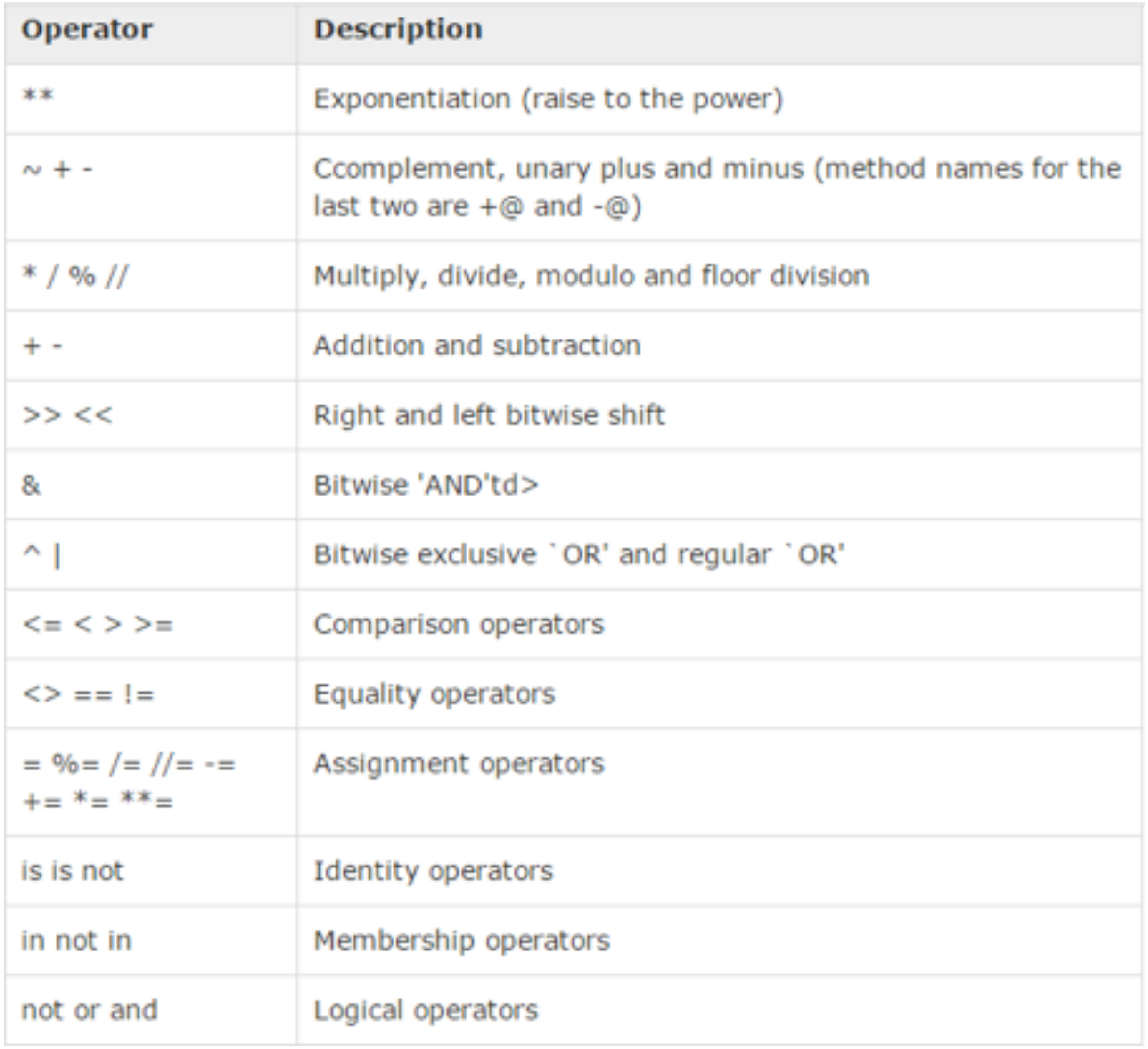
Numbers: Complex

- Built into Python
- Same operations are supported as integer and float

```
>>> x = 3 + 2j
>>> y = -1j
>>> x + y
(3+1j)
>>> x * y
(2-3j)
```

Operators

Operations in Python are based on sign precedence



Operator	Description
**	Exponentiation (raise to the power)
~ + -	Ccomplement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'td>
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Image from: http://
www.tutorialspoint.com/
python/
operators_precedence_exa
mple.htm

Operators

Integer vs float operations

- Integer operation will result in only the “integer” part of the operation
  - 5/3 equals 1

- Float operation will result in the “float” value of the operation
  - 5/3.0 equals 1.66666667
  - 5.0/3 equals 1.66666667
  - 5.0/3.0 equals 1.666666667
- You can fix that by adding the words:

from __future__ import division

  - At the beginning of your code

- Let’s try it together

After class

1. Practice and get familiar with Atom, command prompt
2. Try examples using python, such as Integer, Strings