(PRACTICAL) COMPUTATIONAL PHYSICS

Physics 551
Lecture 9

Extra Reading

Optional Exercise

Recommended

NOTATION

- This lecture slides for this course will attempt to use a uniform notation throughout. A normal paragraph looks like this.

✒ *Italicized paragraphs with pen bullets will indicate definitions, with the defined word or phrase shown in* ***SMALL-CAPS****.*

✏ Pencil bullets will indicate the introduction of **new notation**.

☞ Pointing hand bullets indicate important points that might otherwise be overlooked.

ANNOUNCEMENTS

ANNOUNCEMENTS

- To clone this week's **C++ demonstration materials** please invoke

```
$git clone https://github.com/hughdickinson/CompPhysL9CPP.git
 /home/computationalphysics/Documents/cPlusPlus/lecture9
```

- To clone this week's **Python demonstration materials** please invoke

```
$git clone https://github.com/hughdickinson/CompPhysL9Python.git
 /home/computationalphysics/Documents/python/lecture9
```

☞You can also find these commands on the Blackboard Learn website.

ANNOUNCEMENTS

- There is some new software to install **on VirtualBox**.
- The **doxygen** utility automatically documents C++ source code using the comments it contains. Install **doxygen** using

```
$ sudo apt-get install doxygen
```

- The **doxygen** utility **is able to** make use of the **$\LaTeX$** typesetting utility, which **may require several hours to install** and can be obtained using

```
$ sudo apt-get install texlive
```

ANNOUNCEMENTS

Midterm Survey Results

- Only **one student** did **not** complete the survey
- Everyone who did complete the survey indicated that they did **not** want a special midterm project to be set.
- The **final project** will contribute **30%** of the final grade for the class.
- The **weekly homework problems** will contribute **70%** of the final grade for the class.

CLARIFICATIONS

PYTHON TUPLE LITERALS

- Python `tuple`-type literals are specified as a **parenthesized**, **comma-separated** list of elements.

```
    literalTuple = (2.0, anotherVar, 'cheese')
```

- For **non-empty** `tuple`s, the parentheses are **optional**.
- For **single-element** `tuple`s, a **comma is mandatory**.

```
  singleElementTuple = (100,) # type is tuple
```

- Without it the expression is interpreted as a **parenthesized object** with an appropriately inferred type.

```
      parenthesizedInt = (100) # type is int
```

C++

AUTOMATIC DOCUMENTATION

- The **extensive online documentation** that is typically provided by libraries such as the GSL **substantially** facilitates their usage.
- You can help collaborators to make better use of the software that **you write** by providing similar **HTML documentation**.

☞ The **doxygen** utility can be used to **automatically generate** HTML documentation using the **comments** that annotate your code.

```
$ man doxygen
```

ADOPTING DOXYGEN

The **doxygen** homepage: http://www.stack.nl/~dimitri/doxygen/

☞ The **doxygen** utility searches for **specially formatted comments** within source code files and uses them to **automatically** generate documentation.

- To expose multiline comment blocks to the **doxygen** utility, add an extra "*****" character to the **opening** "/*" token i.e.

  "/*" → "/**".

- To expose single-line comments to the doxygen utility, add an extra "/" character to the opening "//" token i.e.

  "//" → "///"

ADOPTING DOXYGEN

For more information about **doxygen syntax**, consult:
The doxygen quick reference

- The **doxygen** utility also interprets a several **special tokens** that control the formatting of the documentation that is generated.
- For example, a **class identifier** can be specified by preceding it with a “`\class`” token.
- Typeface formatting is also possible. For example, **emboldened words** should be preceded by a “**\b**” token.

☞ **LaTeX-formatted equations** can also be provided between paired “`\f`” tokens.

ADOPTING DOXYGEN

- The **doxygen** utility is controlled using a **configuration file**. To generate a **template** configuration file, invoke

```
$ doxygen -g templateFilePath
```

- Editing the configuration file enables **customization** of the **directory hierarchies** and **file types** that the **doxygen** utility should scan, as well as the **formats** and **paths** of the generated documentation.
- To generate documentation using **doxygen** with a configuration file called *`configFilePath`*, invoke

```
$ doxygen configFilePath
```

DEMONSTRATION

Automatic documentation using **doxygen**

Clone the C++ demonstration material from GitHub:

```
$ git clone https://github.com/hughdickinson/CompPhysL9CPP.git
 /home/computationalphysics/Documents/cPlusPlus/lecture9
```

PYTHON

REVIEW - LIST GENERATION

- The Python `range(start, stop, step)` **requires** `start`, `stop` and `step` to be **integer** values.
- The function generates a Python `list` of **integer** all values between `start` and `stop−1` in increments of `step`.
- The `range(…)` function is often used to provide a list of values for iteration in the opening clauses of `for`-loops.

☞ The `start` and `step` arguments are **optional**. If they are omitted, `start` defaults to `0` and `step` defaults to `1`.

