Object oriented software engineering: Spatial Algorithms

Lecture and Workshop 1

Gary Watmough, Peter Alexander

Admin

- All slides, handouts and materials will be on Learn.
- If you have questions outside this time, use the message board on learn so all your colleagues can see.

Course Essentials

- Monday 09:00 – 13:00
  - February 26th
  - March 5th
  - March 12th
  - March 19th
  - March 26th
- Location: Geography Computer Lab

Assignments

- 1 learning diary
  - 40%
  - Each week reflect on the material and the end of workshop task
  - What made sense? What didn't?
  - How could it be useful in other problems
- 1 project
  - 60%
  - Raster analysis
  - set in week 4 of the course
  - Due week 6

Getting Help

- Python website has a lot of help files
- StackOverflow can be great help for problems
- My Office hours: 14:00 – 16:00 Monday
- Outside of this time email me: gary.watmough@ed.ac.uk

Course aims

- Allow you to appreciate issues of algorithm design and development so you can manipulate spatial data and solve spatial analytical problems
- Give you practical experience in developing codes in python that can tackle specific problems.

Intended learning outcomes of the course

- understand principles of algorithm development
- understand generic concepts employed in algorithm design
- be familiar with a range of algorithms used to manipulate and analyse spatial data
- develop python classes suited to the representation and analysis of spatial data
- appreciate the concept of algorithm efficiency and how this can be assessed and improved
- undertake spatial data input/output in standard formats with other proprietary software.

Week by week guide

1. Handling spatial data:
    a) Simple geometric calculations, range searching and data sorting.
2. Divide and Conquer
    a) Binary searching, recursion and line generalisation
3. Grid data and arrays
    a) Handling, traversing and searching raster data. Point and focal functions.
4. Problem solving by task partitioning
    a) Nearest Neighbour Analysis and cartogram generation
5. Cartogram, ArcMap and Help
    a) Not assessed but good for possible dissertations and completing coursework

Important prerequisite skills and experience

- You should have completed the TIGIS course in semester 1
  - we will build directly on some of the work you did with Magnus
- You need to be able to develop code in Spyder or another development environment (eMac's etc)
  - I will use Spyder
- You need to have an appreciation of Object-oriented design principles and how to base an overall design around the specification of classes

Introductions

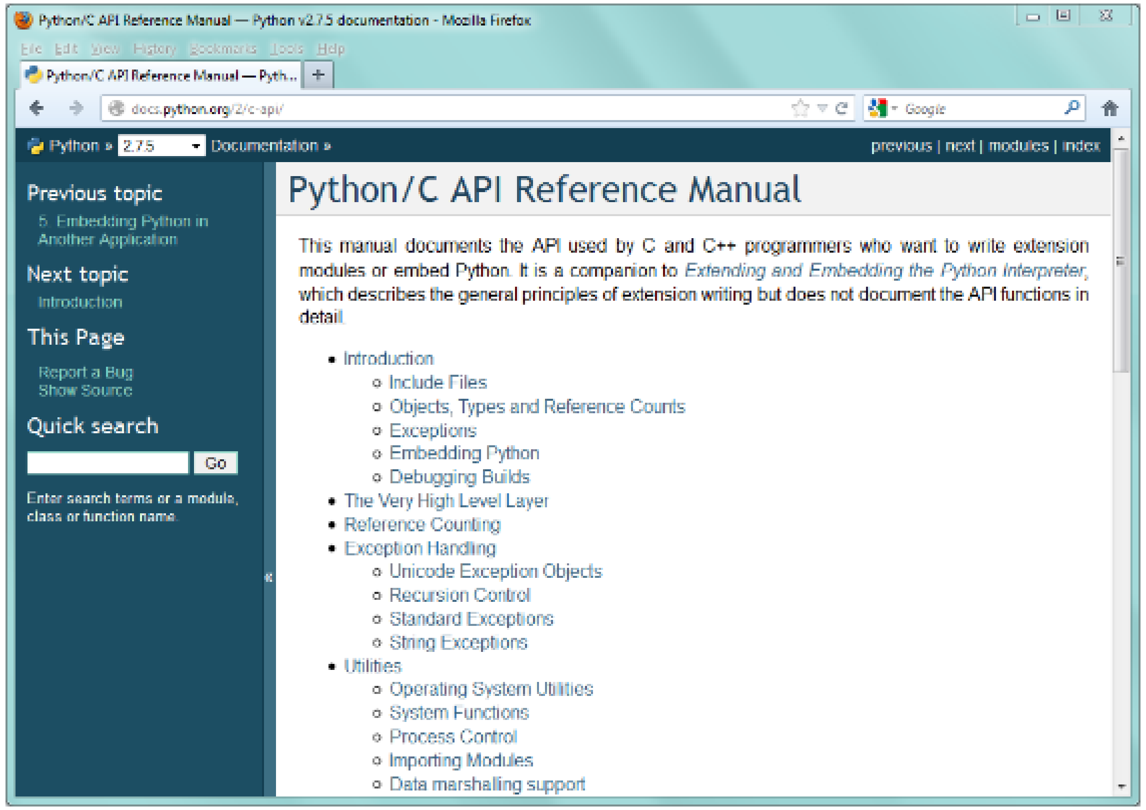
- Lots of people involved in the course with a range of backgrounds and skills
- Me
- Peter
- Magnus (week 3 and week 4)
- Demonstrators

TIGIS feedback

- Name
- Degree programme
- Experience of programming
- Do you have experience of spatial algorithms/GIS?
- What do you want to get out of the course?

These lectures

- We cannot teach you anything!
- We can facilitate your learning, but it is up to you to be active
- Python online resources:
- www.python.org
- docs.python.org (tutorials, reference manuals)


Python/C API Reference Manual — Python v2.7.5 documentation - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Python/C API Reference Manual — Pyth...
docs.python.org/2/c-api/
Google
Python » 2.7.5 Documentation »
previous | next | modules | index

Previous topic

5. Embedding Python in Another Application

Next topic

Introduction

This Page

Report a Bug
Show Source

Quick search

Go

Enter search terms or a module, class or function name.

Python/C API Reference Manual

This manual documents the API used by C and C++ programmers who want to write extension modules or embed Python. It is a companion to *Extending and Embedding the Python Interpreter*, which describes the general principles of extension writing but does not document the API functions in detail.

- Introduction
  - Include Files
  - Objects, Types and Reference Counts
  - Exceptions
  - Embedding Python
  - Debugging Builds
- The Very High Level Layer
- Reference Counting
- Exception Handling
  - Unicode Exception Objects
  - Recursion Control
  - Standard Exceptions
  - String Exceptions
- Utilities
  - Operating System Utilities
  - System Functions
  - Process Control
  - Importing Modules
  - Data marshalling support

Texts and other resources

- Python in a Nutshell (2009) Alex Martelli, O’Reiley
- Learning Python (2009) Mark Lutz, O’Reiley
- Some of the texts are in Python 2 and we will work in Python 3.
  - so if you have trouble with errors check first that Python 3 uses the same syntax and has the same function/algorithm
  - eg
    - print something
    - print (something)

Algorithm Design Texts

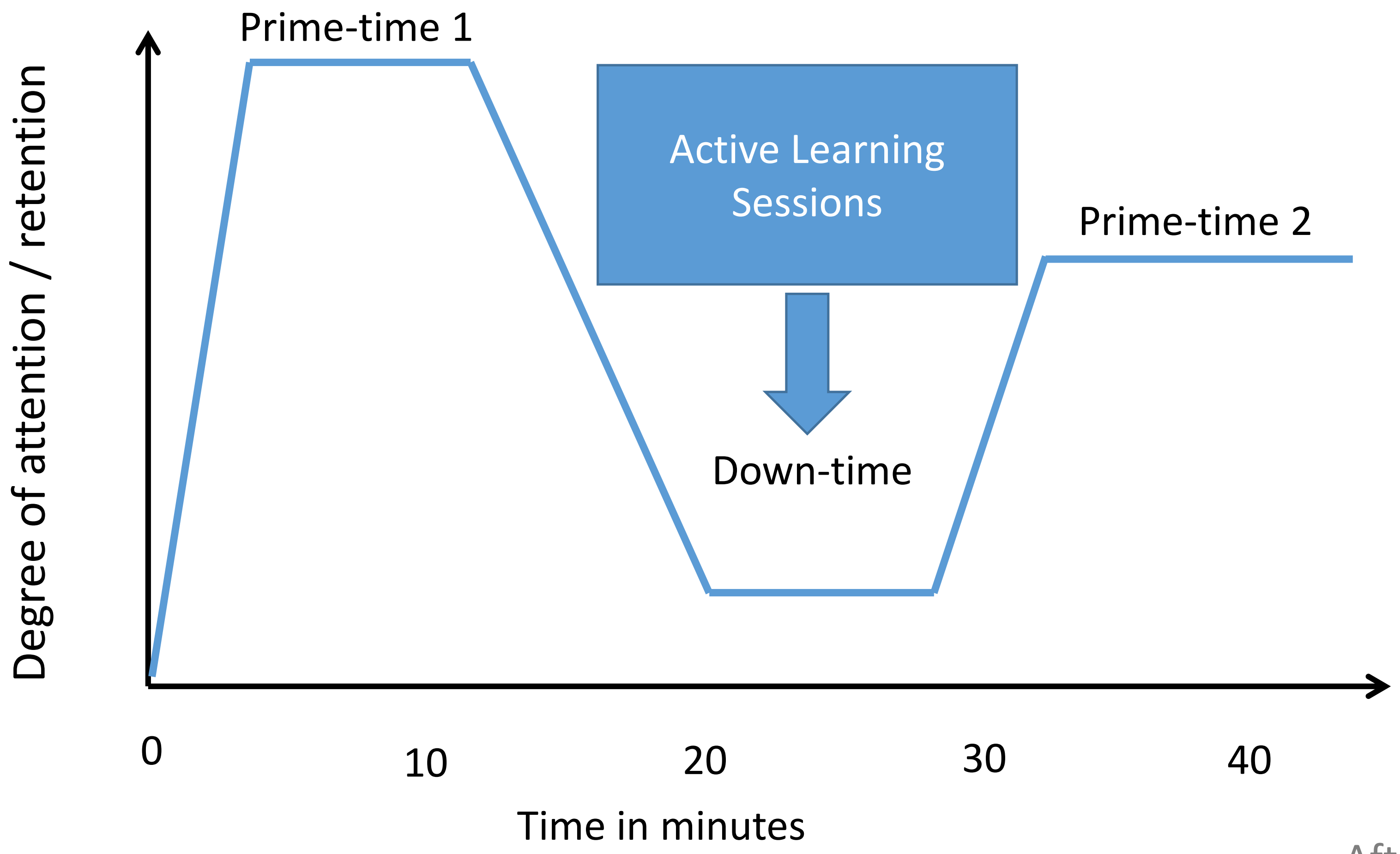
- Sedgewick and Wayne (2011) Algorithms 4th Edition
- Cormen, Leiserson, Rivest and Stein (2009) Introduction to Algorithms
- Skiena (2011) The Algorithm Design Manual
    - http://www.algorist.com
    - this book has an accompanying website which has lecture slides and youtube videos of lectures all about algorithm design. It is quite heavy going and I don't expect you to get all of it, but it might help with some of the general issues around algorithm design.

Structure of the lectures/workshops

- Interactive sessions
- Keep as much time as possible each week for writing code and playing with Python.

- Present chunks of code as an introduction
- In the folder you will also get pre-built python modules during the workshops you will need to work through the modules, identify what they are doing and add in the chunks of code we work on

Structure of lectures/Workshops



After Sousa 2001

Source: Ole Lauridsen 2017

Any Questions?

