

IPython

The Python interpreter and aids to development

How can Python be called?

- On the commandline
- Direct execution of script
- Importing
- The interactive prompt

On the commandline

Can define script and interpreter:

```
home> python extract_concensus.py
home> python23 extract_concensus.py
home> /usr/bin/local/python23 extract_concensus.py
home> /usr/bin/local/python23
      projects/biopy/extract_concensus.py
```

-V for version

Execute scripts directly

(In Unix) make file executable with:

```
chmod +x myscript.py
```

First line of script is:

```
#!/usr/bin/env python
```

(or `#!/usr/bin/python` etc.)

Just like calling `python myscript.py` etc.

Raw code on the commandline

- Pass code straight to Python
- Delimit statements with ;
- Use as a calculator

- For example:

```
home> python -c "x=3; y=4; print x*y"
12
```

What is `__main__`?

- The script called defines its name as `__main__`
- Can use to make script behave two ways:

```
if __name__ == "__main__":
    print "Running as main"
else:
    print "Not main, probably imported"
```

Commandline arguments

Commandline args trapped in `sys.argv`:

```
if __name__ == "__main__":
    import sys
    for item in sys.argv:
        print "Argument:", item
```

Use `__main__` for testing?

Counting args

Watch the spacing:

```
# sys.argv contains 4 arguments
checkargs.py inseq.fasta true strip_spaces=true

# sys.argv contains 4 arguments
python checkargs.py inseq.fasta true strip_spaces=true

# sys.argv contains 6 arguments (use quotes)
checkargs.py inseq.fasta true strip_spaces = true
```

The interactive prompt

- Call with `python`
- Can write and manipulate code on the fly (advantage of interpreted language)
- `python -i myscript.py` runs the script and then dumps you into the prompt
- Awkward (IPython!)

Programming features

`help()` Inbuilt help system, mostly for built-ins

`dir (obj)` Gives object attributes

`type (obj)` Type of object

`reload (module)` Force re-import

IPython

- `ipython`
- An enhanced Python prompt
- Access to OS facilities
- Object introspection
- Run scripts
- Tab completion

OS

Change working (current) directories with `cd`, list directory with `ls`, find where you are with `pwd`:

```
Documents/Projects/MyPython/Casimir> python
>>> import xmltools
ImportError: no module named xmltools
>>> import Documents/Projects/MyPython/xmltools
SyntaxError: invalid syntax
>>> import "Documents/Projects/MyPython/xmltools"
SyntaxError: invalid syntax
```

Tab completion

Complete (loaded) object names:

```
In [1]: o<TAB>
or      oct      ord      open      object
In [2]: import os
In [3]: o<TAB>
or      oct      ord      open      object      os
In [2]: os.ch<TAB>
os.chdir  os.chmod
```

Introspection

Get objects to tell us about themselves:

```
>>> import string
>>> string?
[ ... summary, documentation ...]
>>> string??
[ ... the source ...]
```

And much more

- Logging
- Easier debugging
- Running scripts:

```
>>> run my_script.py
[...]
```

Resources

- IPython <http://ipython.scipy.org/moin/>
- IPython wiki <http://ipython.scipy.org/moin/About>
- O'Reilly on IPython <http://www.onlamp.com/pub/a/python/2005/01/27/ipython.html>
- Newsforge on IPython <http://programming.newsforge.com/article.pl?sid=05/08/29/1638201>

Advanced Python, NBN 2007. P-M Agapow (agapow@bbsrc.ac.uk)