Week 3b: list comprehension, list indexing and slicing, import

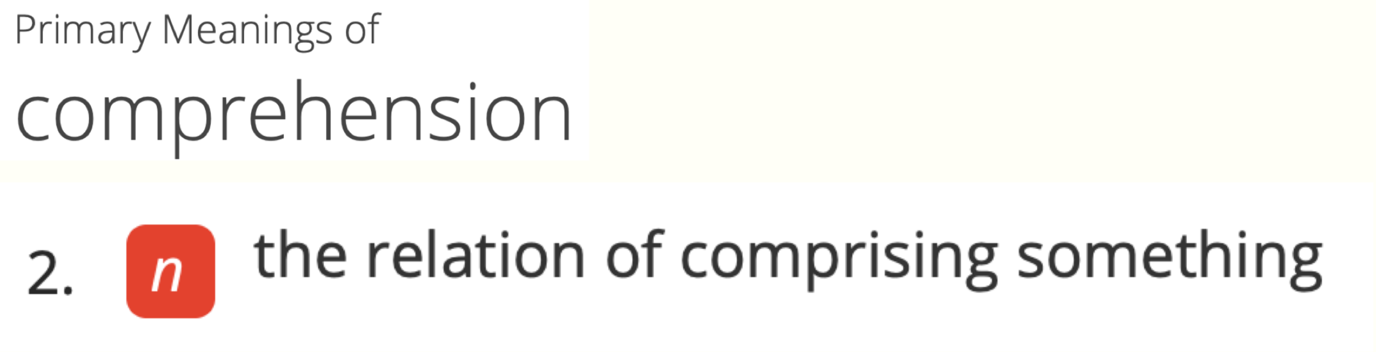
Professor Aaron Quigley

Thanks to Chun Tung Chou

and Ashesh Mahidadia

Week 3B

- List comprehension
- Lists
  - Indexing
  - Slicing lists
- Import



Primary Meanings of

comprehension

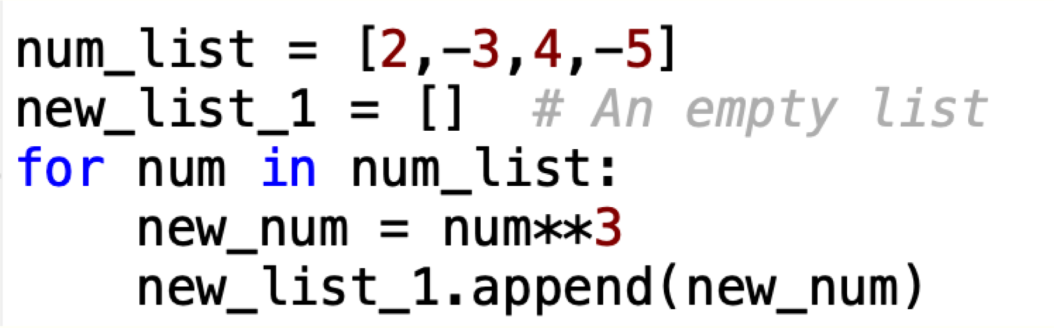
2. n the relation of comprising something

Construction

https://www.vocabulary.com/dictionary/comprehension

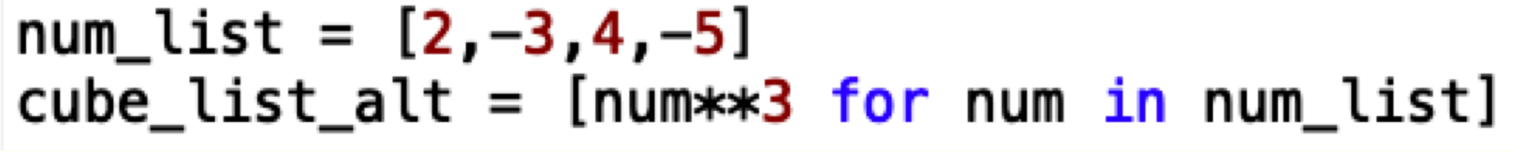
List comprehension

- A concise method to create a list from another list
- E.g., compute the cube of each element in the list
  - Using .append() [Example 1 in create_list_ex.py]



```
num_list = [2,-3,4,-5]
new_list_1 = []   # An empty list
for num in num_list:
    new_num = num**3
    new_list_1.append(new_num)
```

  - Using list comprehension



```
num_list = [2,-3,4,-5]
cube_list_alt = [num**3 for num in num_list]
```

