

Introduction to Python

Part 1

COMPSCI 260
27 / 28 August 2014

Part 1 Topics:

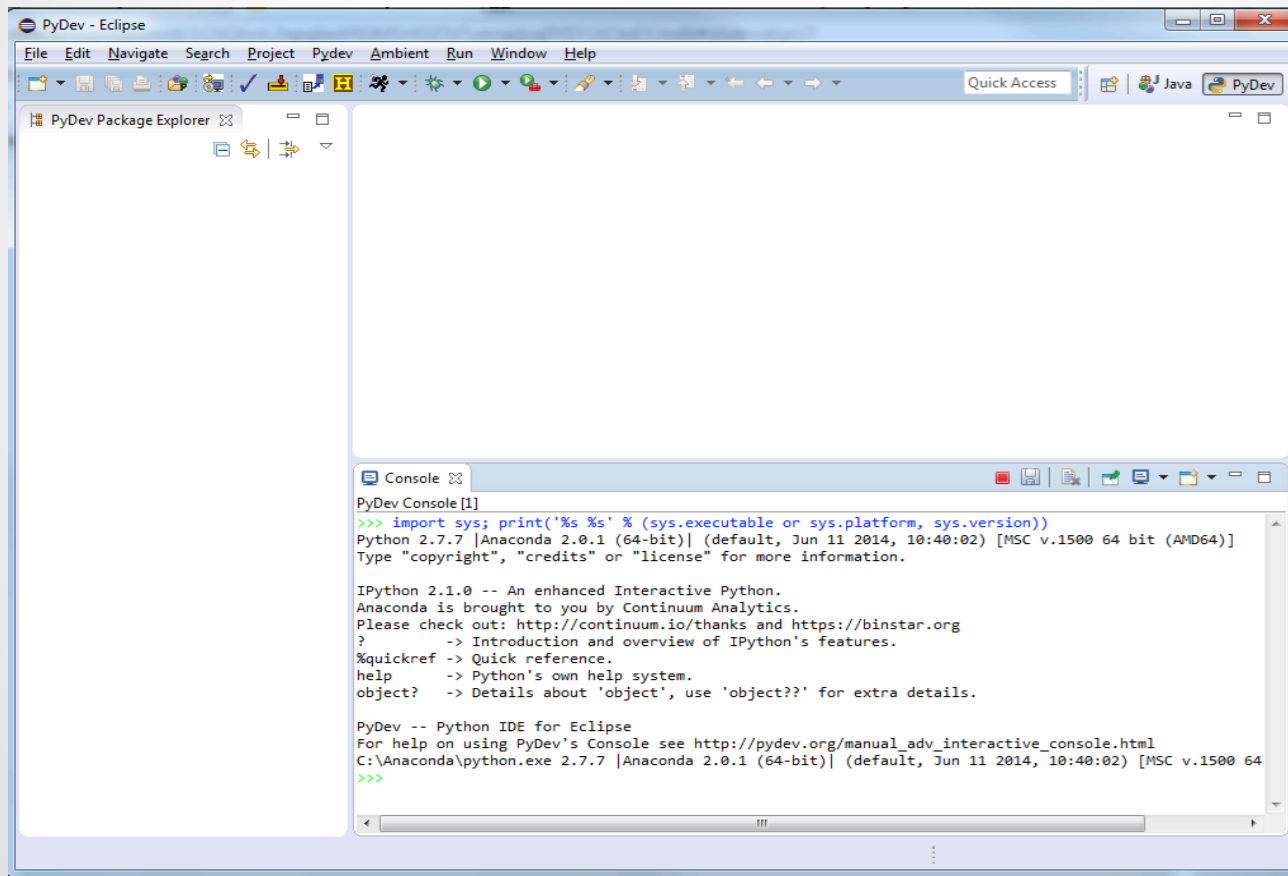
- Python language and how to run it
- Basic data types and syntax
- Control structures
- Important data structures in Python: lists (strings), tuples, and dictionaries
- Function declarations

The Python Programming Language

- Dynamically vs. statically typed
- Automated memory management
- General purpose programming / scripting language
 - Thanks to a ton of modules and libraries
 - <https://pypi.python.org/>