REVIEW - SEQUENCE SLICING

- *Specific subsections of Python sequence types can be extracted using the **Slicing** syntax.*
- Schematically, the statement

```
sequenceIdentifier[start:stop:step]
```

  extracts elements from `sequenceIdentifier` between `start` and `stop–1` in increments of `step`.
- If `step` is **omitted**, a default of `1` is assumed.
- If `start` or `stop` are **omitted**, defaults of the **first** and **last** elements in the sequence are assumed.

REVIEW - SETS

- The Python `set` type provides a **heterogeneous**, **unordered** collection of **unique** elements.
- Literal `set`s are specified as comma separated lists of elements between paired braces e.g. `setVar = { 'two', 3.0, 4 }`
- Uniqueness is established using the **value** of the element. The **inferred type** is **not** important.
- Sets are designed to model abstract mathematical sets and provide methods to perform **set-theoretical** calculations e.g. `union()`, `intersection()` etc.

REVIEW - DICTIONARIES

- The Python `dict` type provides a **heterogeneous**, **unordered**, associative container that defines a mapping between **unique *keys*** and **arbitrary *elements***.
- Literal `dict`s are specified as comma separated lists of colon-separated **key**-**value** pairs between paired braces e.g. `dictVar = { 'two':2, 3.0:2L, 4:2.0 }`
- Although `dict`s are **unordered**, specific element **values** can be "indexed" via their associated keys e.g. `dictVar['two']`

PYTHON SCRIPTS

☞ It is possible to "invoke" a **file** that contains Python source code.

- To do so, invoke the Python **interpreter** and supply the **path to the file** as an **argument**.

```
$ python pythonSourceCode.py
```

☞ By **convention**, files that contain Python source code are assigned the suffix "**.py**".

- ***IPython Notebook*** can **export** Python source code files. Select: *File*→*Download As*→*Python (.py)*, from the Notebook interface.

MODULES

See https://docs.python.org/2/tutorial/modules.html for details.

- *The Python language provides **MODULES** as a mechanism for **unifying** multiple Python entities e.g. **variables**, **functions** or **classes** within a **common namespace**.*
- *Modules and the entities they contain can be **IMPORTED** and referenced or invoked by other Python code, facilitating code reuse.*
- Practically, **implementing a module** entails **aggregation** of Python source code into a single file with a "***.py***" suffix.
- The **name** of the module file corresponds to the **identifier** that can be used to **import** the module and reference or invoke its entities (or ***attributes***) from other Python source code.

PACKAGES

See https://docs.python.org/2/tutorial/modules.html#packages.

- ***PACKAGES** are used to **organize** modules within a higher-level namespace.*
- Practically, a package is a **directory** containing **one or more** Python module files and a **special file** called "***`__init__.py`***".
- The "***`__init__.py`***" file could be **empty** but may also contain **initialization** code that applies to **all** modules in the package.
- The **name** of the package directory file corresponds to the **identifier** that can be used **qualify** imports of the modules it contains.

MODULES AS SCRIPTS

See https://docs.python.org/2/tutorial/modules.html#executing-modules-as-scripts for details.

- Python modules are simply files containing Python source code. In principle, module files can be invoked as Python scripts.
- Any **preliminary setup code** that the module requires in order to execute as a standalone script can be provided in the body of an `if`-block that opens with

```
if __name__ == "__main__" :
```

- The `__name__` **identifier** is automatically defined by Python. If the module is imported `__name__` is equal to the module name. It is only equal to `"__main__"` if the module is invoked as a script.

DEMONSTRATION

Working with **Python modules**

***If necessary*, clone the *Lecture 8* Python demonstration material from GitHub:**

```
$ git clone https://github.com/hughdickinson/CompPhysL8Python.git
  /home/computationalphysics/Documents/python/lecture8
```

NUMPY (CRUNCHING NUMBERS)

