

Practice 4.1

Practice sheets are not assessed. The intention is to use material from lectures in preparation for Competence tests and Assignments. You are encouraged to use `thing.[TAB]` and `thing?` in IPython.

Multiprocessing

The following imports the `time` function from the `time` module. (Yes, it's annoying that these have the same name.)

```
from time import time
```

The `time` function returns the number of seconds since the UNIX epoch, Thursday 1st January 1970 UTC.

```
time()
```

```
1575376666.984755
```

This can be used to see how long something takes to run. e.g.

```
def factors(N):
    return [n for n in range(2,N//2) if N%n == 0]
```

```
start_time = time()
numbers = factors(10000000)
end_time = time()
duration = end_time - start_time
duration
```

```
0.24155521392822266
```

Use multiprocessing to factor the following numbers and record how long it takes.

```
Ns = [327668137, 293041039, 302952547, 316772299, 323579981, 286725589, 313867091, 360613859,
      307593911, 334270493, 327857639, 369562007, 344286581, 288065551, 313064867, 306248237]
```

You can adapt the following code snippet:

```
from multiprocessing import Pool

def f(n):
    return n**2

if __name__=='__main__': # this is encouraged in general and necessary on Windows
    with Pool() as p:
        out = p.map(f, range(10))
```

(For a discussion of `__name__` etc see e.g. [here](#) or docs.)

Compare this with the time taken *without* using multiprocessing. Can you make it go any faster?

Classes

Make a Python class to represent a cat. Initialise the object with whatever properties you think are important about a cat (perhaps including some sensible defaults), and be sure to include the mass.

Make `__str__` display a human-readable description of the cat: if the mass is greater than 100 kg, tell the user it's a lion; if the mass is less than 10^{-20} kg, tell the user that this is Schrödinger's cat.

Make a list of cats of varying masses.