Python Scripting 3

657.019 Scripting for Biotechnologists (WS 2018/19)

Objectives

- Part 1: To learn how to
  - Work with text
  - Work with files
  - Work with lists
  - Use loops
- Part 2: To learn how to
  - Write my own functions
  - Use conditionals
  - Work with dictionaries
- Part 3: To learn how to
  - Interact with operating systems
  - Use basic Biopython

Interacting with operating system

- Modules to import as necessary
  - `sys`
  - `subprocess`
  - `os`
  - `shutil` (shell utilities)

Getting command line arguments

- Can give any number of arguments to a Python script
  - `sys.argv` is a list containing script name and arguments

- Command line: `myscript.py arg1 arg2`

```
import sys
print(sys.argv)     # ['myscript.py','arg1','arg2']
print(sys.argv[0]) # myscript.py
print(sys.argv[1]) # arg1
print(sys.argv[2]) # arg2
```

Running external programs

- Run external program from within Python

```
import subprocess
subprocess.call("uname") # runs uname command
```

- Use command line options

```
subprocess.call("uname -a", shell=True)
```

- Save output

```
os_name = subprocess.check_output("uname -a")
```

File manipulation

- List files and directories

```
import os
os.listdir()
```

- Check for existence

```
os.path.exists("/export/home/jsoh/xxxx.tsv")
```

- Copy files and directories

```
import shutils
shutils.copy("file.txt", "file2.txt")
shutils.copytree("dir", "dir2")
```

- Make directory

```
os.mkdir("newdir")
```

More on file manipulation

- Rename (works like Linux **mv** command)

```
os.rename("oldname.txt", "newname.txt")
os.rename("olddir", "newdir")
```

- Remove file (like **rm**)

```
os.remove("temp.txt")
```

- Remove empty directory (like **rmdir**)

```
os.rmdir("tempdir")
```

- Remove non-empty directory (like **rm -r**)

```
shutil.rmtree("olddir")
```

Biopython

- A set of freely available tools for biological computation written in Python

- Dealing with assorted sequence file formats is easy in Biopython
  - Primary module is `Bio.SeqIO` module: standard Sequence Input/Output interface

Sequence input

- Main function: **`SeqIO.parse()`**
  - Takes a file name (or file handle) and format name, and returns a **`SeqRecord`** iterator

- Read sequences using a **`Seq.Record`** iterator

```
from Bio import SeqIO
for record in SeqIO.parse("example.fasta", "fasta"):
  print(record.id) # print sequence ID
```

Sequence input

- Save sequences as a list for random access

```
from Bio import SeqIO
rec_list = list(SeqIO.parse("example.fasta", "fasta"))
print(rec_list[0].id)  # first sequence ID
print(rec_list[-1].id) # last sequence ID
```

- Access sequences by ID

```
from Bio import SeqIO
rec_dict = SeqIO.index("example.fasta", "fasta")
print(rec_dict["gi:1234"]) # use ID to get record
```

Sequence input

- Get a single sequence, when file has one (and only one) sequence

```
from Bio import SeqIO
record1 = SeqIO.read("example.fasta", "fasta")
```

- Get a single sequence, when file has multiple sequences

```
from Bio import SeqIO
record1 = SeqIO.parse("example.fasta", "fasta").next()
```

Sequence output

- Read sequences and write them in a different format

```
from Bio import SeqIO
input_filename = "example.gbk"
output_filename = "example.fasta"
records = SeqIO.parse(input_filename, "gb")
SeqIO.write(records, output_filename, "fasta")
```

- Read a single sequence and write it in a different format

```
from Bio import SeqIO
input_filename = "example.gbk"
output_filename = "example.fasta"
record1 = SeqIO.read(input_filename, "gb")
SeqIO.write(record1, output_filename, "fasta")
```

Sequence file format conversion

- Use a shortcut combining **`SeqIO.parse()`** and **`SeqIO.write()`**

```
from Bio import SeqIO
input_filename = "example.gbk"
output_filename = "example.fasta”
# convert gb to fasta
# returns number of records converted
count = SeqIO.convert(input_filename, "gb",
output_filename, "fasta")
```