Python Interactive Shell

- Python interactive shell inside Eclipse



```
PyDev - Eclipse
File Edit Navigate Search Project Pydev Ambient Run Window Help
PyDev Package Explorer
Console
PyDev Console [1]
>>> import sys; print('%s %s' % (sys.executable or sys.platform, sys.version))
Python 2.7.7 |Anaconda 2.0.1 (64-bit)| (default, Jun 11 2014, 10:40:02) [MSC v.1500 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 2.1.0 -- An enhanced Interactive Python.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://binstar.org
?          -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.

PyDev -- Python IDE for Eclipse
For help on using PyDev's Console see http://pydev.org/manual_adv_interactive_console.html
C:\Anaconda\python.exe 2.7.7 |Anaconda 2.0.1 (64-bit)| (default, Jun 11 2014, 10:40:02) [MSC v.1500 64
>>>
```

Python on command line

- Collect Python code into a text file
- Save it with `.py` suffix, say `script.py`
- Open a terminal on your machine
- Then type: `python script.py Arg1 Arg2 ...`

Python Data Types & Syntax

- Numerical data types
 - `a = 34` (integer)
 - `a = 34.0` (floating point numbers – single & double precision)
 - `a = True` (boolean)
- Characters are strings of length 1 (More on strings later!)
 - `c = "u"` OR `c = 'u'`
- Converting (casting) from one type to another
 - `int()`, `float()`, `double()`, `boolean()`, `str()`
 - `i = int("3")`
- Check type of variables with `type()`
 - `type(34) = int`

Python Data Types & Syntax (cont'd)

- Whitespace matters!
 - Think curly braces ('{') in C or Java
 - All lines with the same indentation are part of the same block/scope
 - Example:

```
if check == 1:  
    do something...  
else:  
    if check2 == 1:  
        do something...
```

- Comments (start with '#')
 - Example:

```
if check == 1:  
    # comments can be on their own line  
    do something... # or inline with code
```

Mathematical & Logical Operations

- Addition, subtraction, multiplication
 - $2 + 2$, $42 - 6$, $4 * 3$
- Division (type matters)
 - $3 / 2 = ?$
- Modulus
 - $3 \% 2 = 1$
- Exponentiation
 - $3 ** 2 = 9$
- Logical operators
 - and, or, not
 - $==$, $!=$, $<$, $<=$, $>$, $>=$

print Statement

- Places output on the screen

```
i = 34  
print i
```

- Use comma between variables you want printed on the same line (i.e., comma will suppress a newline)

```
print i  
print i,j
```

- Useful for debugging!

Control Structures

- `if-else` statement:

```
if check == 1:  
    do something...  
elif (check == 2) and (not embarrassed):  
    do something...  
else:  
    at least do something...
```

- `while` statement

```
while i < 40:  
    do something
```

Control Structures (cont'd)

- `for` statement

```
for i in [1,2,4]:  
    print i
```

- `break` and `continue`

```
for i in range(50):  
    if (i == 0) or (i == 1):  
        continue
```

```
for i in range(50):  
    if i == 10:  
        break
```

- <http://docs.python.org/2/tutorial/controlflow.html>

Lists

- Creating a list

```
list0 = [] # create empty list manually
```

```
list1 = ['a', 1, 324.3] # create with values
```

```
list2 = list(list1) # creates a copy of list1
```

- Lists are mutable – can be changed in place

```
list1 = ['a', 1, 324.3]  
list1[0] = 'b'  
del list1[1]
```

- Can iterate over lists with for loops

```
for item in list1:  
    print item
```

- List can group many different data types

```
a = [99, 'bottles of beer', ['on', 'the', 'wall']]
```

Lists (cont'd)

- List comprehension

```
[str(x) for x in [1,2,3]]  
→ ['1', '2', '3']
```

- Slicing

```
a = [99, 'bottles of beer', ['on', 'the', 'wall']]  
print a[0:2]  
→ [99, 'bottles of beer']  
print a[1:]  
→ ['bottles of beer', ['on', 'the', 'wall']]
```

- Reverse indexing

```
print a[-1]  
→ ['on', 'the', 'wall']  
print a[-3:-1]  
→ [99, 'bottles of beer']
```