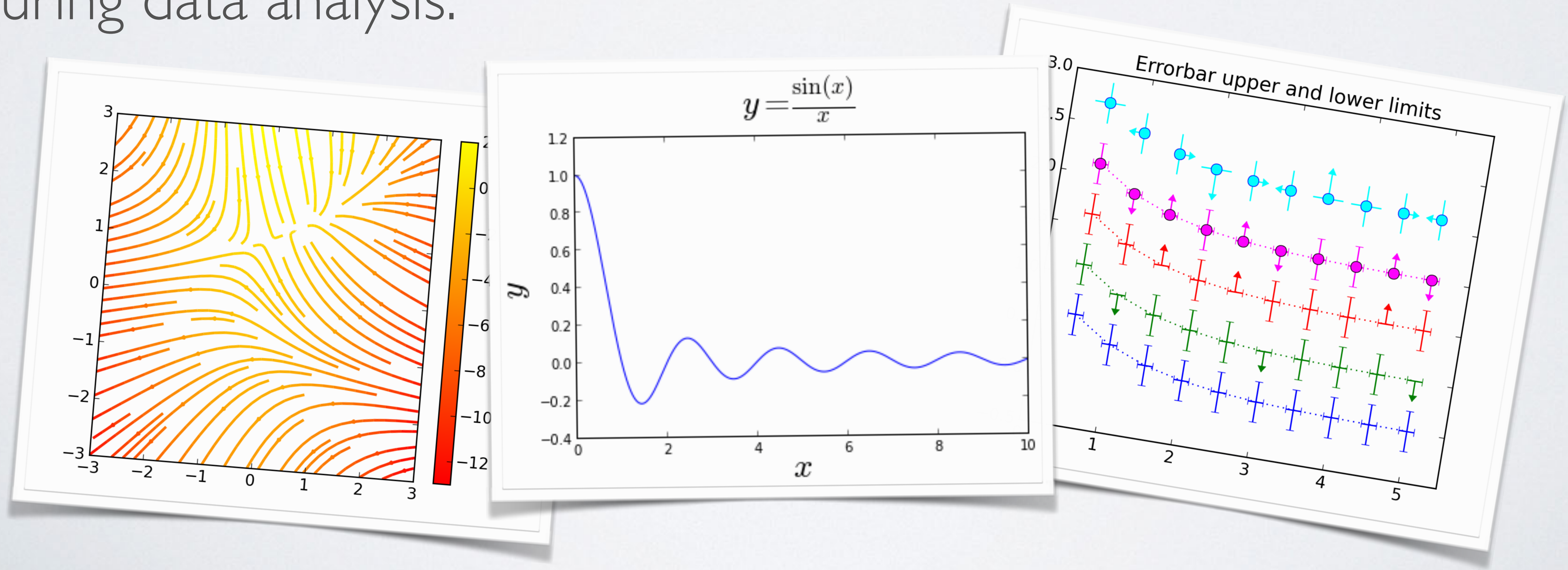
✑ *The **NumPy** package provides invaluable **data-handling** support for numerous **scientifically targeted** Python utilities.*

- Its primary functionality is the provision of a **homogeneous** (i.e. elements all have the **same type**), **multidimensional** array type.
- This array type supports **indexing**, **slicing** and **reshaping** as well **highly efficient** array-wise mathematical operations.

☞ **Mathematical operations** can be performed on whole multidimensional arrays **without the need for explicit loops**.

MATPLOTLIB (PLOTTING RESULTS)

- *MATPLOTLIB is a feature-rich 2-dimensional **plotting package** for Python.*
- Matplotlib can be used to generate **publication quality figures** in a **variety of formats**.
- It can also be used to plot and **review results** at intermediate stages during data analysis.



DEMONSTRATION

Using **NumPy** and **Matplotlib**

***If necessary*, clone the *Lecture 8* Python demonstration material from GitHub:**

```
$ git clone https://github.com/hughdickinson/CompPhysL8Python.git
  /home/computationalphysics/Documents/python/lecture8
```

LECTURE 9 SUMMARY

- After reviewing the material from this lecture (including the demonstration material) **and completing the reading exercises** you should know:
    1. How to use the **doxygen** utility to automatically generate documentation from **specially formatted** comments within C++ source code.
    2. How to **expose** C++ comment text to the **doxygen** utility.
    3. How to use **special tokens** in your comments to control the format of **doxygen**-generated documentation.

LECTURE 9 SUMMARY

4. That the **operation** of the **doxygen** utility can be **customized** using a **configuration file**.
5. Where to find **more information** about the **doxygen** utility and the **comment syntax** it requires.
6. How Python facilitates code reuse with **packages** and **modules.**
7. How to **extend the functionality** of Python by **import**ing modules.

LECTURE 9 SUMMARY

8. Some simple ways in which to **instantiate** multidimensional NumPy arrays.
9. How to **index** and **slice** NumPy arrays and the built-in Python sequence types.
10. How to apply **array-wise mathematical operations** to instances of the NumPy array type.
11. The **basics** of generating figures using **Matplotlib**.
12. Where to find **online reference material** for NumPy and Matplotlib.

OPTIONAL READING

The Python Online Tutorial
docs.python.org/2/tutorial/

The Matplotlib “Beginner’s Guide”
matplotlib.org/users/beginner.html

The doxygen “Getting Started” guide:
www.stack.nl/~dimitri/doxygen/manual/starting.html

The NumPy Online Tutorial
wiki.scipy.org/Tentative_NumPy_Tutorial

LECTURE 9 HOMEWORK

Review the **C++** and **Python** demonstration material from Lecture **9**!

- Read ***Section 6: Modules*** from the **Python Online Tutorial**
  docs.python.org/2/tutorial/
- Read **all** subsections of ***Section 2: The Basics*** from the **NumPy Online Tutorial**
  wiki.scipy.org/Tentative_NumPy_Tutorial
- Read the ***Pyplot Tutorial*** section from the **Matplotlib "Beginner's Guide"**
  matplotlib.org/users/beginner.html

- Complete the **Lecture 9 Homework Quiz** that you will find on the course Blackboard Learn website.