

More python



Some more advanced features of python:

1. Lambda Functions
2. Advanced List Processing
3. List Comprehension
4. Object Orientated Programming (OOP)
5. Additional Libraries

Lambda functions

- Short single line functions with fixed list of variables, typically used to tidy up code. For example:

```
import math

def main():
    length = lambda x,y,z: math.sqrt(x*x + y*y + z*z)
    distance = length(3.0,4.0,5.0)
    print(distance)

main()
```

- Allows you to create quick, easy to call and numerically efficient functions that are LOCAL to where they are created.

Advanced list processing

- Series of built-in functions that process whole lists at a time:
 - `map` applies a function to each element of a list
 - `filter` applies a logical operation to each element of a list
 - `reduce` processes list elements recursively in pairs
 - `zip` combines two lists into one list with two elements per element
- These functions provide a very efficient way of processing large lists
- They also implement the key functions of Functional Programming
- Typically used to pre-process data, for example when reading in or writing out; they are less useful for numerical calculations

List comprehension

- List comprehension allows you to put flow control within the definition of lists. For example:

```
vals = [x**2 for x in range(0,30)]
```

- This will produce a list of elements x^2 for $x = 0 \rightarrow 29$ with a single statement.
- This syntax is unique to Python. It is very efficient, but can produce very cryptic code
- You will find many examples on the Web but suggest you don't use this approach until you are confident you can program well!

Object oriented programming (OOP)

- Create objects with internal variables and methods
 - generally considered a ‘modern way’ to write code
- Similar syntax to the “pure” OOP languages (Java and C++), a good technique to learn for future skills
 - See online notes for basic example (based on a 3D vector)
 - See (non-assessed) Checkpoint 6 for a more advanced example
- For more on OOP, see optional Computer Simulation next semester

Additional libraries

- The real power of Python for scientific problems lies in the extensive, efficient, add-on libraries.
 - **Matplotlib**: the library you have been using to plot graphs. Very powerful, all forms of graphs, animation, 3-d plots, image display, display of maps, etc.
 - **NumPy**: the main numerical library. N-dimensional arrays, efficient vector / matrix manipulation, linear algebra operations, Fourier transforms, integration with databases
 - **SciPy**: advanced numerical algorithm library, numerical integration, optimisation, special functions, statistics, signal processing, images manipulation
 - **SymPy**: computer algebra package, (symbolic mathematical manipulation)
- ...and many, many (thousands) more
- These are typically written in an efficient (fast) language (typically C) and allow fast processing, access to multi-core / multi processor programming and “big data” from Python