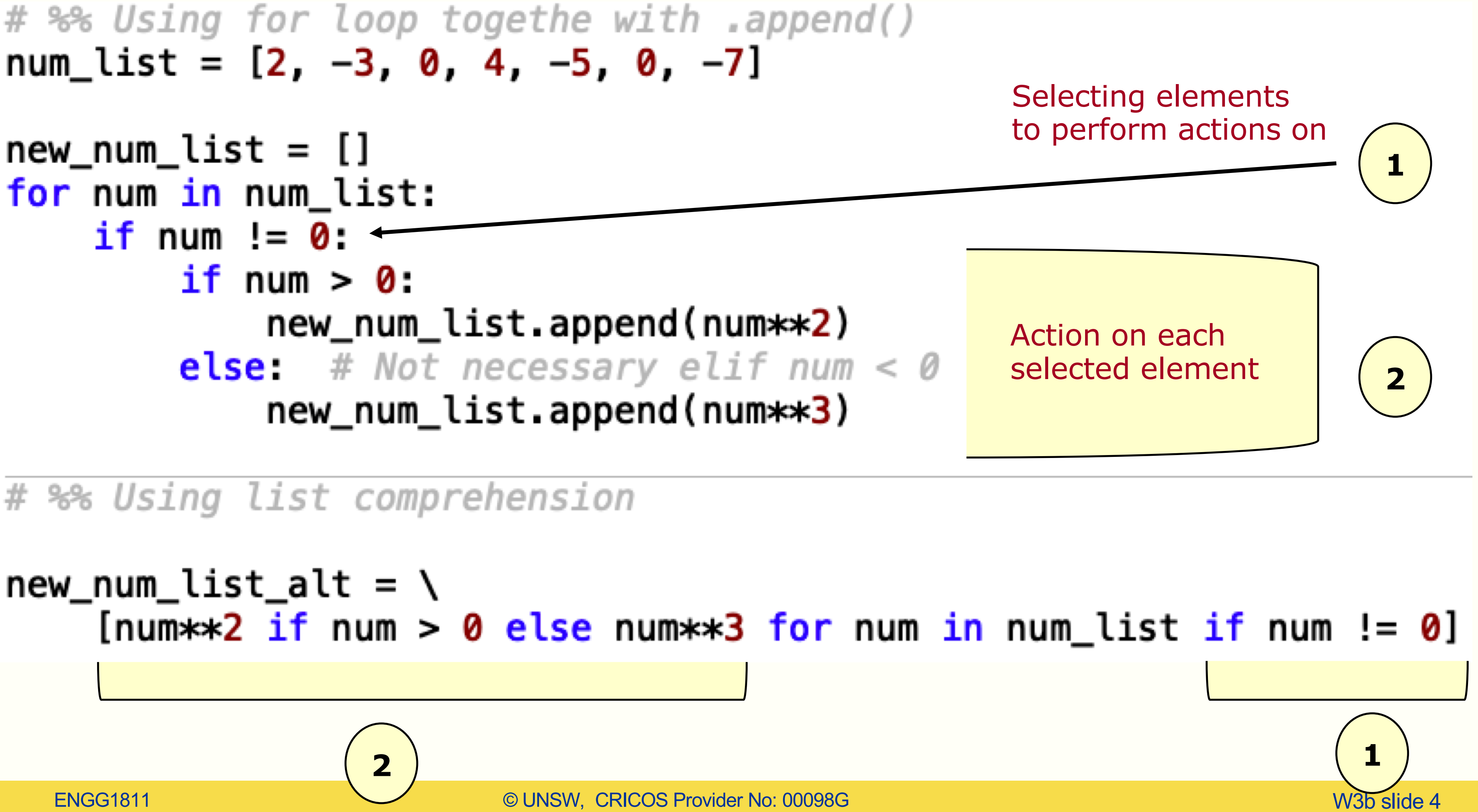
for statement

Action to be applied
to the elements in the given list

- More examples in create_list_with_comprehension.py

List comprehension: general format

- Code in list_comprehension_general.py



```
# %% Using for loop togethe with .append()
num_list = [2, -3, 0, 4, -5, 0, -7]

new_num_list = []
for num in num_list:
    if num != 0:
        if num > 0:
            new_num_list.append(num**2)
        else:  # Not necessary elif num < 0
            new_num_list.append(num**3)

# %% Using list comprehension

new_num_list_alt = \
    [num**2 if num > 0 else num**3 for num in num_list if num != 0]
```

Selecting elements
to perform actions on

Action on each
selected element

Project: use list comprehension

- You did this project in Week 3A
- If you drop an object of mass *m* in a medium with drag coefficient *d* and acceleration due to gravity *g*, then the object's speed *v(t)* at time *t* is given by:

$$v(t) = \frac{gm}{d}\left(1 - e^{-\frac{d}{m}t}\right)$$

- Given the numerical value of *m*, *g* and *d*, the goal of the project is to plot *v(t)* against *t*
  - for t = 0, 0.5, 1, 1.5, …., 39.5, 40

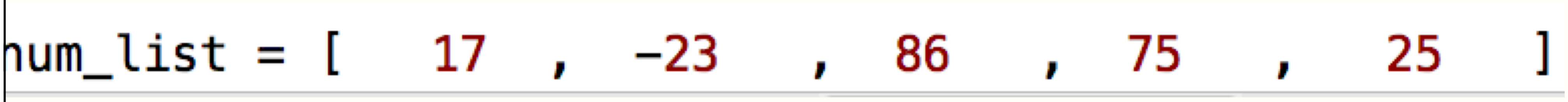
- **Re-do the project using list comprehension**

Week 3B

- List comprehension
- Lists
  - Indexing
  - Slicing lists
- Import

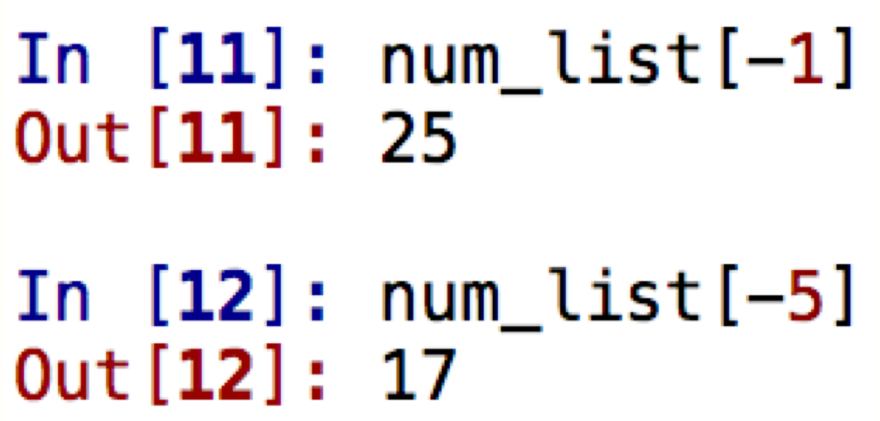
List indexing

- Each element in the list can be indexed in two ways



```
num_list = [   17   ,  -23    ,  86    ,  75    ,    25    ]
```

