π

Welcome back

Python course 2016 - Week 2 - Day 4

Lucas Sinclair
Alexander Eiler
Ludovic Dutoit

Files

Live demo

Files

```
handle = open('test.txt')
type(handle)
handle.name
handle.read?
handle.read(100)
handle.read()
handle.write('a')
x = 4
for elem in x: print elem
x = [1,2,2,3]
for elem in x: print elem
x = "iuoiuoiu"
for elem in x: print elem
for elem in handle: print elem
handle.seek(0)
for elem in handle: print elem
handle = open('sari_pfam_list.rtf', 'w')
handle.write("aaa")
```

Files summary

1. open() returns an object of type "file" which has nice methods.
2. "file" objects are iterable.
3. open(path, 'w') deletes a file completely, be careful.

Modules: sys and os

Live demo

Modules: sys and os

```
os.chdir(path)
os.mkdir('temp')
os.makedirs(/tmp/temp/temp")
os.rmdir('/tmp/temp')
os.rename()
os.walk()
os.environ


sys.platform
sys.version
sys.argv
sys.stderr
sys.stdout
sys.maxint
sys.exit()
```

sys and os summary

1. "os" is for interacting with the filesystem mostly as well as some operating system functions.
2. "sys" is for requesting system specific parameters.

Exercise Day 2 - #3

Parsing and data structure

A file distance.dat is given. Each of its line contains the name of two cities and the distance between them; these quantities are separated by spaces or tabsWrite a python program that parses this file and builds a data structure in memory to contain the result. It's important to chose the right data structure for the task at hand. Would you use a list ? A tuple ? A list of lists ? A dictionary ? A set ? A tuple indexed dictionary ? A dictionary of dictionaries ?Using this data structure, compute the total distance of a given tour, specified in a tuple tour containing, in order, the names of the cities to visit.

Solution

```python
import os
# Set the path #
in_path = "distance.dat"
```

Distance.dat

```
Boston   New-York 301
New-York Geneva   6220
New-York Paris    5838
Geneva   Boston   5918
Boston   Paris    5536
Paris    Geneva   411
```

Solution

```python
import os
# Set the path #
in_path = "distance.dat"
# Check that the path is valid #
if not os.path.exists(in_path):
    raise Exception("No file at '%s'." % in_path)
```

Solution

```python
import os
# Set the path #
in_path = "distance.dat"
# Check that the path is valid #
if not os.path.exists(in_path):
    raise Exception("No file at '%s'." % in_path)
# The file contains cities and distances tab separated #
parsed = (l.strip('\n').split('\t') for l in open(in_path))
```

Solution

```python
import os
# Set the path #
in_path = "distance.dat"
# Check that the path is valid #
if not os.path.exists(in_path):
    raise Exception("No file at '%s'." % in_path)
# The file contains cities and distances tab separated #
parsed = (l.strip('\n').split('\t') for l in open(in_path))
# Make a dictionary with tuples as keys #
d = {(cityA,cityB):int(dist) for cityA, cityB, dist in parsed}
```

Solution

```python
import os
# Set the path #
in_path = "distance.dat"
# Check that the path is valid #
if not os.path.exists(in_path):
    raise Exception("No file at '%s'." % in_path)
# The file contains cities and distances tab separated #
parsed = (l.strip('\n').split('\t') for l in open(in_path))
# Make a dictionary with tuples as keys #
d = {(cityA,cityB):int(dist) for cityA, cityB, dist in parsed}
# We need distances in both directions: flip the keys ! #
d.update({(key[1],key[0]):val for key,val in d.items()})
```

Solution

```python
import os
# Set the path #
in_path = "distance.dat"
# Check that the path is valid #
if not os.path.exists(in_path):
    raise Exception("No file at '%s'." % in_path)
# The file contains cities and distances tab separated #
parsed = (l.strip('\n').split('\t') for l in open(in_path))
# Make a dictionary with tuples as keys #
d = {(cityA,cityB):int(dist) for cityA, cityB, dist in parsed}
# We need distances in both directions: flip the keys ! #
d.update({(key[1],key[0]):val for key,val in d.items()})
# Test tour #
test_tour = ('Boston', 'New-York', 'Paris', 'Geneva')


```

Solution

```python
import os
# Set the path #
in_path = "distance.dat"
# Check that the path is valid #
if not os.path.exists(in_path):
    raise Exception("No file at '%s'." % in_path)
# The file contains cities and distances tab separated #
parsed = (l.strip('\n').split('\t') for l in open(in_path))
# Make a dictionary with tuples as keys #
d = {(cityA,cityB):int(dist) for cityA, cityB, dist in parsed}
# We need distances in both directions: flip the keys ! #
d.update({(key[1],key[0]):val for key,val in d.items()})
# Test tour #
test_tour = ('Boston', 'New-York', 'Paris', 'Geneva')
# Print result #
print "Total distance: %i" % compute_distance(d, test_tour)
```

Solution

```python

def compute_distance(all_distances, tour):
    """Given the *all_distances* dictionary, will computes the
total distance in the city tour *tour*.
    return sum((all_distances[tour[i],tour[i+1]] for i in
range(len(tour)-1)))



# Test tour #
test_tour = ('Boston', 'New-York', 'Paris', 'Geneva')
# Print result #
print "Total distance: %i" % compute_distance(d, test_tour)
```

Exercise Day 2 - #5

Combining files

A collaborator gives you 3 different files. One with the chromosome names (chrom.txt), one with the start and end of fragments (start_end.txt) and one with the genes corresponding to it (gene.txt). Write a python script that will reconstruct one BED formatted file with these 3 files.

Solution

```python
import os
# Get the shell arguments #
in_paths = ("chrom.txt", "start_end.txt", "gene.txt")
out_path = "combined.txt"
# Check that the paths are all valid #
if not all((os.path.exists(p) for p in in_paths)):
    raise Exception("No file at %s." % p)
# Will hold the contents of the files #
contents = [[l.strip('\n') for l in open(p)] for p in in_paths]
# Combine them in #
combined = zip(contents[0], contents[1], contents[2])
# Add the tabs and new lines #
serialized = [' '.join(l) + '\n' for l in combined]
# Write the thing back to disk #
with open(out_path, 'w') as f: f.writelines(serialized)
```

15 minutes break