This week – intended learning outcomes

- Principles of good algorithm design and code
    - understand principles of algorithm development
    - understand generic concepts employed in algorithm design
    - Introduction to algorithm efficiency

- Familiarity with algorithms:
    - Simple geometric calculations
    - Range searching

    - Sorting
    - Sorting lists
    - Sorting list of point objects

Algorithm Design (Skeina 2011)

- Algorithm
  - procedure to accomplish specific task
- Algorithmic problem specified by describing:
  - the complete set of instances it must work on, and:
  - the output after running on one of the instances

For example:

- problem = sorting
- input = sequence of *n* keys a_1 … a_n
- output = reordering of input sequence so that $a_1 \leq a_2 \leq a_{n-1} \leq a_n$

Desirable properties of algorithm

1. Correct
2. Efficient
3. Easy to implement

Keep these in mind throughout your work and assignments!

It is often not obvious that an algorithm correctly solves a given problem

Expressing Algorithms

- English
- Pseudo-code
- Python

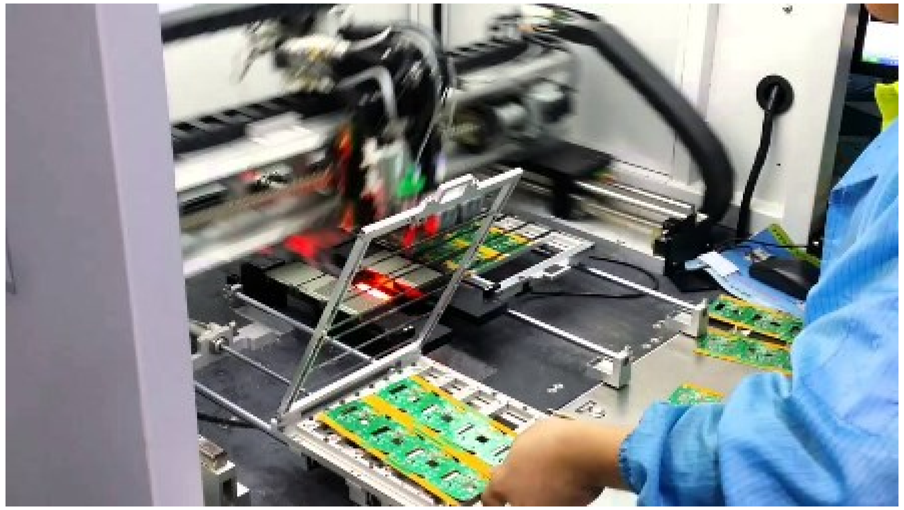
Use all three when writing up your projects.

Take Home message

- An idea is at the heart of any algorithm – we should be able to see the idea
- if idea is not clearly written when expressing algorithm the notation needs improving.

Robot Tour Optimization

- Circuit board
    - each chip has contact points needed to be soldered
    - program the arm to visit each contact point and return to start before next board
    - minimize the time taken to assemble each board
    - fixed speed movement of arm so the time to travel is proportional to the distance between points.

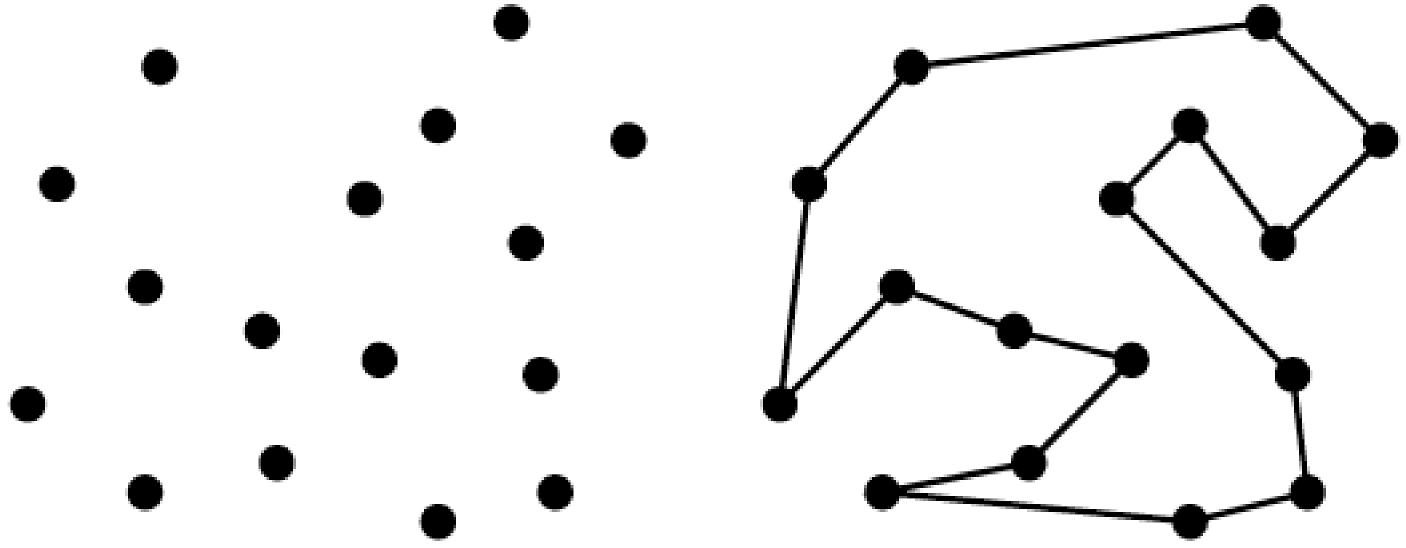


Robot Tour Optimization a.k.a traveling salesperson problem

- Problem = robot arm tour
- input = a set of n points in the plane
- output = the shortest cycle tour that visits each point in the set

How can you do it?

Any ideas on the algorithm?



Nearest Neighbour

- Starting at Point 0 (p_0)
    - travel to nearest neighbour p_1
    - from p_1 travel to nearest unvisited neighbour
    - repeat until run out of unvisited points
    - then return to p_0

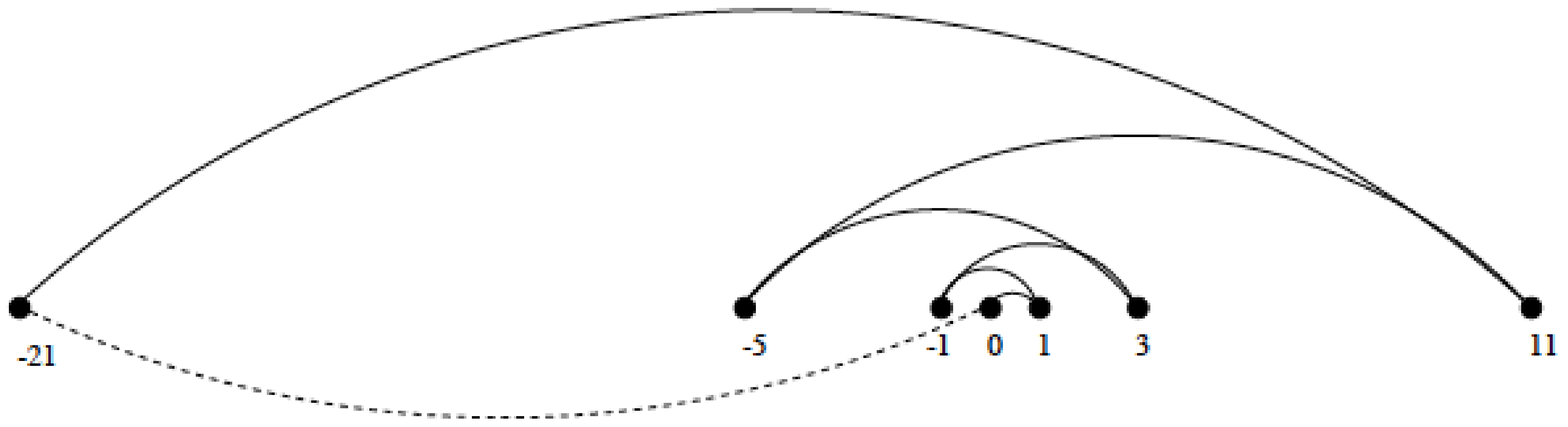
Nearest Neighbour

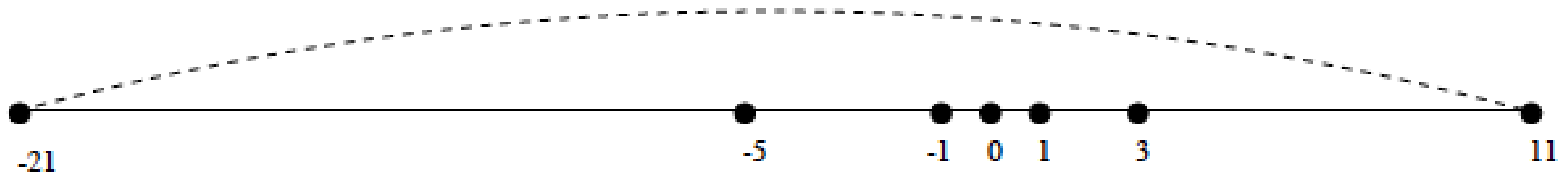
```
NN (p)
    pick and visit an initial point $p_0$ from P
    p = $p_0$
    i = 0
    while there are still unvisited points:
        i = i+1
        select $p_i$ to be closest unvisited point to $p_{i-1}$
        visit $p_i$
    return to $p_0$ from $p_{n-1}$
```

Nearest Neighbour

- Simple to implement and understand
- makes sense to go to nearest P each time
- relatively efficient
- but it can be completely wrong!

Nearest neighbour

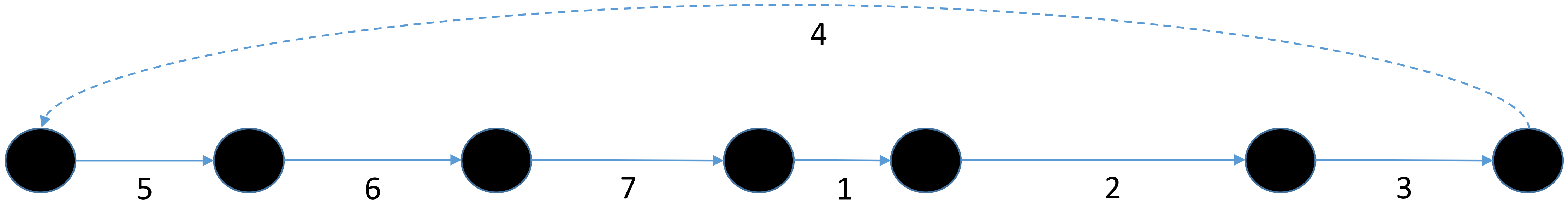




- NN nearest point each time means jumps left to right
- optimal solution starts at -21 and travels right until 11 then returns home.

Closest Pair Rule?

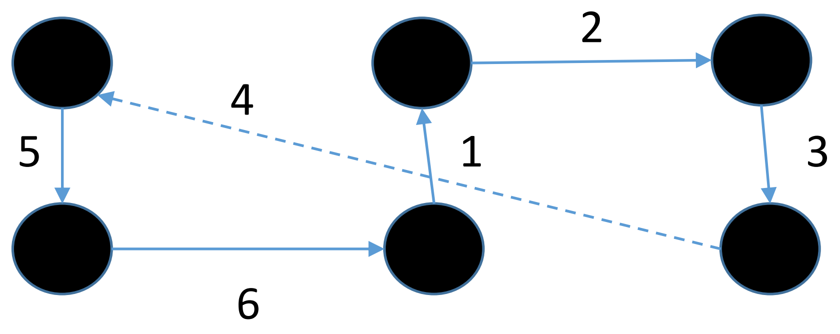
- Repeatedly connect closest pair of endpoints
- works in some cases

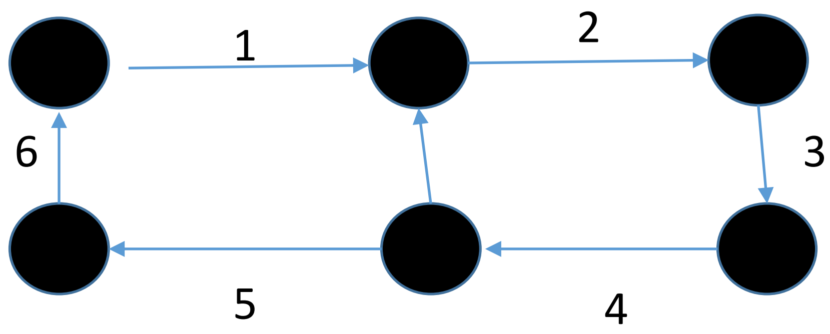


Closest Pair Rule?