Index	0	1	2	3	4
	-5	-4	-3	-2	-1



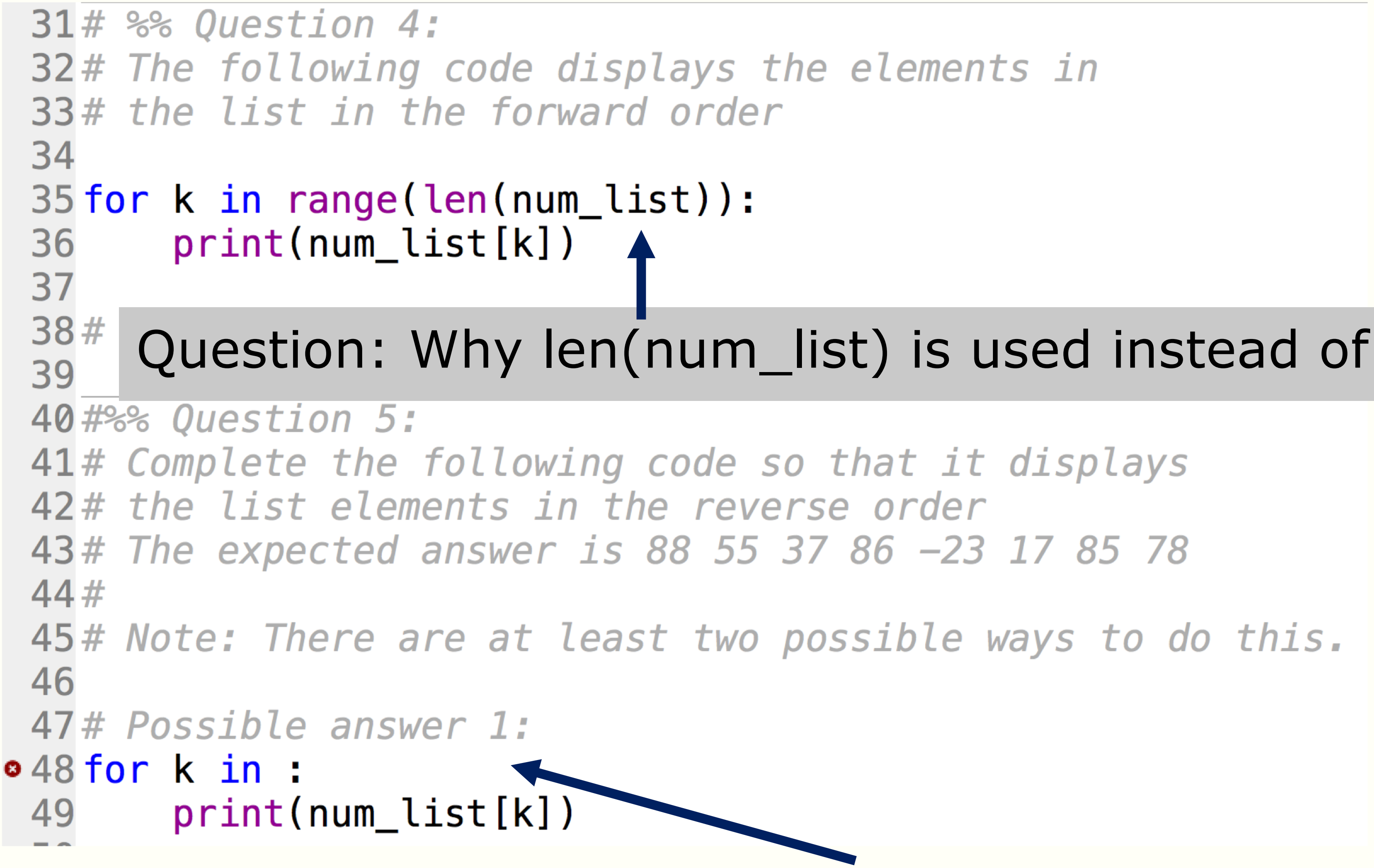
```
In [11]: num_list[-1]
Out[11]: 25

In [12]: num_list[-5]
Out[12]: 17
```

The index starts at 0. There are reasons for that, you'll see later.

Quiz

5 questions in quiz_indexing.py.

Questions 4 and 5 are shown below.



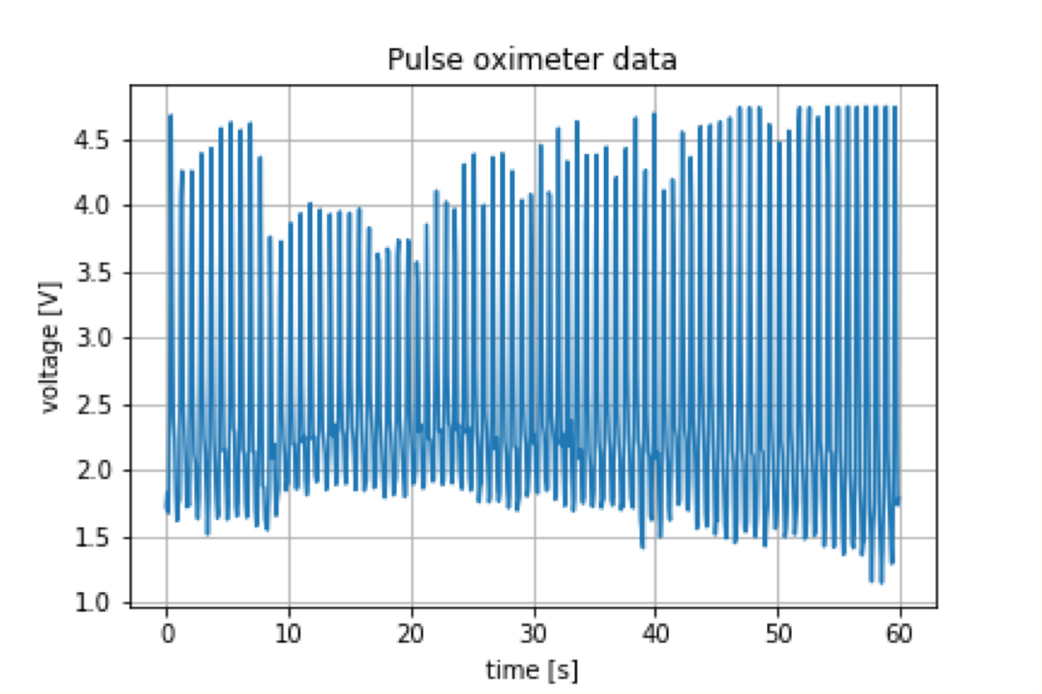
```
# %% Question 4:
# The following code displays the elements in
# the list in the forward order

for k in range(len(num_list)):
    print(num_list[k])

#

#%% Question 5:
# Complete the following code so that it displays
# the list elements in the reverse order
# The expected answer is 88 55 37 86 -23 17 85 78
#
# Note: There are at least two possible ways to do this.

# Possible answer 1:
for k in :
    print(num_list[k])
```

