

# Help & introspection, the structure of programming languages, strings

Week 1, Day 2

# Getting help

- Online at <http://python.org/docs>
- Online at <http://ipython.org>
- Python reference
- Introspection - examining objects - during a live session (both Python & IPython)
- Magic words (IPython only)

# Help & introspection

- `help()` - interpreters interactive help
- `help(obj)` - documentation for *obj*
- `dir(obj)` - get attributes of *obj*
- `?obj` or `obj?` - examination of *obj*
- `??obj` or `obj??` - detailed examination of *obj*, including source code if available (IPython only)

# IPython

- `?` - list special features of IPython
- `magic` - list “magic words” (even more special features)
- `ls` and `cd dir` - list and change directory
- `edit file` - use console editor on *file*
- `run file args` - execute the file with the given commandline arguments
- `pdef obj` and `pdoc obj` and `psource obj` - get the definition, documentation or source code for an object

# What does a programming language need?

- Variables & assignment
- Operators & expressions
- Statements
- Input & output

```
x = 3
```

```
y = "genbank"
```

```
z = x * y
```

```
theName = raw_input ("what  
is your name? ")
```

```
print "hello world"
```

# What does a programming language need? (2)

- Conditionals
- Booleans
- Loops
- Blocks
- Comments

```
if (x < 5):  
    print "string too short"  
  
# False, [], (), {}, 0, ""  
# True, everything else  
  
for i in range (10):  
    print i  
while i < 10:  
    print
```

# What does a programming language need? (3)

- data structures & containers
- strings
- functions
- modules

```
theList = [1, 7, 10]
theTuple = (1, 7, 10)
theDict = {1: 'a', 2: 'b'}

theString = "abcde"

def convertSeq (in_seq):
    return in_seq.lower()

import cStringIO
```

# Strings

- Text, of any length
- Sequence
- Can use some operators
- Immutable

```
str1 = 'accggt'
str2 = """gctata"""

for x in str1:
    print x

for i in range (len (str2)):
    print str2[i]

str3 = str1 + str2 * 3

str[3] = "t"    # error!
```

# Many “types” of strings

- Mainly just different ways of writing text
- `'str'`, `"str"` - single quoted
- `"""str"""`, `'''str'''` - triple quoted, can span lines
- `r'str'` - raw, literal
- `u'str'` - unicode, an actual type for different character sets

# Examples

```
>>> single_str = "\n    ac\tccgt\n"
>>> triple_str = '''
    ac\tccgt
'''
>>> print single_str

    ac    ccgt

>>> raw_str = r"\n    ac\tccgt\n"
>>> print raw_str
"\n    ac\tccgt\n"
```

# Strings as sequences

- Length
- Indices
- Slices
- Membership

```
>>> str1 = 'acccgt'  
>>> len (str1)  
6  
>>> str1[0]  
a  
>>> str1[-1]  
t  
>>> str1[3:]  
cgt  
>>> str1[:3]  
acc  
>>> str1[::-1]  
tgccca  
>>> 'a' in str1  
True
```

# String methods

```
>>> name = 'Swaroop'  
>>> name.startswith('Swa')  
True  
>>> name.find('war')  
1  
>>> name.find('zoo')  
-1  
>>> delimiter = '_ * _'  
>>> mylist = ['Brazil', 'Russia', 'India', 'China']  
>>> delimiter.join(mylist)  
Brazil_ * _Russia_ * _India_ * _China  
>>> name.upper()  
SWAROOP  
>>> name.replace('o', 'xx')  
Swarxxxxp  
>>> name[0].isdigit()  
False
```

# strings and string

- historical leftover
- counterparts of most string methods
- useful lists

```
>>> name = "Swaroop"  
>>> import string  
  
>>> name.count ('o')  
2  
>>> string.count (name, 'o')  
2  
  
>>> string.lowercase  
'abcdefghijklmnopqrstuvwxyz'  
>>> string.digits  
'0123456789'
```

# strings and string

- historical leftover
- counterparts of most string methods
- useful lists

```
>>> name = "Swaroop"  
>>> import string  
  
>>> name.count ('o')  
2  
>>> string.count (name, 'o')  
2  
  
>>> string.lowercase  
'abcdefghijklmnopqrstuvwxyz'  
>>> string.digits  
'0123456789'
```

# Formatting strings

```
>>> x = 3.5
>>> str(x)
'3.5'
>>> "x equals " + str(x)
'x equals 3.5'

>>> "x equals %s" % x
'x equals 3.5'
>>> "%s times %s is %s" % (3, 4, 3*4)
'3 times 4 is 12'
>>> "the length of %s is %s" % (st, len(st))
'the length of Swaroop is 7'
>>> "list %s has %s items" % (ll, len(ll))
'list [1, 2, 3] has 3 items'
```

# Formatting with dictionaries & codes

```
>>> "the length of %(word) is %(ln)s" %  
{'word': st, 'ln': len(st)}  
'the length of Swaroop is 7'
```

```
>>> "%(x)s times %(y)s is %(prod)s" % {'y':  
4, 'x':3, 'prod': 3*4}
```

```
>>> "%d, %i are integers, %3f and %3.2g" %  
(12, 13, 129.4, 28.90001)
```

# String efficiency

- Adding strings is slow (need to create each string)
- Use `format` where possible
- Use `StringIO` & `cStringIO` for fi"file-like: strings

# Resources

- 300+ Python tutorials at <http://awaretek.com/tutorials.html>
- Python documentation at <http://www.python.org/doc/>
- IPython home at <http://ipython.org>
- Free book “Text processing with Python”