- Repeatedly connect closest pair of endpoints
- would not work here:

Best approach is:





- because 1st is smallest the tour will start there.

What does the correct algorithm do?

- enumerate all possible orderings of the set of points and select the order that minimises the total length:
- **Brute force method**
    - **since all possible orderings/tours are considered we will get right answer**
    - **but it is extremely slow**
    - **Order n2 means that doubling the number of cells would roughly quadruple the time**
    - **Computational effort is high and expensive**
    - **because is it factorial**
        - **huge potential number of possible outcomes to consider.**
        - **for a circuit board with > 1000 points it is impractical.**

Take Home Message

- Algorithms – always produce correct result (brute-force)
- Heuristics – can do a good job but no guarantee (NN, closest pairs)

Any Questions?

Take a break

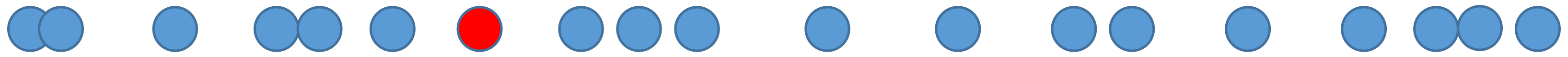
Remainder of lecture: Algorithms

- Range searching

- Sorting
    - simple lists
    - list of points

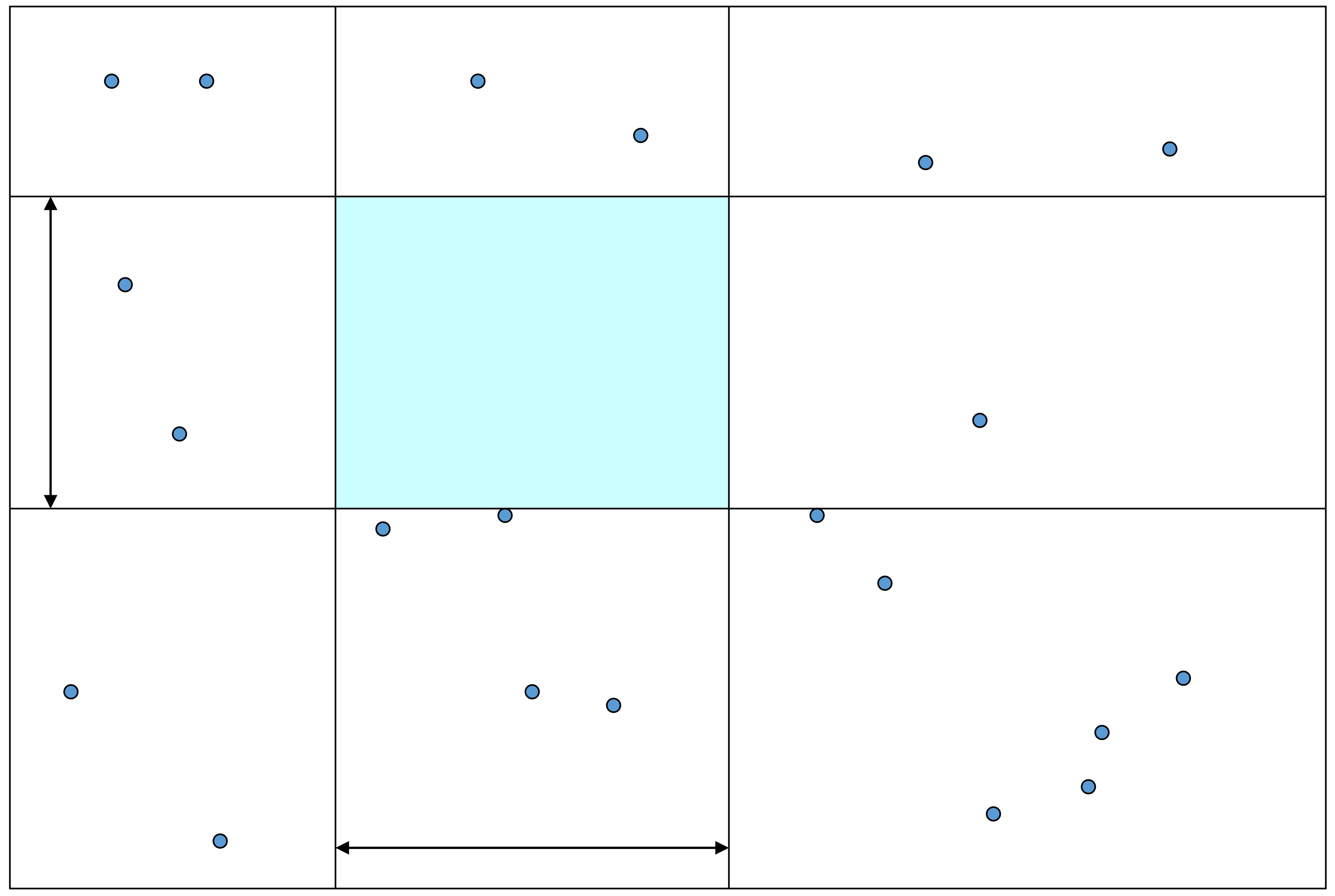
In 1-dimension

- How do we find the closest one to another object (red dot)?



- If the data representing the points are not ordered – what can we do?
- On average, how many data items do we need to look at?
- If the data are sorted along the line, can we find the nearest one more quickly

Similar concepts in 2D - Projection searching



Searching in 2D is harder because you can’t easily store points simultaneously sorted in X and Y

Finding the minimum

- We want a function or method which finds the ‘minimum’ of a list
- What argument would such a function take?
- What value would it return?

Coding problem 1: Locate Minimum

- Create a function to generate a random list of values
- You can use the function

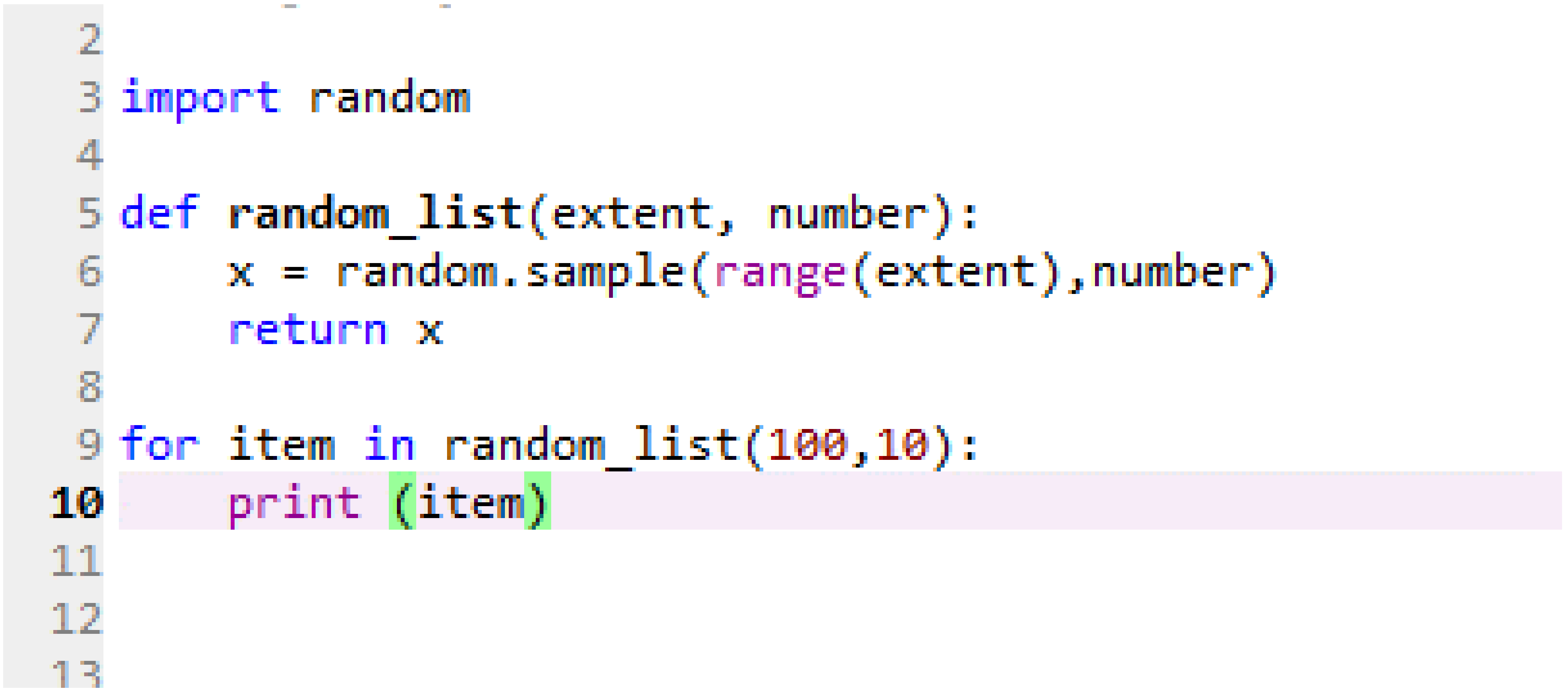
```
    random.sample(sequence,number)
```

- This will generate random values if you

```
import random
```

- Can you print out the list?
- Can you modify the function to control the list size and value range?
- For simplicity you might want to do all this in new Python module

Create a random list of integers



```

import random

def random_list(extent, number):
    x = random.sample(range(extent),number)
    return x

for item in random_list(100,10):
    print (item)
```

What is each line doing? Add comments to your script before moving on…

Coding problem 1: Locate Minimum

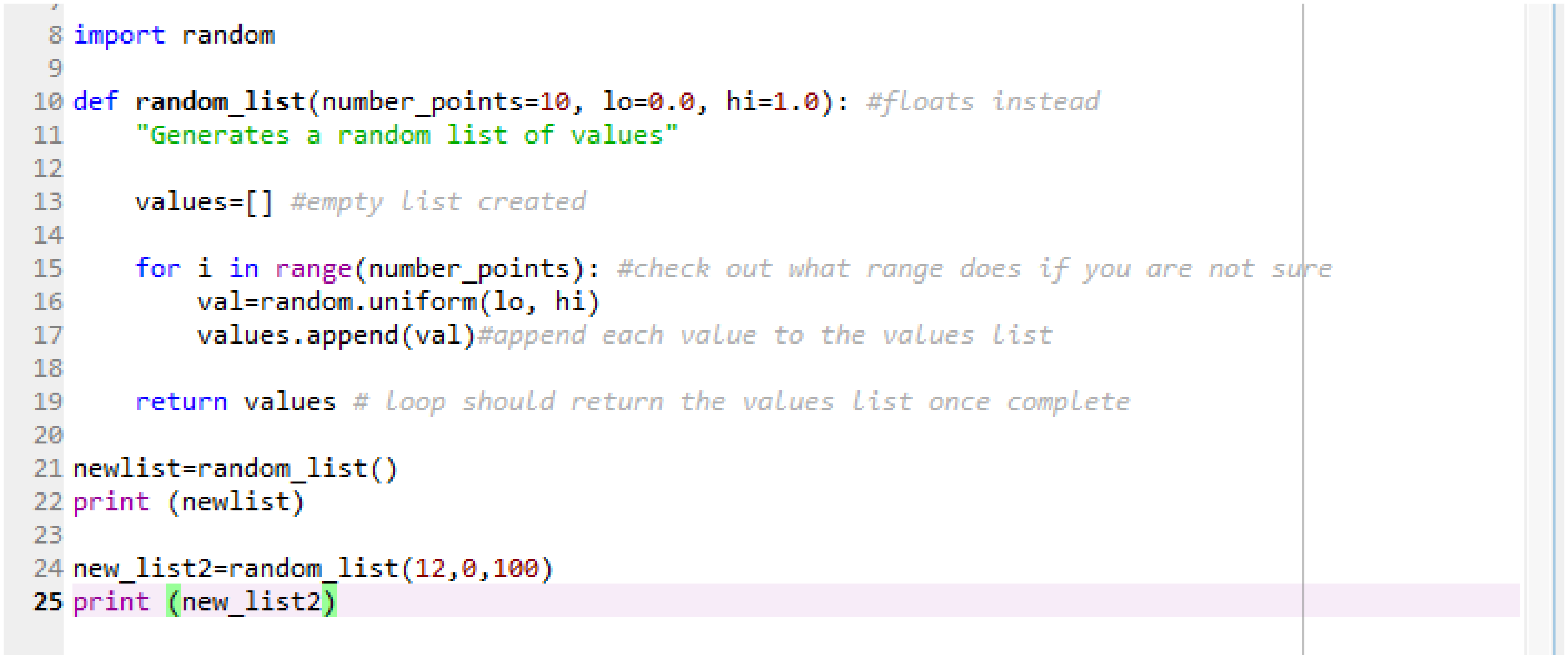
- Create a function to generate a random list of values
- You can use the function

```
random.sample(sequence,number)
```