Question: Why len(num_list) is used instead of 8?

Question: Complete this code here

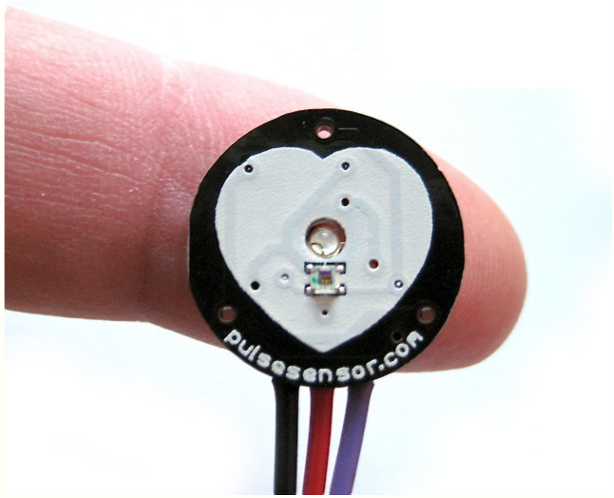
Slicing - motivation

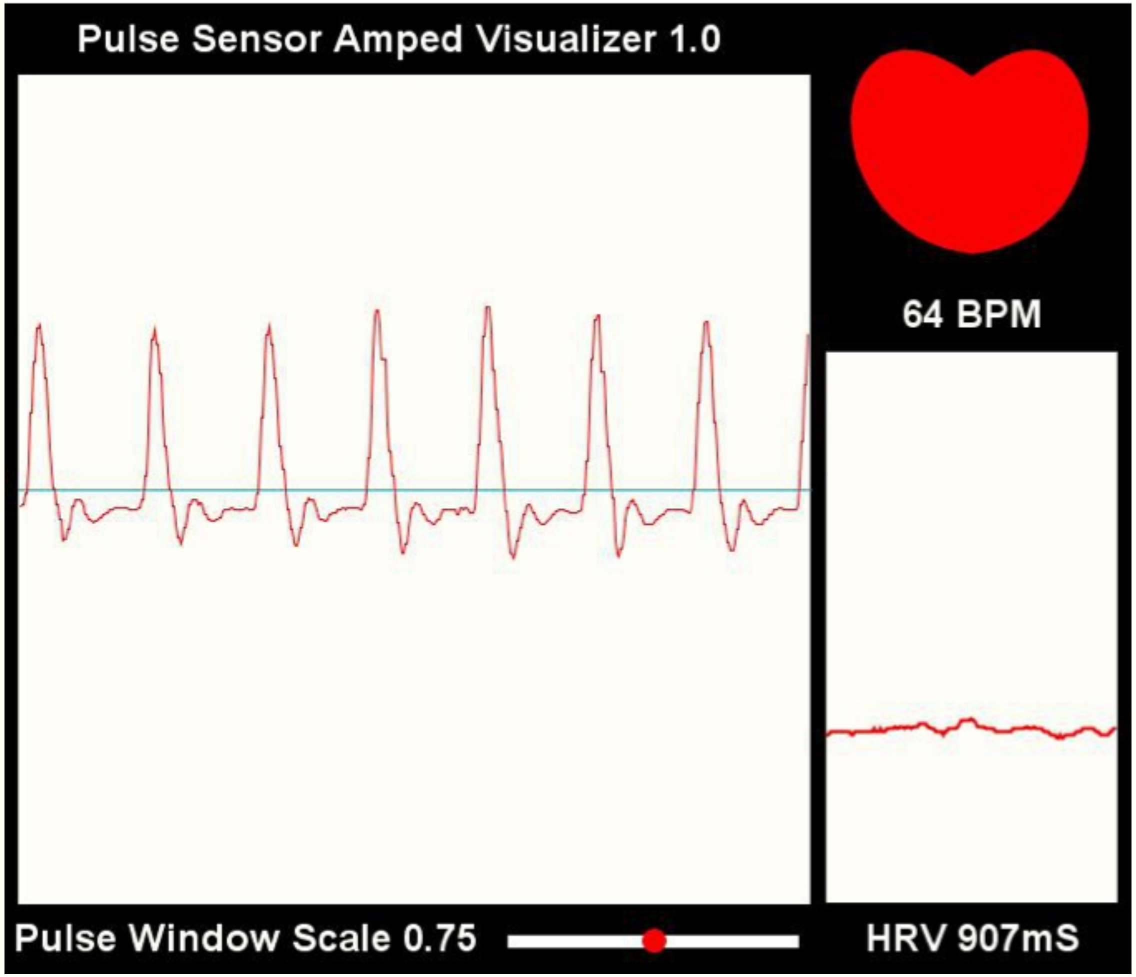
- Sometimes you may want to work on a section of a list
- Motivation:
  - Remember you can use a list to store a data sequence
  - You have graphed the data and you find a section of data interesting
  - You can use slicing to get a section of data and graph only that section



