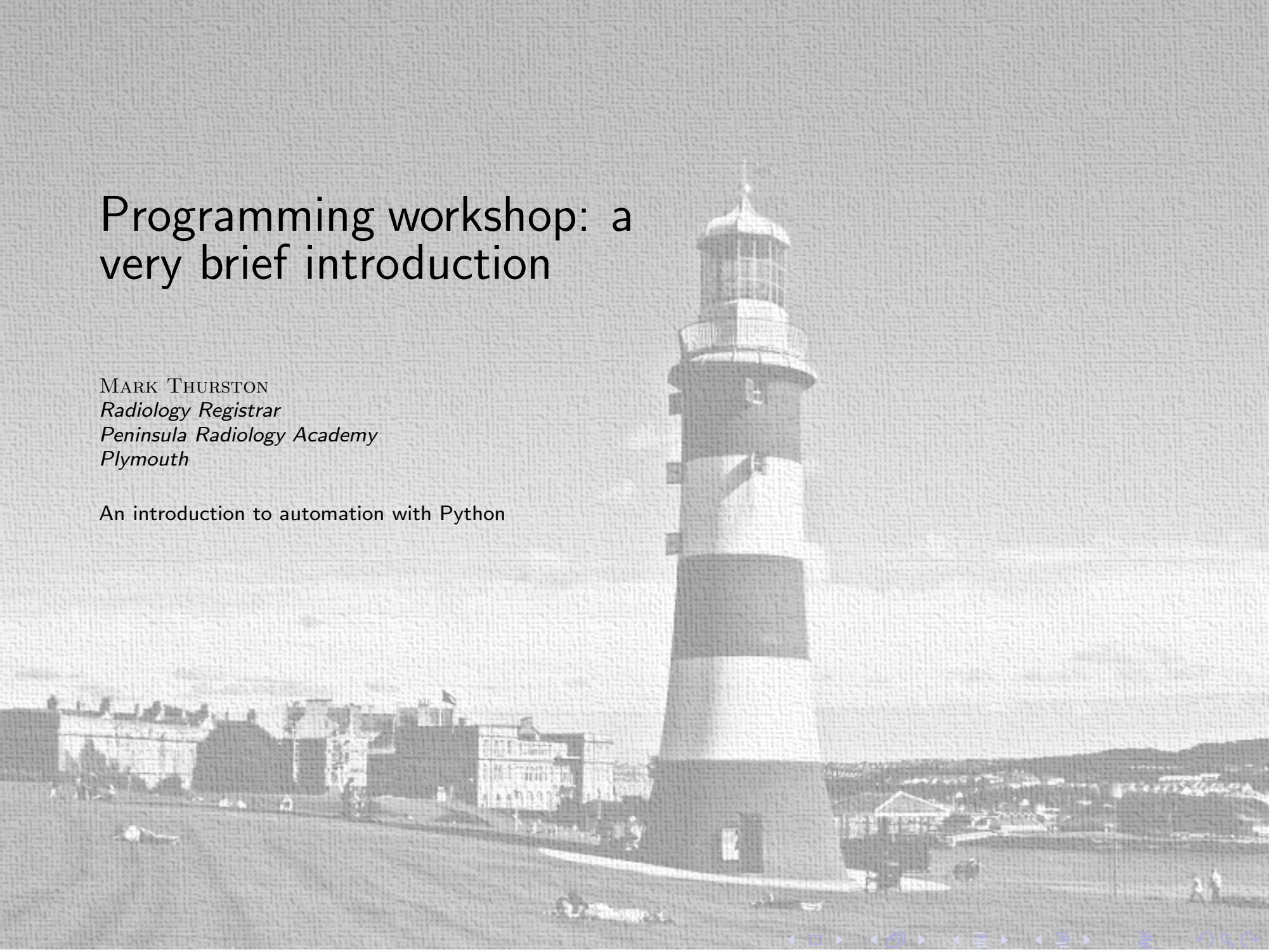
Programming workshop: a very brief introduction

Mark Thurston
Radiology Registrar
Peninsula Radiology Academy
Plymouth

An introduction to automation with Python



Aims and objectives

By the end of this session, you will be able to:

- Part 1: Theory
  - understand basic terms
  - have a good idea of some tasks that are possible
  - know where to learn more, if you are interested
- Part 2: Practice
  - set up a Python environment
  - run some pre-written example programs
  - Write your own simple example programs
- questions welcome throughout

Why?

how are these skills useful or relevant to me?