- This will generate random values if you

```
import random
```

- Can you print out the list?
- Can you modify the function to control the list size and value range?
- For simplicity you might want to do all this in new Python module


```
import random

def random_list(number_points=10, lo=0.0, hi=1.0): #floats instead
    "Generates a random list of values"

    values=[] #empty list created

    for i in range(number_points): #check out what range does if you are not sure
        val=random.uniform(lo, hi)
        values.append(val)#append each value to the values list

    return values # loop should return the values list once complete

newlist=random_list()
print (newlist)

new_list2=random_list(12,0,100)
print (new_list2)
```

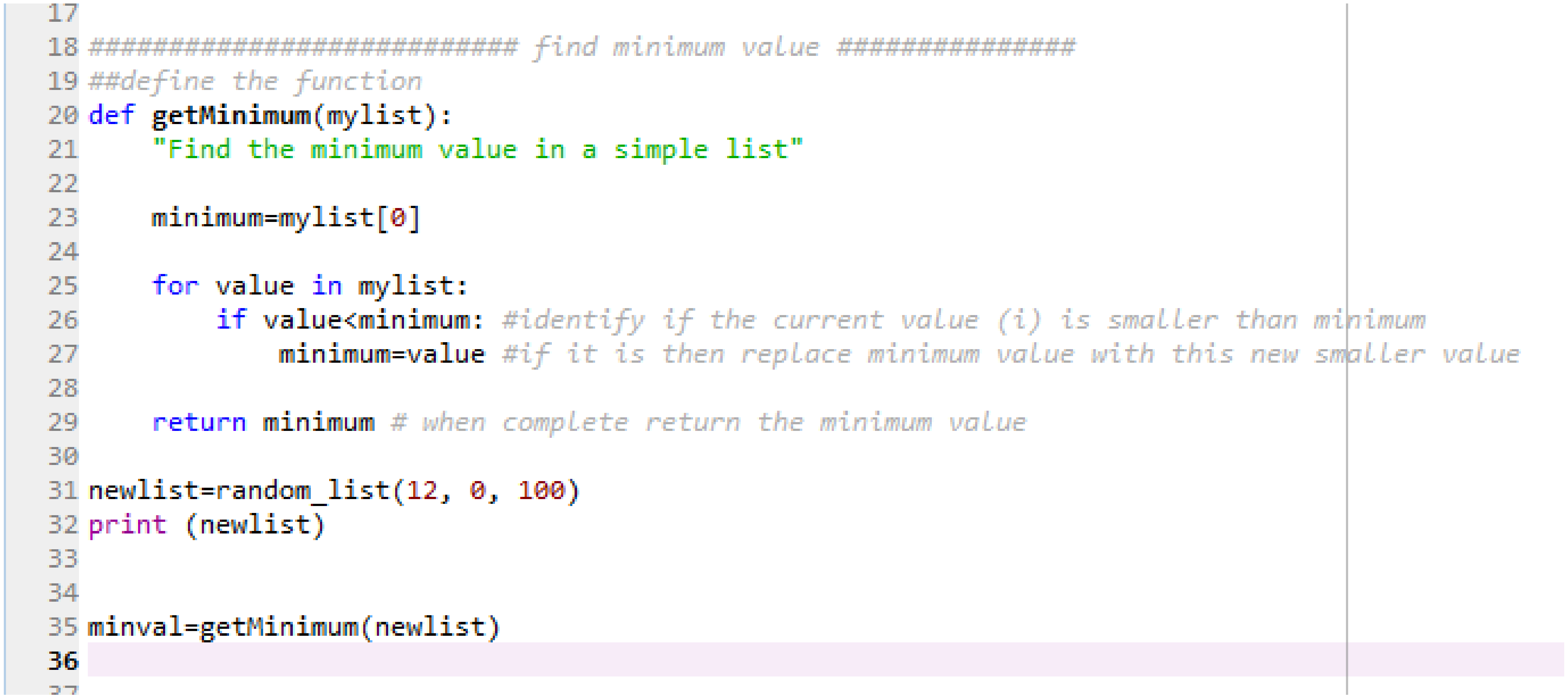
We expect to see this sort of commenting for assignments. We want to know that you are aware of what the functions are doing

But don’t just copy blindly!

Coding problem 1: Locate Minimum

- Locate the minimum value in your list of values?
- The key is to ‘hang on’ to a ‘current minimum’ and test against the others.
- Pseudo-code:

```
Function to get minimum from a list:
    min = define a variable which starts with a value
    loop through the list
            if the ith value is smaller than the predefined variable (min)
                    update the value of min to the ith value
```



```python

############################ find minimum value ################
##define the function
def getMinimum(mylist):
    "Find the minimum value in a simple list"

    minimum=mylist[0]

    for value in mylist:
        if value<minimum: #identify if the current value (i) is smaller than minimum
            minimum=value #if it is then replace minimum value with this new smaller value

    return minimum # when complete return the minimum value

newlist=random_list(12, 0, 100)
print (newlist)


minval=getMinimum(newlist)

```

Can Python do this already?

- We have got you to do this for a reason….
- But.. Can you locate an inbuilt Python way to do this?

- Hint:

```
min(list)

print("the minimum is:"+str(min(newlist)))
```

Python functions

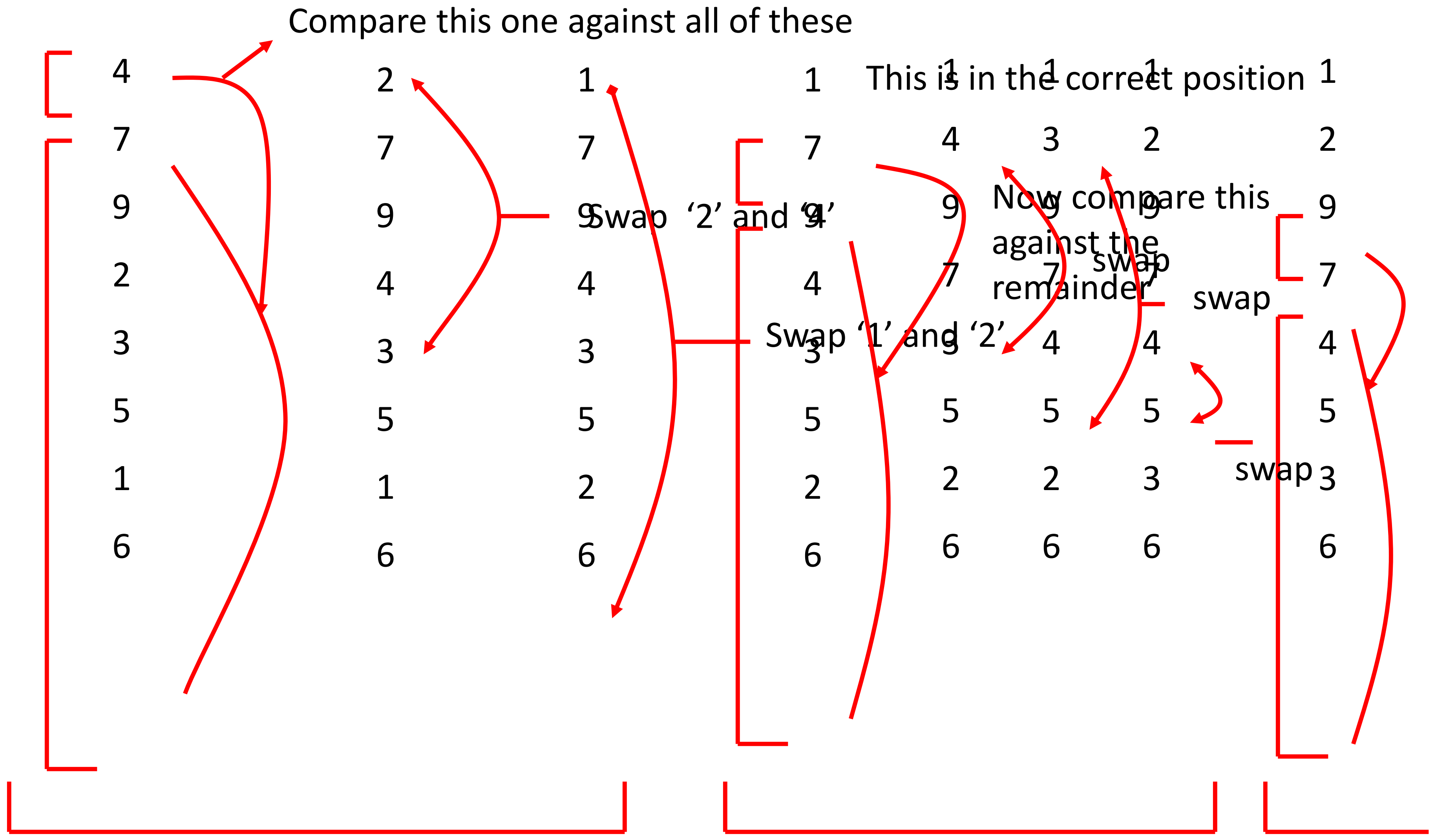
- Often there will be pre-built functions in Python
- But this course is focused on understanding how the algorithms work

Sorting a selection

Selection sort

- The procedure is as follows:
  - Find the **smallest** element in the set and swap it with the **first** element
  - Repeat the procedure finding the **second** smallest element in the set and swapping it with the **second** element (the second smallest element is the smallest element in the remaining data set)
  - Carry on until the entire set is sorted
- This sorting technique is not very efficient - it will require roughly n^2 iterations, where n is the number of elements we must sort.
- If you talk about the efficiency of algorithms you often give a value like this as an indication of the algorithm's efficiency.