Pulse oximeter

Pulse oximetry sensor





Pulse Sensor Amped Visualizer 1.0

64 BPM

Pulse Window Scale 0.75

HRV 907mS

http://pulsesensor.com

Slicing a list

- We will use the following list to illustrate slicing

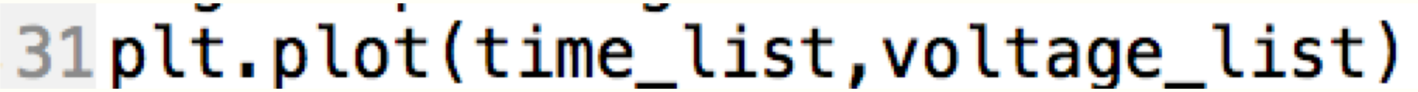
```
num_list = [  17  , -23  , 86  ,  37  ,  55  ,  76  ,  -91  ]
```

0	**1**	**2**	**3**	**4**	**5**	**6**
-7	**-6**	**-5**	**-4**	**-3**	**-2**	**-1**

- We will use the file slicing_example.py and type commands into the console

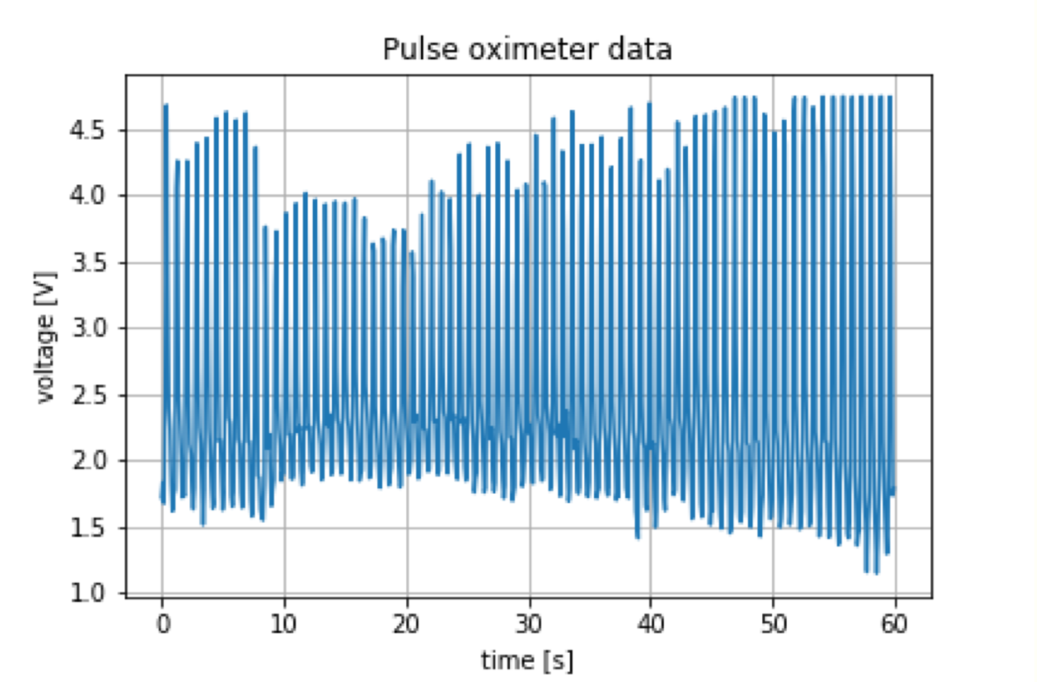
Exercise: Slicing and graphing (1)

- The file quiz_slicing.py contains the code to load and plot data obtained from a pulse oximeter
- The code produces the graph below
- Line 31 of the code does the plotting