- Delete elements

```
del a[1]  
print a  
→ [99, ['on', 'the', 'wall']]
```

Lists (cont'd)

```
a = [0, 1, 2]
b = [3, 4]
a + b → [0, 1, 2, 3, 4]
a * 3 → [0, 1, 2, 0, 1, 2, 0, 1, 2]
```

```
a.append(5) → [0, 1, 2, 5]
a.pop(1) → [0, 2, 5]
a.insert(1, 42) → [0, 42, 2, 5]
a.reverse() → [5, 2, 42, 0]
a.sort() → [0, 2, 5, 42]
sorted(a) → [0, 2, 5, 42]
```

```
print len(a)
→ 4
```

Strings

- A string is similar to a list of characters

```
a = 'hello'  
print a[0]  
→ 'h'  
print a[1:4]  
→ 'ell'  
print a[-1]  
→ 'o'
```

```
for c in a:  
    print c,  
→ h e l l o
```

- But a string is immutable
 - Test: Try to change a single character in a string variable

```
a[0] = 'j'
```

Strings

- To create a string:

```
strvar1 = 'abc'  
strvar2 = str(123) # can cast objects as strings
```

```
strvar5 = ''  
strvar5 += 'cr'  
strvar5 += 'ude' # concatenation
```

- String formatting

```
# using string formatting  
strvar3 = 'Pi is about %.4f' % 3.142951  
→ 'Pi is about 3.1430'  
  
# more formatted strings  
strvar4 = '%s Student #%d!' % ('Hello', 42)  
→ 'Hello Student #42!'
```


String Operations

```
'hello' + 'world' → 'helloworld' # concatenation
```

```
'hello' * 3 → 'hellohellohello' # repetition
```

```
'hello'[::-1] → 'olleh' # reversing by slice
```

```
len('hello') → 5 # size
```

```
'hello' < 'jello' → 1 # comparison
```

```
'e' in 'hello' → True # membership
```

```
'hello'.find('lo') → 3 # finding substrings
```

```
'hello_world'.count('o') → 2 # counting substrings
```

```
# splitting strings
```

```
'hello_world'.split('_') → ['hello', 'world']
```

```
# remove whitespace
```

```
'hello_world \n'.strip() → 'hello_world'
```

Practice Time:

- Use 15 minutes to practice with Part1.py in Tutorial 1
- Use 15 minutes to practice with Part2.py in Tutorial 1

Tuples

- Create a tuple

```
tup = ()  
tup = ('32', 4, 'yes', 3.14)
```

- Quite similar to a list

```
tup[1:4]  
→ (4, 'yes', 3.14)
```

- But tuples are immutable

```
tup[1] = 12 → error
```

- <http://docs.python.org/2/library/functions.html#tuple>

Dictionaries

- Dictionary are "associative arrays" mapping keys to values: {key: value}

```
d = {  
    'Marky': 'Mark', 'Funky': 'Bunch', 3: '4', (1,2):[1,2,3]  
}
```

- Dictionary assignment & operations

```
print d.keys()    → [(1, 2), 'Funky', 3, 'Marky']  
print d.values() → [[1, 2, 3], 'Bunch', '4', 'Mark']  
print d.items()  
→ [((1, 2), [1, 2, 3]), ('Funky', 'Bunch'), (3, '4'),  
    ('Marky', 'Mark')] # a list of key-value pairs in tuples
```

```
print d['Marky'] → 'Mark'  
d['Donnie']     → error # raises KeyError exception  
d['Donnie'] = 'Wahlberg' # value assignment
```

```
# check for presence of key  
d.has_key('Donnie')  
# item deletion  
del d['Marky']
```

```
# iterate over keys  
for dkey in d:  
    print dkey
```

Practice Time:

- Use 15 minutes to practice with Part3.py in Tutorial 1

Function Declarations

```
def func1(arg1, arg2):  
    function statements  
    return val    # Optional
```

- Functions are pass-by-(object)-reference

```
def f1(a):  
    a.append(4)  
  
b = [1,2,3]  
f1(b)
```

```
def f2(a):  
    a = 4  
  
b = [1,2,3]  
f2(b)
```

Pointers (or Reference vs. Copy)

- Suppose you perform the following:

```
list1 = [1,3,4]
list2 = list1
list3 = list(list1)
list3.append(6)
list2.append(5)
```

- What are list1, list2, and list3 now?
- Be careful not to confuse references with copies
 - Use casting functions like `str()`, `list()` when you need to make copies
 - For dictionaries, use `copy()` and `deepcopy()`

Practice Time:

- Use 20 minutes to practice with Part4.py in Tutorial 1