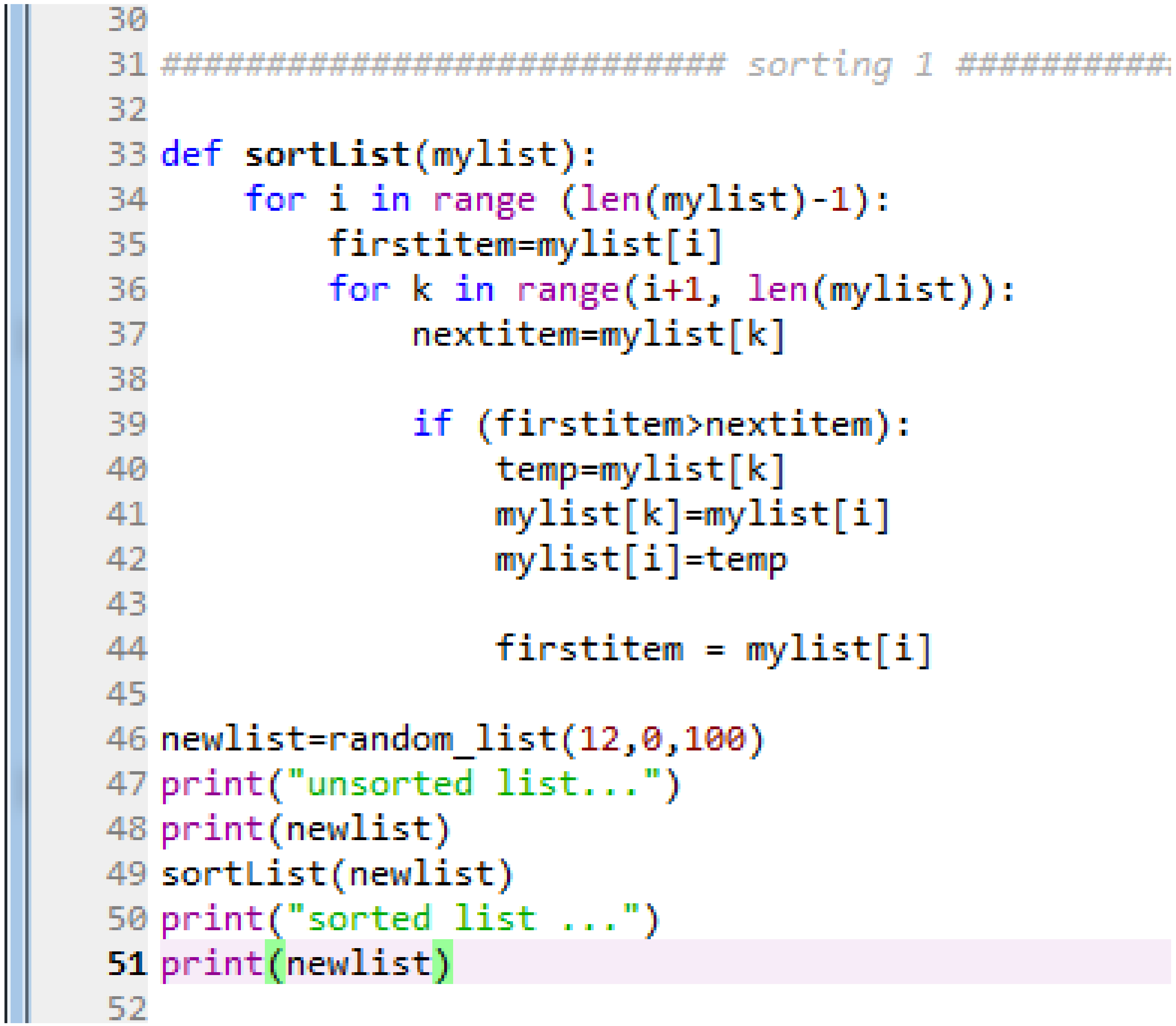
Selection Sort


Compare this one against all of these
Swap '2' and '4'
Swap '1' and '2'
This is in the correct position
Now compare this against the remainder
swap
swap
swap

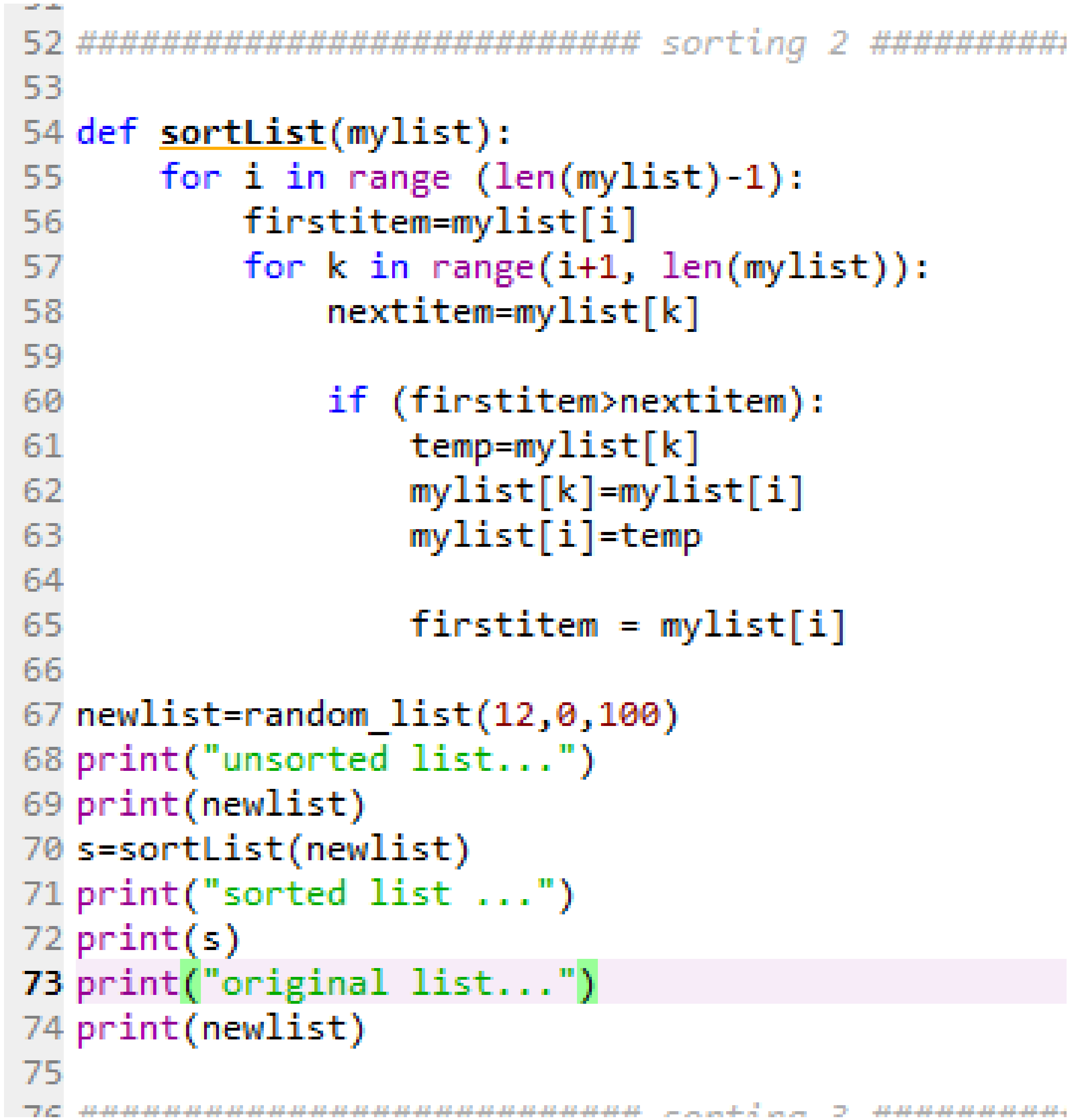
Coding problem 2: selection sort

- Try and code a simple selection sort for a list of values
- If you can do this, sorting other things is a little more complex, but works on exactly the same principles

- Remember: if you are struggling think in English, then pseudo-code, then attempt to build the python code.

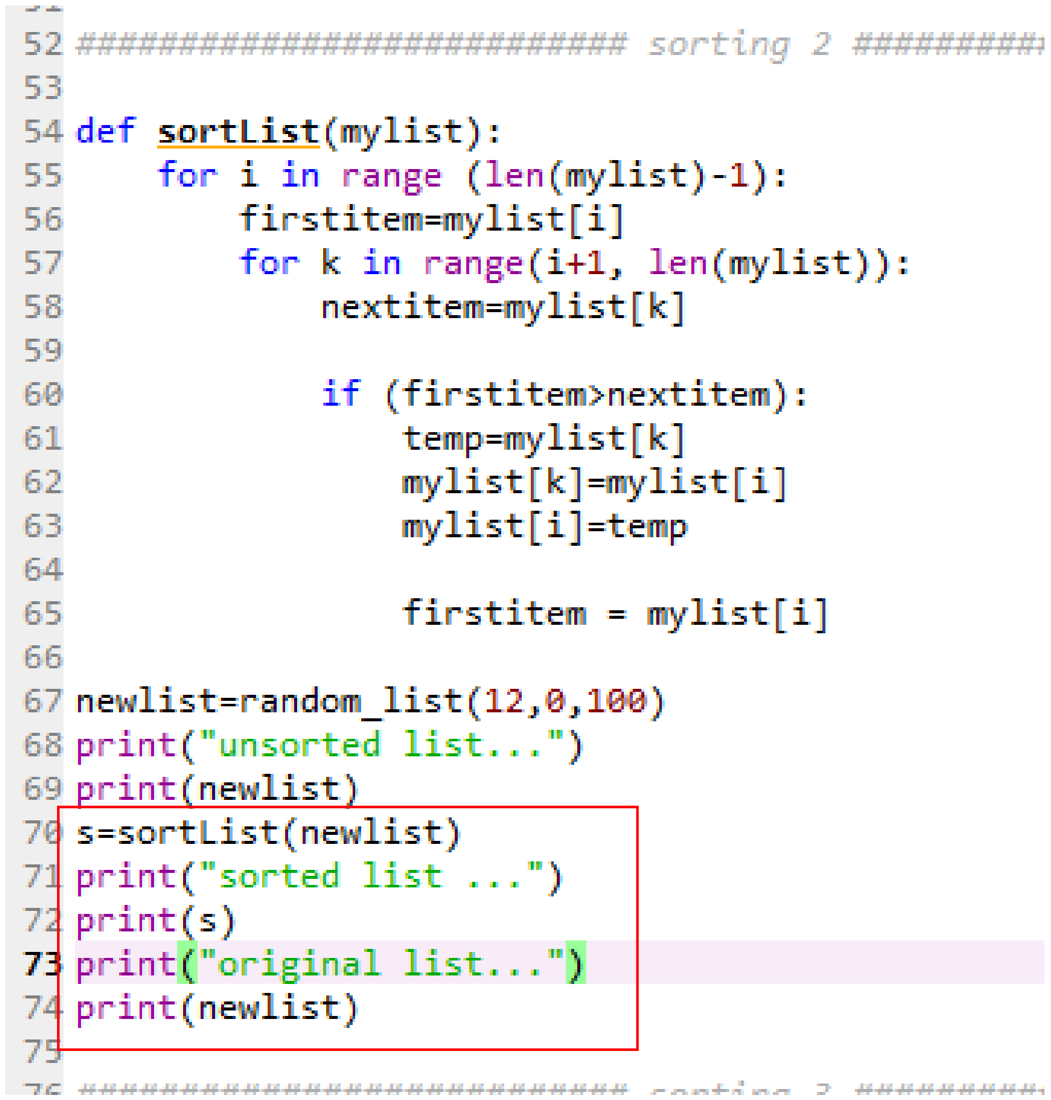


```

############################## sorting 1 ###########

def sortList(mylist):
    for i in range (len(mylist)-1):
        firstitem=mylist[i]
        for k in range(i+1, len(mylist)):
            nextitem=mylist[k]

            if (firstitem>nextitem):
                temp=mylist[k]
                mylist[k]=mylist[i]
                mylist[i]=temp

                firstitem = mylist[i]

newlist=random_list(12,0,100)
print("unsorted list...")
print(newlist)
sortList(newlist)
print("sorted list ...")
print(newlist)

```



```
############################### sorting 2 ##########

def sortList(mylist):
    for i in range (len(mylist)-1):
        firstitem=mylist[i]
        for k in range(i+1, len(mylist)):
            nextitem=mylist[k]

            if (firstitem>nextitem):
                temp=mylist[k]
                mylist[k]=mylist[i]
                mylist[i]=temp

                firstitem = mylist[i]

newlist=random_list(12,0,100)
print("unsorted list...")
print(newlist)
s=sortList(newlist)
print("sorted list ...")
print(s)
print("original list...")
print(newlist)
```