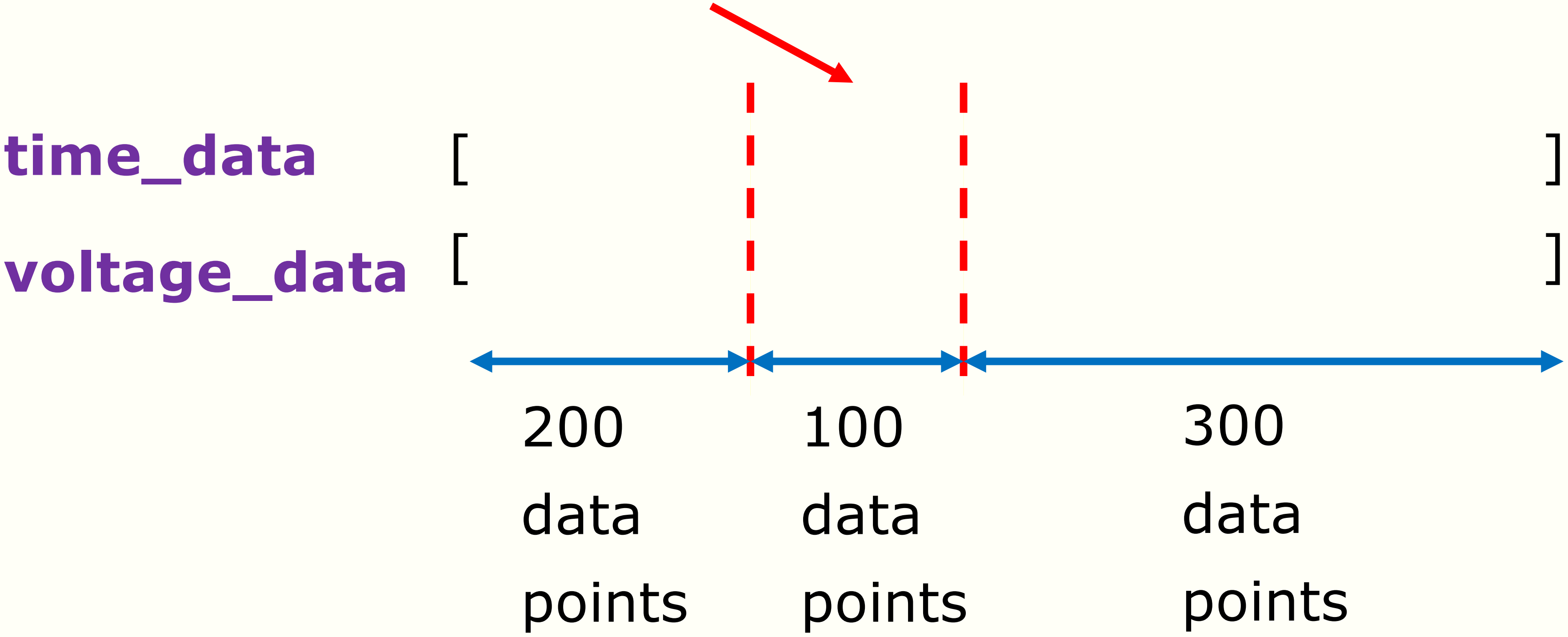
```
31 plt.plot(time_list,voltage_list)
```

- Both time_list and voltage_list are lists with 600 elements



Exercise: Slicing and graphing (2)

- You want to plot **this section** of the data



time_data [ ]

voltage_data [ ]

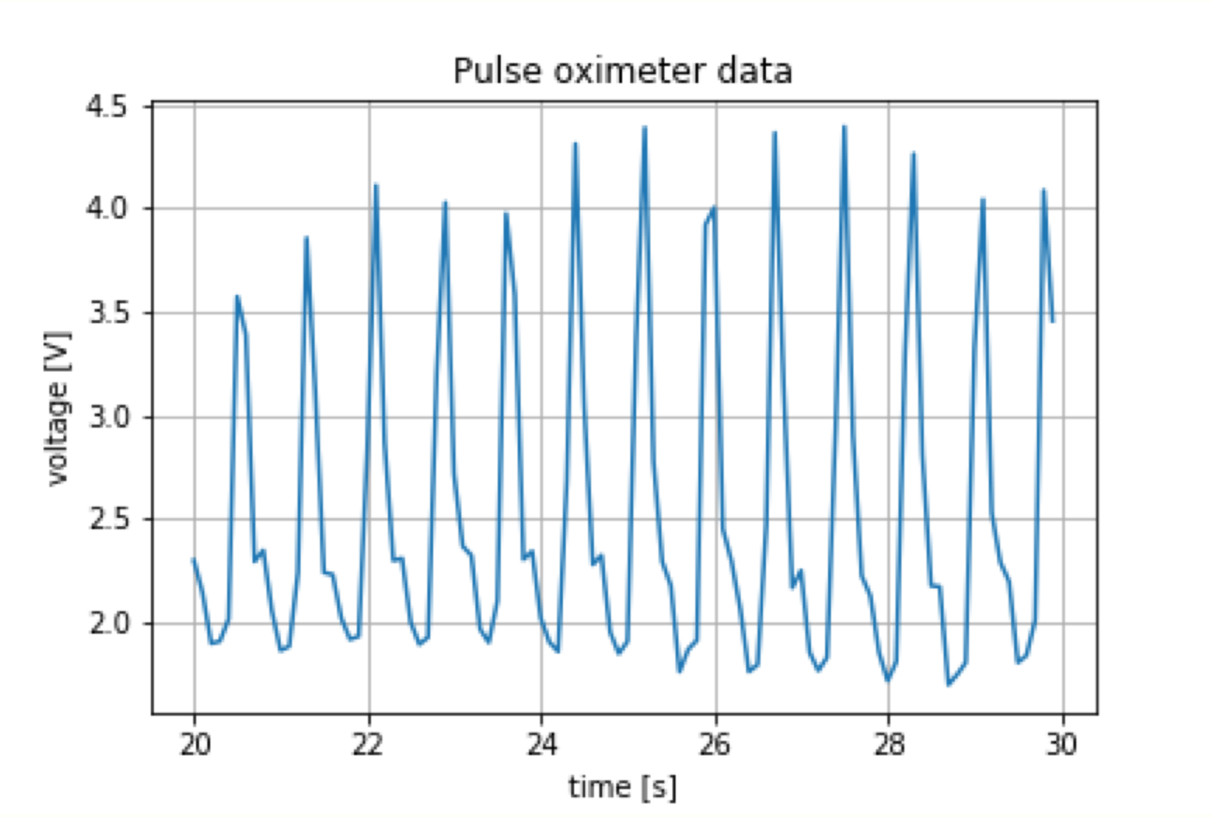
200 data points

100 data points

300 data points

- Exercise: Modify Line 31 to realise your goal
  - You can see what the graph should look like on the next slide