- make your life easier: automate repetitive and boring tasks
- get a new angle on a problem
  - some simple problems are impossible without very basic programming skills
  - master your research and audit data
- useful in understanding and implementing emerging radiology technologies
  - AI, radiomics
  - knowledge enables you to navigate the hype – useful at conferences and as a consultant
- interesting and fun :) – especially if you enjoy learning new things

Definitions: programming language

"formal language that specifies a set of instructions that can be used to produce various kinds of output"

- like any language:
  - spelling
  - grammatical rules
  - common useage
  - skill level – basic to mastery
    - the basics can still be put to very good use
    - mastery may take a lifetime
- categorisation
  - high level (Python, Java, etc) vs. lower level (C, assembly) vs. low level (machine code)
  - by programming paradigm: imperitive (procedural, object orientated), declarative (functional, logical), symbolic

Definitions: Python

"The Python philosophy rejects exuberant syntax in favor of a simpler, less-cluttered grammar."

- based on the *ABC language* — optimised for ease of learning
  - "Simplicity is the ultimate sophistication" — William Gaddis
- open source:
  - *Python Software Foundation Licence (PSFL)*
  - BSD style licence
- massive community
  - loads of extra functionality available as modules
  - domains include data processing, artificial intelligence, image recognition
  - help is easily available online for free if you get stuck

Definitions: abstraction

"Ignore the characteristics that we don't need in order to concentrate on those that we do."

- in general:
  - divide a programming problem into simpler, analogous pieces
  - solve the problem by combining solutions to simpler pieces
- Application Programming Interface (API)
  - a set of clearly defined methods of communication between various software components

```
def drive_car(location):
        <precise instructions>

drive_car("50.376289, -4.143841")
drive_car("51.509865, -0.118092")
```

Basic language syntax: hello world

your first program

"The only way to learn a new programming language is by writing programs in it. The first program to write is the same for all languages: Print the words: *hello, world* ... With these mechanical details mastered, everything else is comparatively easy."
– Kernighan and Ritchie ("K&R"), 1978

- the interpreter: python3
- saving the file and running as a script: the *"shebang"*
- the print function: prints objects to a specified place (in our case, the screen *stdout*)

```
print( *objects, sep=' ', end='\n',
       file=sys.stdout, flush=False)
```

Basic language syntax: comments and docstrings

ensure maximum readibility... for others (and yourself)

- a # character makes the interpreter ignore the rest of the line
- at the start of a function definition, a *docstring* describes the function

```
def test_function():
        '''Runs a test.'''
        print("Testing... 1, 2, 3")

# run the test
test_function()
```

Basic language syntax: variables & data

data are the fundamental building blocks of any program

- variables allow storage of data by a name
- be careful how you name them!
  - clear and consise names: easy to read, descriptive
  - don't clash with other names in use (including language reserved keywords
  - Python style guide recommendation: variable names should be lowercase, with words separated by underscores as necessary to improve readability. Uppercase global variable names.

```
my_var = "value"
denominator = 7
GLOBAL_VARIABLES = "all␣caps"
```

Basic language syntax: lists, iterables

a list is a type of sequence

- an iterable is an object with multiple elements:
  - including *lists*, *tuples*, *sets*
- accessing multiple times can allow you to perform an operation on every element

```
body_parts = ["hand", "elbow", "shoulder"]
months = ("Jan", "Feb", "Mar", ...)
empty_list = list()

>>> print(months[0]) # 0-indexed
Jan
```