Problem!



```
############################## sorting 2 ##########

def sortList(mylist):
    for i in range (len(mylist)-1):
        firstitem=mylist[i]
        for k in range(i+1, len(mylist)):
            nextitem=mylist[k]

            if (firstitem>nextitem):
                temp=mylist[k]
                mylist[k]=mylist[i]
                mylist[i]=temp

                firstitem = mylist[i]

newlist=random_list(12,0,100)
print("unsorted list...")
print(newlist)
s=sortList(newlist)
print("sorted list ...")
print(s)
print("original list...")
print(newlist)

############################## sorting 3 ##########
```

- What is this doing?

print(s)

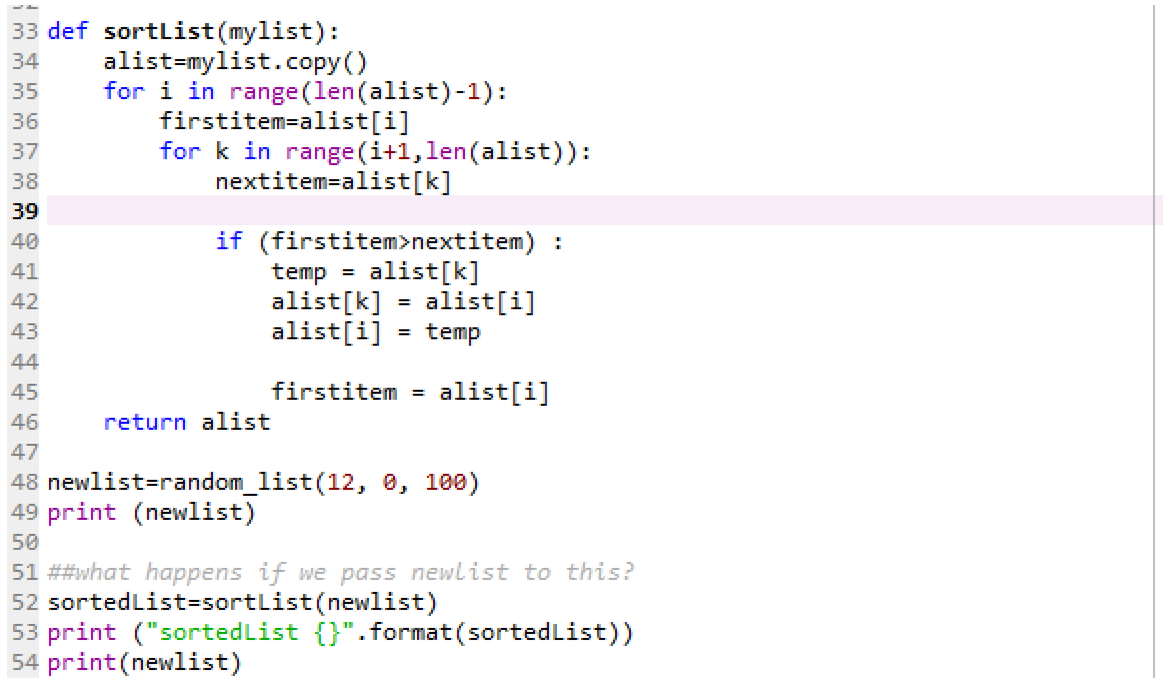
print(newlist)

Both the same?

Why?

Saving the original list and sorting a copy

- How can you do this?



```
def sortList(mylist):
    alist=mylist.copy()
    for i in range(len(alist)-1):
        firstitem=alist[i]
        for k in range(i+1,len(alist)):
            nextitem=alist[k]

            if (firstitem>nextitem) :
                temp = alist[k]
                alist[k] = alist[i]
                alist[i] = temp

                firstitem = alist[i]
    return alist

newlist=random_list(12, 0, 100)
print (newlist)

##what happens if we pass newlist to this?
sortedList=sortList(newlist)
print ("sortedList {}".format(sortedList))
print(newlist)
```

- Before we move on, comment the code. This would be good to add to learning diary with comments showing you know what is happening.

Other sorting algorithms

- There are numerous other sorting algorithms.
- Ours is OK for small data sets, but if we were dealing with larger datasets it would be very inefficient - for example if we have 1000 elements then there will be 1 000 000 iterations.
- A much more efficient sorting algorithm is the QuickSort algorithm - it takes roughly **NLogN** iterations to sort N elements. Thus for 1000 elements it would need to operate around 3000 times - a lot less than our hideously inefficient selection sort.

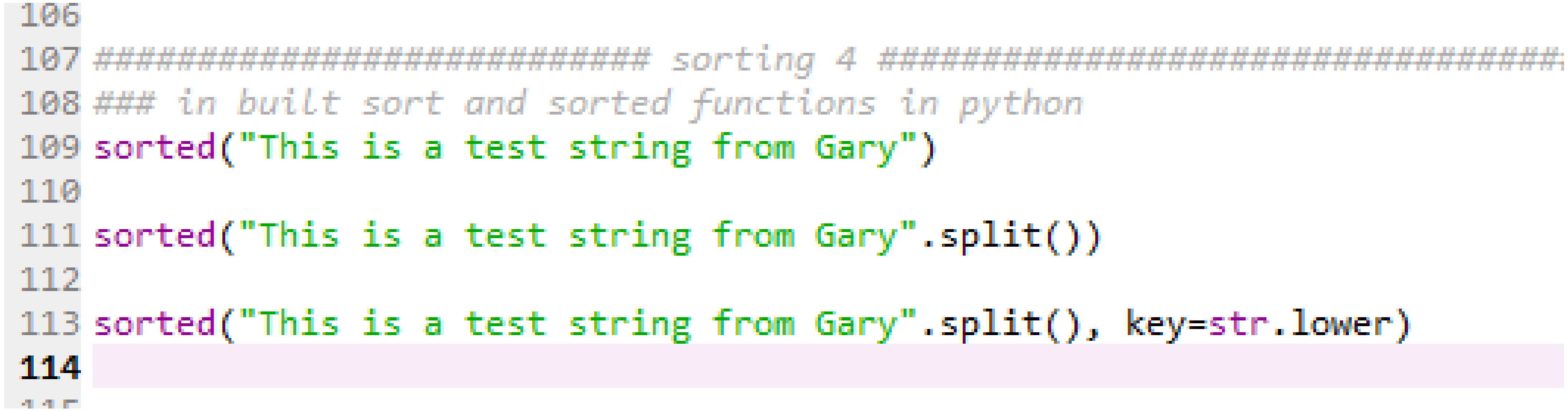
Sorting simple lists

- Python provides a built in way to sort data

```
list.sort()

##copy and paste this into the console to see what sorted function does
sorted([5,2,3,1,4])

#using sort() does the same thing. but sort() only works on lists, sorted can
#work on others
a=[5,2,3,1,4]
a.sort()
a
```

- Sorts the list from ‘low’ to high, if they are sortable items.
- This is a lot quicker than the code we have written!
- But… it was necessary to understand if you embark on more complex sorting processes.

You can also specify a key that specifies a function that is called on each list element before sorting



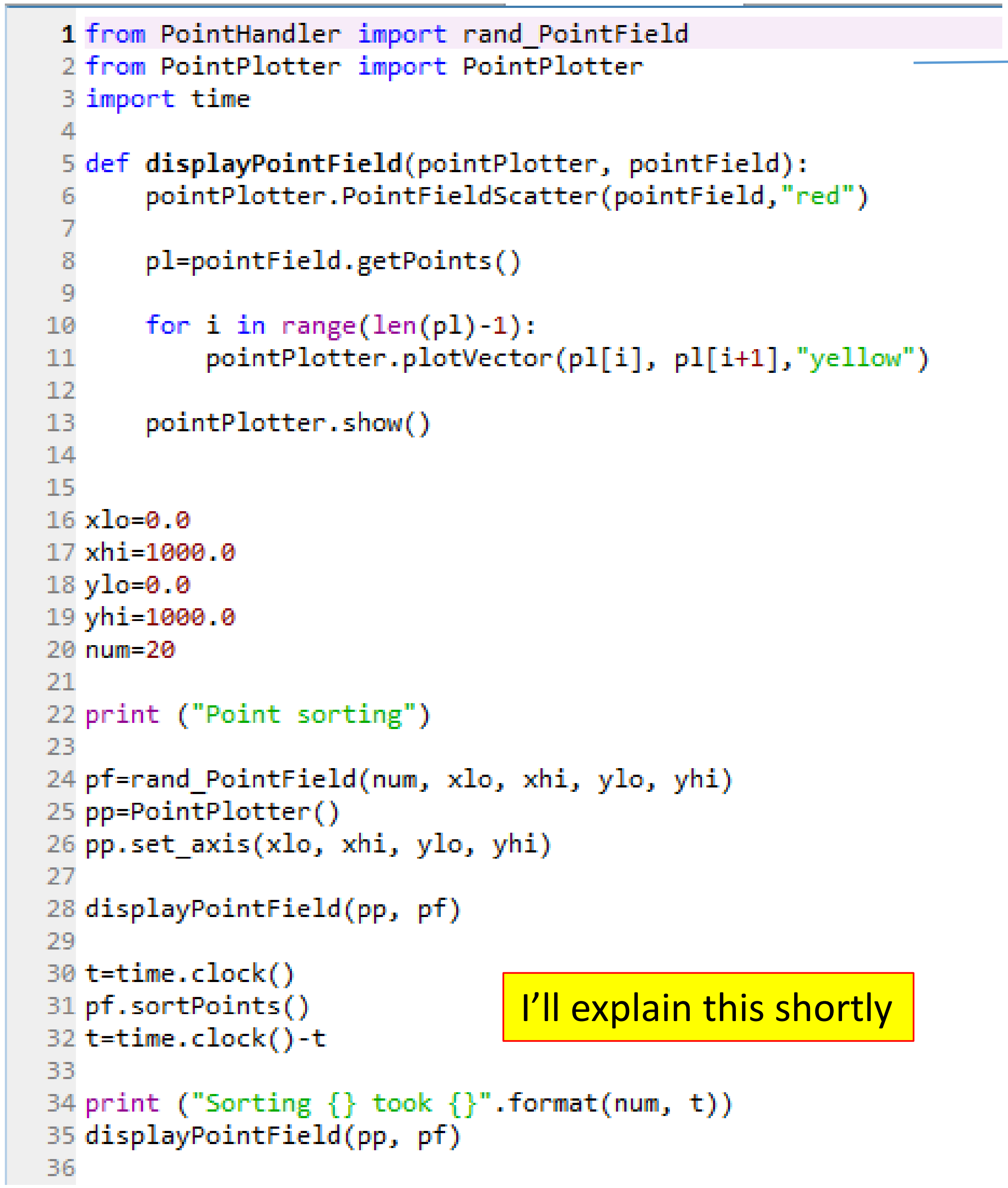
```
############################ sorting 4 ##################################
### in built sort and sorted functions in python
sorted("This is a test string from Gary")

sorted("This is a test string from Gary".split())

sorted("This is a test string from Gary".split(), key=str.lower)

```

What are each of these lines doing?