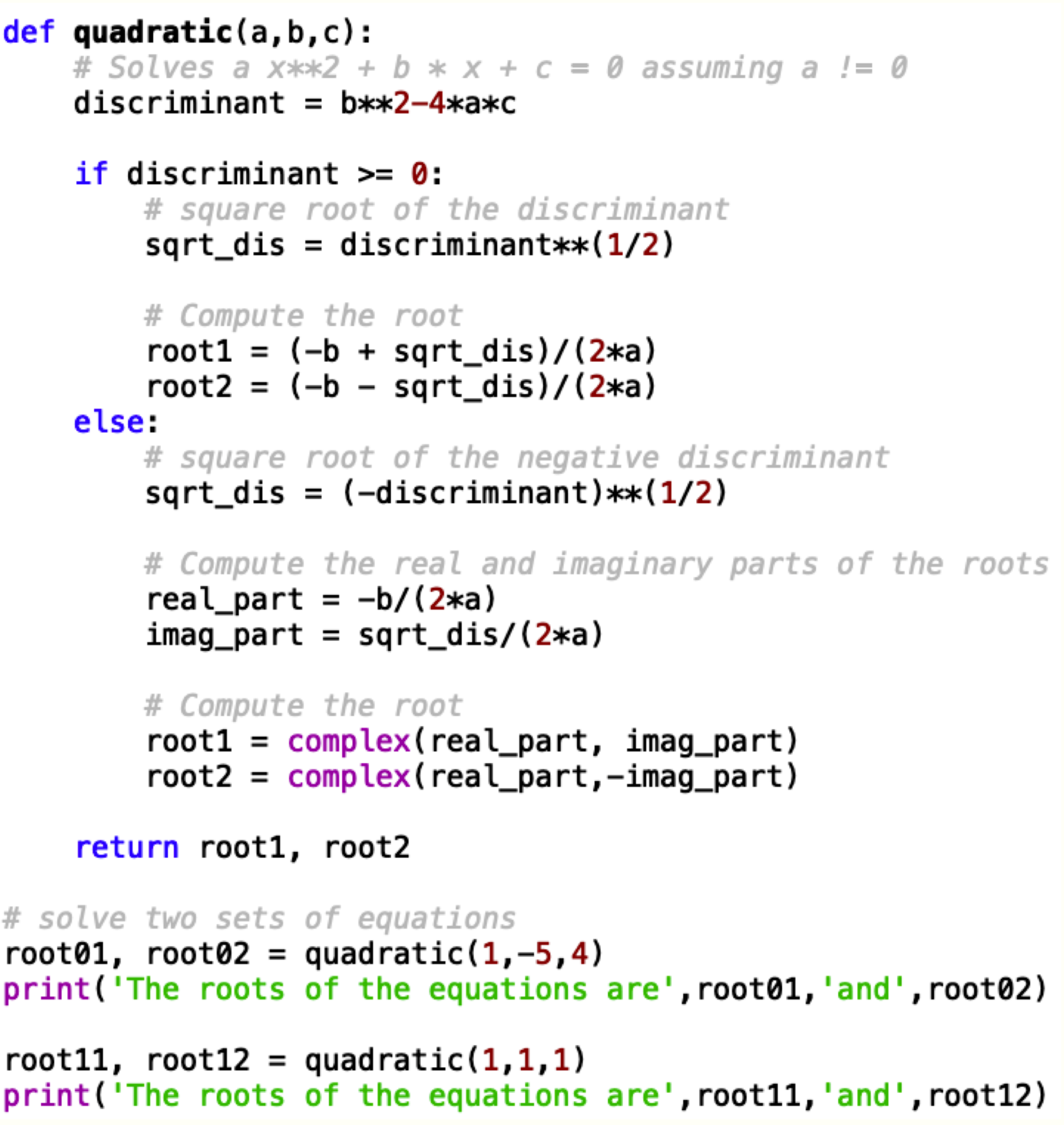
Exercise: Slicing and graphing (3)



Week 3B

- List comprehension
- Lists
  - Indexing
  - Slicing lists
- Import

Motivating import



```python
def quadratic(a,b,c):
    # Solves a x**2 + b * x + c = 0 assuming a != 0
    discriminant = b**2-4*a*c

    if discriminant >= 0:
        # square root of the discriminant
        sqrt_dis = discriminant**(1/2)

        # Compute the root
        root1 = (-b + sqrt_dis)/(2*a)
        root2 = (-b - sqrt_dis)/(2*a)
    else:
        # square root of the negative discriminant
        sqrt_dis = (-discriminant)**(1/2)

        # Compute the real and imaginary parts of the roots
        real_part = -b/(2*a)
        imag_part = sqrt_dis/(2*a)

        # Compute the root
        root1 = complex(real_part, imag_part)
        root2 = complex(real_part,-imag_part)

    return root1, root2

# solve two sets of equations
root01, root02 = quadratic(1,-5,4)
print('The roots of the equations are',root01,'and',root02)

root11, root12 = quadratic(1,1,1)
print('The roots of the equations are',root11,'and',root12)
```