Basic language syntax: Boolean statements

True, False, 1, 0

- Boolean algebra named after George Boole, 1815 – 1864
- the lowest construct that a computer understands
- the return value from a test statement is either True or False

```
>>> 1 == 1
True
>>> 10 <= 0
False
```

Basic language syntax: conditional statements

perform a different action depending on the result of a test

- control which part of the program is activated, depending on a test

```
if <statement>:
        <indented code block>
elif <another statement>:
        <indented code>
else:
        <more code>

<statement> == True # activates
<statement> == False # skips
```

Basic language syntax: loops

repeat a process over and over until a condition is met

- *iteration*: perform an operation on multiple elements
  - each file in a directory
  - each DICOM image in a list
- in Python — *for loop* and *while loop*:

```
for item in <iterable>:
        <code block>

while <condition>:
        <code block>
```

Basic language syntax: user input

store data to operate on

How do I get data in to my program?

- *stdin*: "standard input"
- *args*: command line arguments

```
# get some standard input
user_input = input()

# prints all the items in the argument vector
import sys
for item in sys.argv:
    print(item)
```

Basic language syntax: using external modules

add features to your program

- allows you to take advantage of built in and community contributed external code
- pip: tool to collect and install external modules from internet repository — minimal effort required

```
# use the os module to find all the files
import os
files = os.walk('.')

# install a new python module
$ pip3 install tensorflow
```

Designing your program: drawing board

think about the process

"Computer science is no more about computers than astronomy is about telescopes."

– Dijkstra

- paper and pen
- get the steps right before you start and the rest will be easy
- the programming language syntax is of secondary importance

For example, your digital camera has produced a directory of files named IMG_3624.JPG etc... You would prefer them named Greece2018_xxx.jpg

Think about the steps that would be required for this:

1. select the appropriate directory location
   - either hard-coded or user-selectable
   - can you trust the user to enter a valid location?
2. build a list of all the items you want renamed
3. go through each item, performing the rename operation
   - format for renaming? a counter might be needed
4. exit

Possibilities: remove all duplicate patients from a spreadsheet

1. Precisely define the task: how do you define a duplicate row?
2. import the data to Python using the help of an external module
   - method depends on the format of the data – *Excel*, *comma separated value file*
3. loop through all the records removing those that fit (or don't fit) the criteria
4. write to a new file
5. exit

Possibilities: find all phone numbers in a text document

1. What does a phone number look like?
   - country code [optional], area code, number
   - may include dashes or spaces
2. Design a pattern (*regular expression*) that fits the above
3. Test it, looking for false positives and false negatives

Modules: useful extensions

A few selected modules out of hundreds

- *Tensorflow*: open source machine learning library from Google
- *Pandas*, *numpy*: data analysis toolkits
- *Requests*: allows you to get webpages and text from the web into your program
- *Scrapy*: scrape data from the web
- *wxpython*: build a graphical program

Resources: books

Learning resources available on the internet

- Automate the boring stuff with Python: `https://automatetheboringstuff.com/`
- Another free online tutorial: `https://python-textbok.readthedocs.io/en/1.0/`

Resources: interactive tutorials

Learning resources available on the internet

- Massive open online courses: some free, some cost
  - EdX: `https://www.edx.org/course?search_query=python`
  - Coursera: `https://www.coursera.org/courses?languages=en&query=python&userQuery=python`
  - Udacity: `https://eu.udacity.com/`
  - udemy `https://www.udemy.com/`

Resources: practice to improve your skills

Thousands of resources are available

- coding websites
  - `https://www.hackerrank.com/domains/python`
  - `https://www.codecademy.com/tracks/python`
- open access data
  - `https://data.gov.uk/`
- read other code on `https://github.com`
- contribute to an open source project
  - Horos
  - Orthanc
  - OpenCV
  - Python

- Stackoverflow:
  `https://stackoverflow.com/tags/python/info`
- Internet relay chat: #python on irc.freenode.net
- Mailing lists:
  `https://www.python.org/community/lists/`

Summary

Topics covered

- basic definitions and background
- language syntax
  - variables
  - control structures
  - user input
- program design
- some uses
- help and resources