Coding Problem 3: Sorting spatial point objects

- Now you understand about sorting, try sorting Point objects – start by ordering them by their X-co-ordinate
- You can go back and produce a similar selection sort to the one you already had
- Where would such a method live most logically?
- What arguments would the method have? Would it return anything?
- You'll need a driver to test:

Example Driver

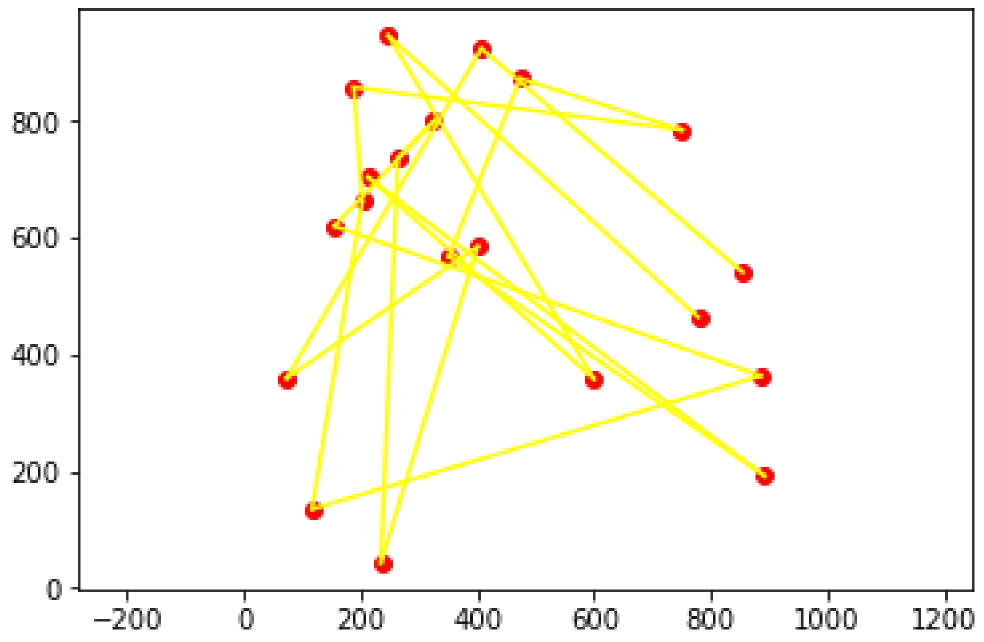


```python
from PointHandler import rand_PointField
from PointPlotter import PointPlotter
import time

def displayPointField(pointPlotter, pointField):
    pointPlotter.PointFieldScatter(pointField,"red")

    pl=pointField.getPoints()

    for i in range(len(pl)-1):
        pointPlotter.plotVector(pl[i], pl[i+1],"yellow")

    pointPlotter.show()


xlo=0.0
xhi=1000.0
ylo=0.0
yhi=1000.0
num=20

print ("Point sorting")

pf=rand_PointField(num, xlo, xhi, ylo, yhi)
pp=PointPlotter()
pp.set_axis(xlo, xhi, ylo, yhi)

displayPointField(pp, pf)

t=time.clock()
pf.sortPoints()
t=time.clock()-t

print ("Sorting {} took {}".format(num, t))
displayPointField(pp, pf)

```

Modules: Key component of Python

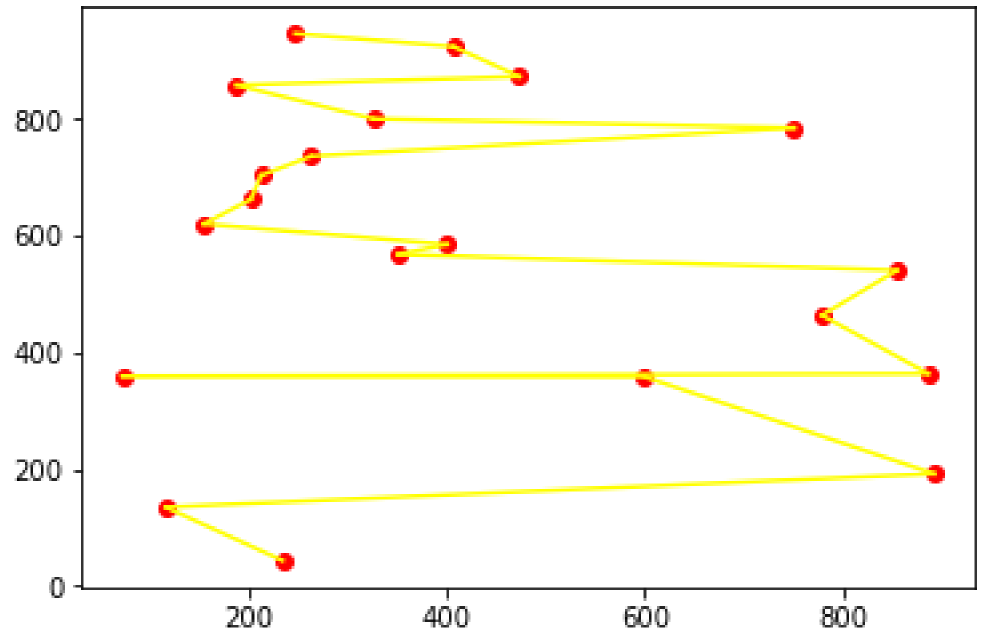
I'll explain this shortly

See Lecture1b.py for your version of this.



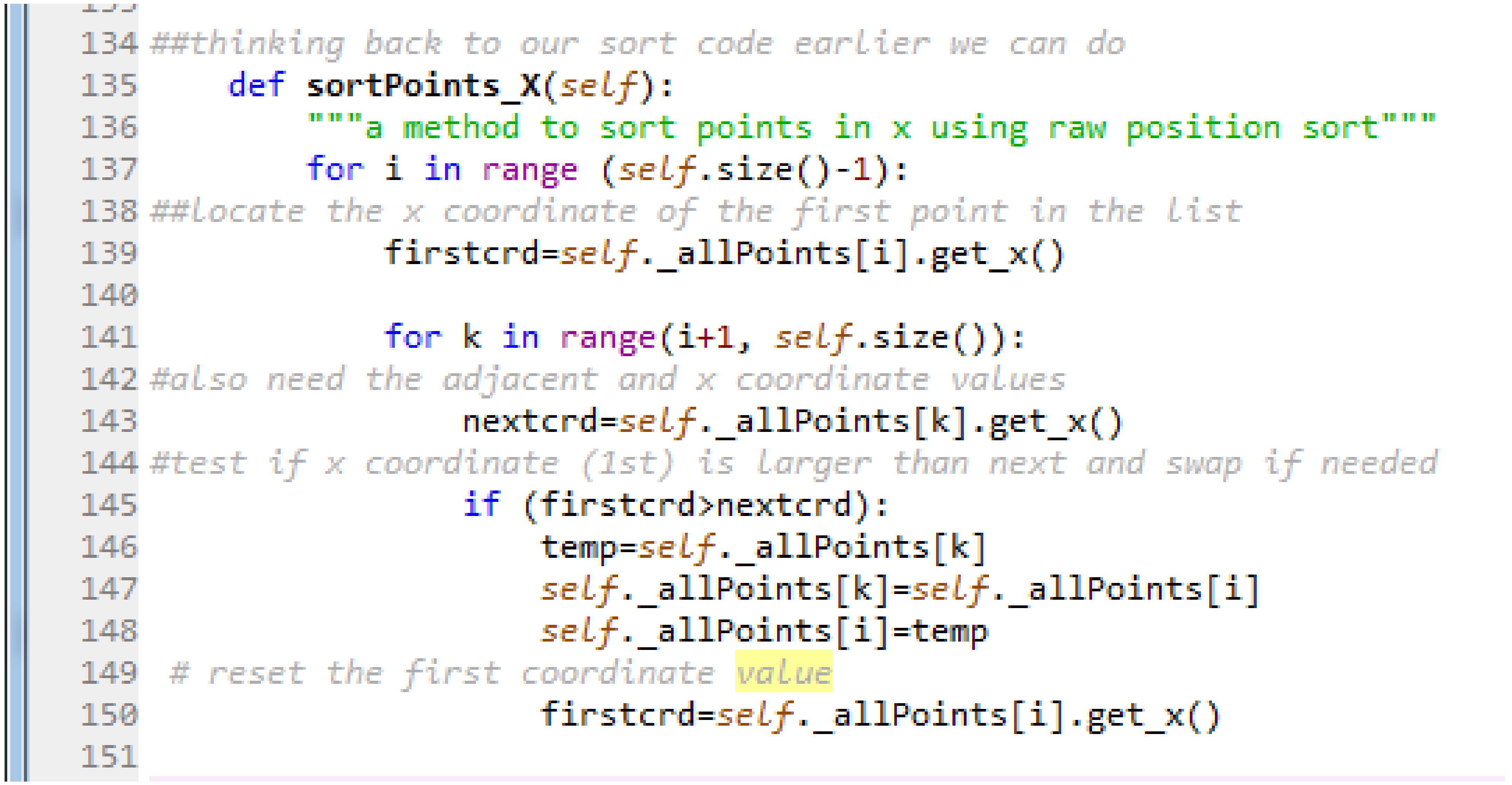
Before Sort

We want to plot these, the code we have given you will do this, once you have added in a sorting function for x or y



After Sort

Coding Problem 3: Sorting spatial point objects



```
##thinking back to our sort code earlier we can do
    def sortPoints_X(self):
        """a method to sort points in x using raw position sort"""
        for i in range (self.size()-1):
##locate the x coordinate of the first point in the list
            firstcrd=self._allPoints[i].get_x()

            for k in range(i+1, self.size()):
#also need the adjacent and x coordinate values
                nextcrd=self._allPoints[k].get_x()
#test if x coordinate (1st) is larger than next and swap if needed
                if (firstcrd>nextcrd):
                    temp=self._allPoints[k]
                    self._allPoints[k]=self._allPoints[i]
                    self._allPoints[i]=temp
 # reset the first coordinate value
                    firstcrd=self._allPoints[i].get_x()

```

Coding problem 3

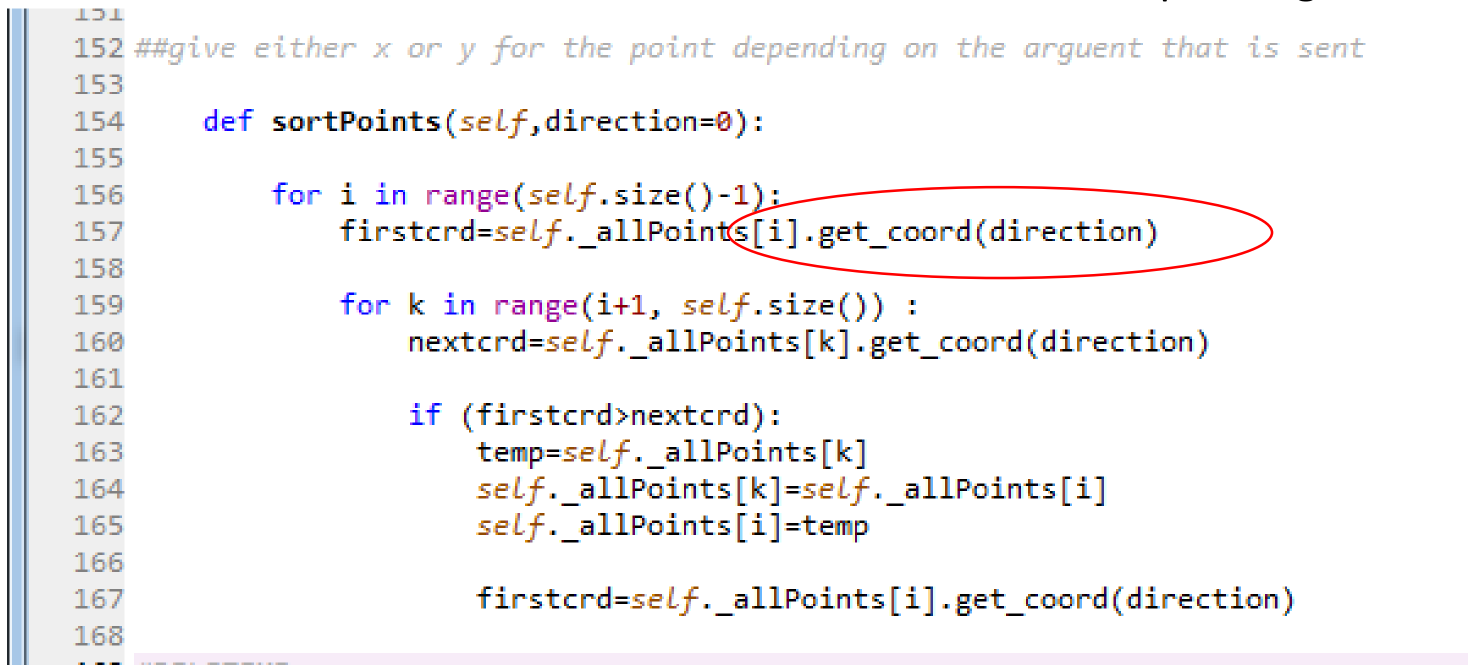
- Why is there an error returned when running Lecture1b.py?
- Sort function needs adding
- Where do you put it?

- The clue is in the Lecture1b.py slide

- Spend some time thinking about this – we will look over it in a few minutes.

Having specific x and y direction sorts is not very elegant. Better to create a method that could handle either direction

Secret is to define a new method for Point that will give either X or Y for the point, depending on the argument you send



```
##give either x or y for the point depending on the arguent that is sent

    def sortPoints(self,direction=0):

        for i in range(self.size()-1):
            firstcrd=self._allPoints[i].get_coord(direction)

            for k in range(i+1, self.size()) :
                nextcrd=self._allPoints[k].get_coord(direction)

                if (firstcrd>nextcrd):
                    temp=self._allPoints[k]
                    self._allPoints[k]=self._allPoints[i]
                    self._allPoints[i]=temp

                    firstcrd=self._allPoints[i].get_coord(direction)

```

In-built python sort algorithm: Sorting more complex lists..

```
list.sort()
```

- Sorts a simple list from low to high, if they are sortable items.
- How would you sort a list of Points?
- If you can write a function to say what is a ‘high’ or ‘low’ point you can still use the sort() function provided by Python

Sorting more complex lists..

`list.sort(`*`myFunction`*`)`

- If you provide `sort()` with an argument of a function name then, if the function operates on any two items from the list and returns

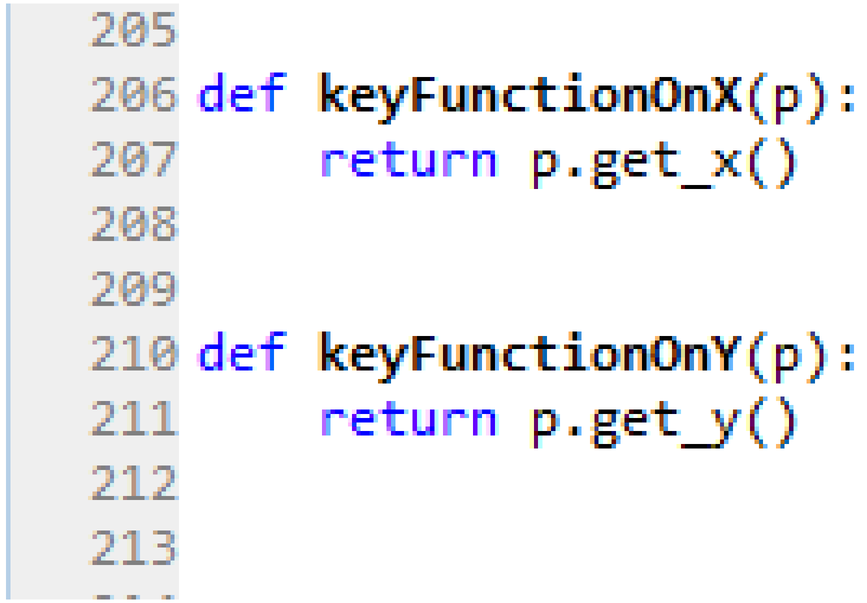
  -1, if the first is ‘less than’ the second

  0, if they are the same

  1, if the second is ‘less than’ the first

  `sort` will work with any kind of list

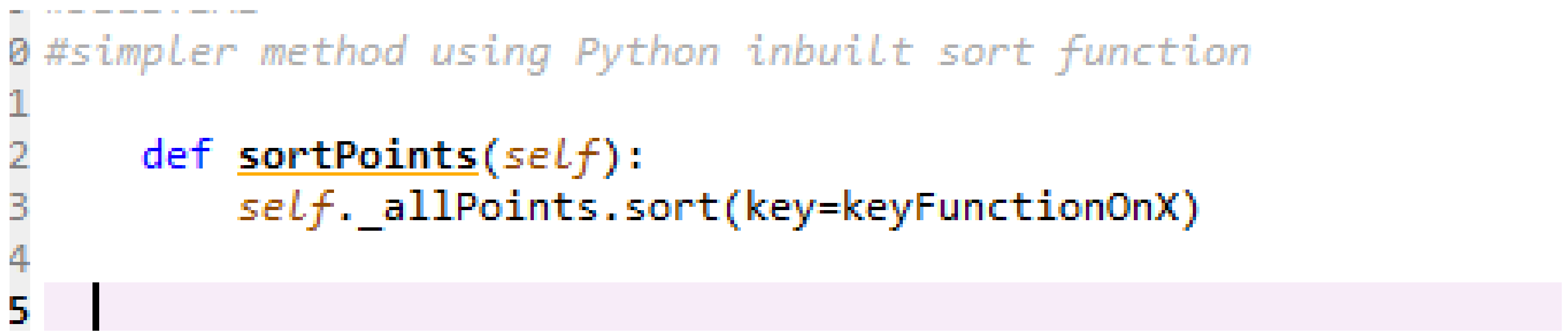
Sort function example for points



```python

def keyFunctionOnX(p):
    return p.get_x()


def keyFunctionOnY(p):
    return p.get_y()


```

Where would this go?

And your PointField sorting method would look like this



```
#simpler method using Python inbuilt sort function

    def sortPoints(self):
        self._allPoints.sort(key=keyFunctionOnX)

```

You also might like to try timing this vs. your own versions for e.g. 20 and then 2000 points

Run the timer for both versions of the code and discuss in the learning diary. Which was faster, why?

Any Questions?

Efficient code and functions

- You can time how long a piece of code takes using the inbuilt system clock. This can be useful if you want to make your code run faster because you can see which bits take the most time overall

```
    time.clock()
```

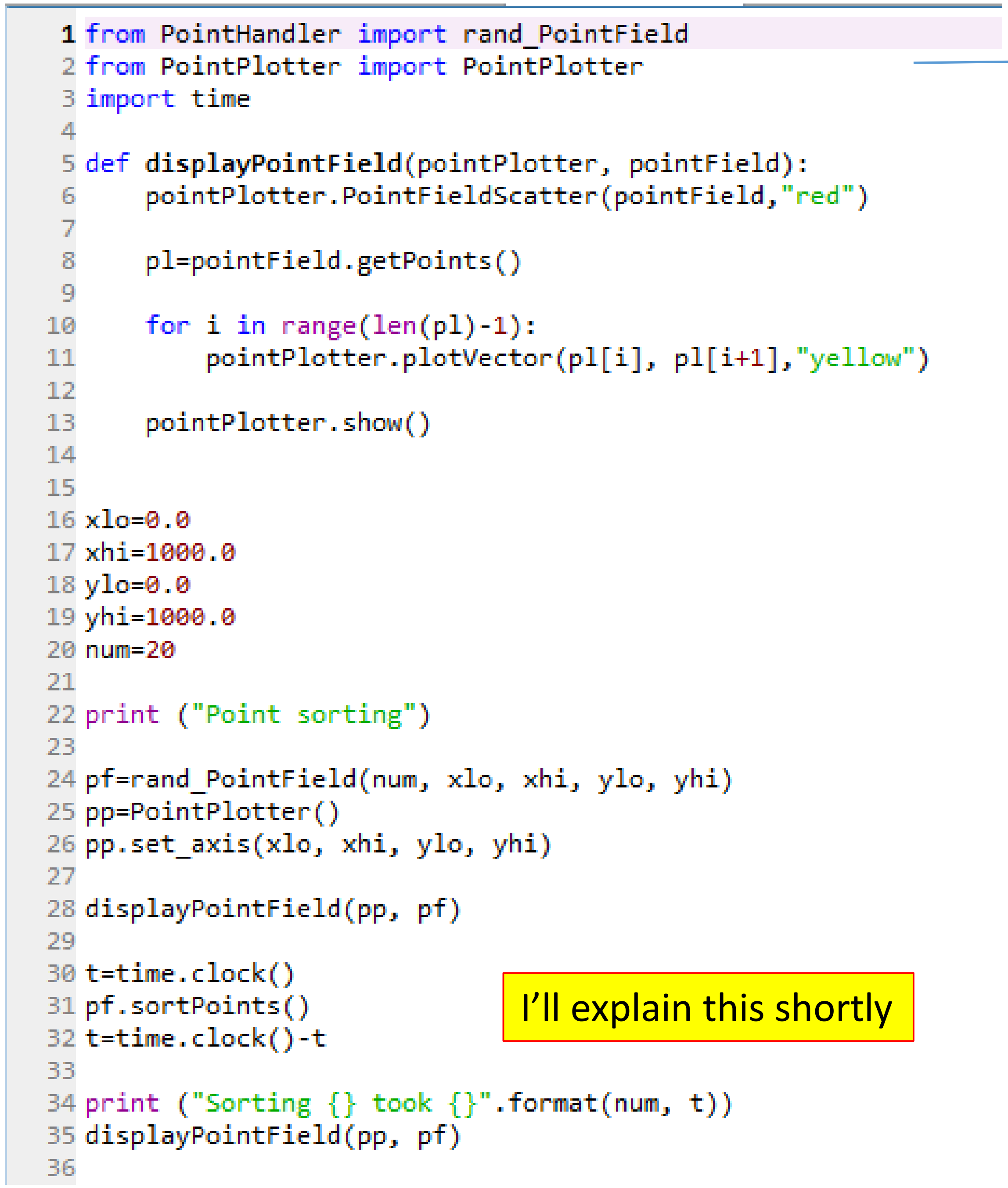
- Returns the system time, and so

```
t=time.clock()
the code block I’m trying to time
t=time.clock()-t
print ("Calculation time "+str(t)+" seconds“)
```

Will let you know how long a particular code sequence has taken

You need to `import time` to make this work

Example Driver



```python
from PointHandler import rand_PointField
from PointPlotter import PointPlotter
import time

def displayPointField(pointPlotter, pointField):
    pointPlotter.PointFieldScatter(pointField,"red")

    pl=pointField.getPoints()

    for i in range(len(pl)-1):
        pointPlotter.plotVector(pl[i], pl[i+1],"yellow")

    pointPlotter.show()


xlo=0.0
xhi=1000.0
ylo=0.0
yhi=1000.0
num=20

print ("Point sorting")

pf=rand_PointField(num, xlo, xhi, ylo, yhi)
pp=PointPlotter()
pp.set_axis(xlo, xhi, ylo, yhi)

displayPointField(pp, pf)

t=time.clock()
pf.sortPoints()
t=time.clock()-t

print ("Sorting {} took {}".format(num, t))
displayPointField(pp, pf)
```

Modules: Key component of Python

I’ll explain this shortly

See Lecture1b.py for your version of this.

Summary

- From these examples you see that many spatial algorithms involve either *searching*, *sorting* them or *traversing* data sets
- These concepts are all related

Expressing Algorithms

- English
- Pseudo-code
- Python

Use all three when writing your learning diary.

Take Home message

- An idea is at the heart of any algorithm – we should be able to see the idea
- if idea is not clearly written when expressing algorithm the notation needs improving.

Exercise for this week

- Implement a point sorting function to sort a field of points either by x or y co-ordinates.
- Add a commented example to your learning blog on Learn
- Then summarise a brief (<500 words) reflection on the weeks workshop and reading, possible discussion topics:
  - what did you learn, what wasn’t clear to you?
  - How could you see these things being used in problem solving