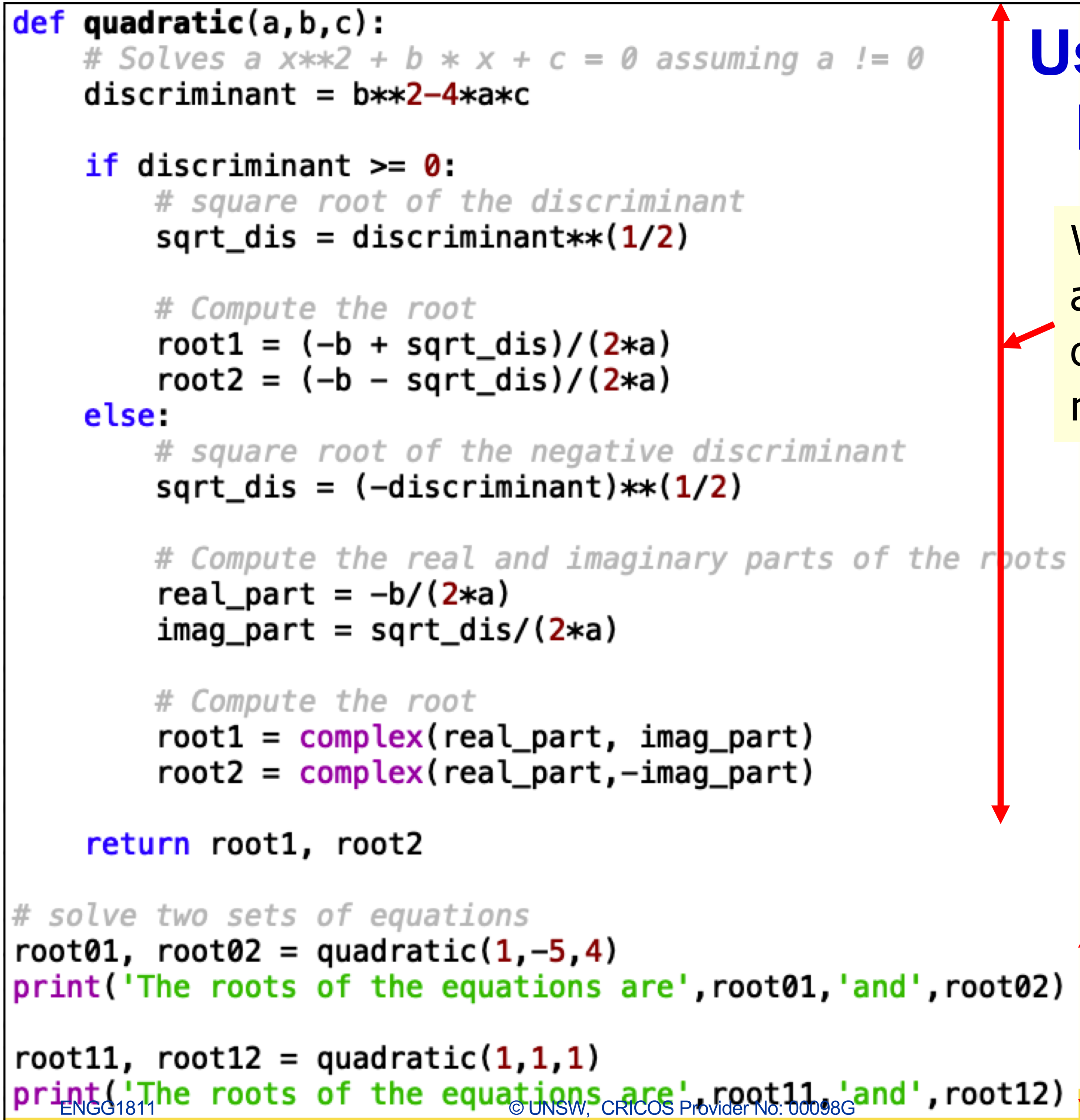
A function to solve a quadratic equation

How can you make this function available to other Python programs?

Bad idea: Copy the code to other files. Need to maintain multiple copies of code.

Better idea: Maintain one copy of the code and use import.

Using separate Python files



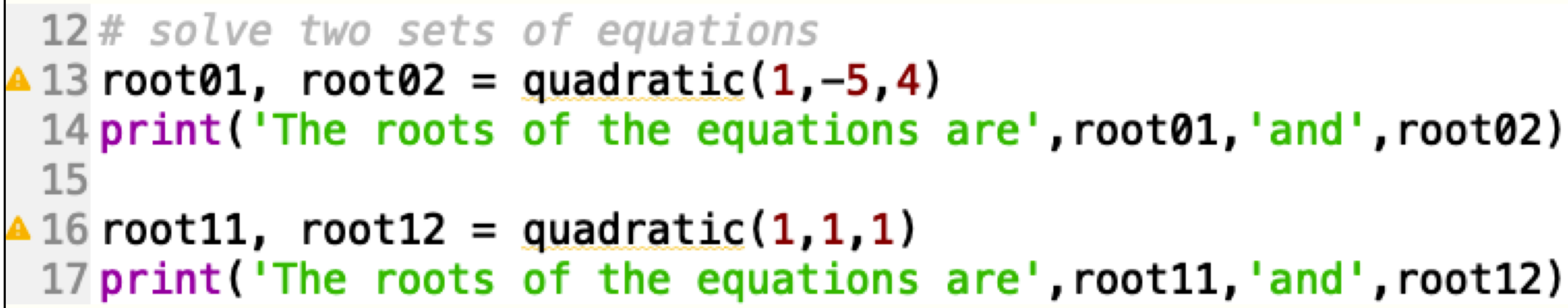
```
def quadratic(a,b,c):
    # Solves a x**2 + b * x + c = 0 assuming a != 0
    discriminant = b**2-4*a*c

    if discriminant >= 0:
        # square root of the discriminant
        sqrt_dis = discriminant**(1/2)

        # Compute the root
        root1 = (-b + sqrt_dis)/(2*a)
        root2 = (-b - sqrt_dis)/(2*a)
    else:
        # square root of the negative discriminant
        sqrt_dis = (-discriminant)**(1/2)

        # Compute the real and imaginary parts of the roots
        real_part = -b/(2*a)
        imag_part = sqrt_dis/(2*a)

        # Compute the root
        root1 = complex(real_part, imag_part)
        root2 = complex(real_part,-imag_part)

    return root1, root2

# solve two sets of equations
root01, root02 = quadratic(1,-5,4)
print('The roots of the equations are',root01,'and',root02)

root11, root12 = quadratic(1,1,1)
print('The roots of the equations are',root11,'and',root12)
```

We have copied and saved this part of the code in my_lib.py

We have saved this part of the code in use_import_prelim.py

Getting to use import (1)

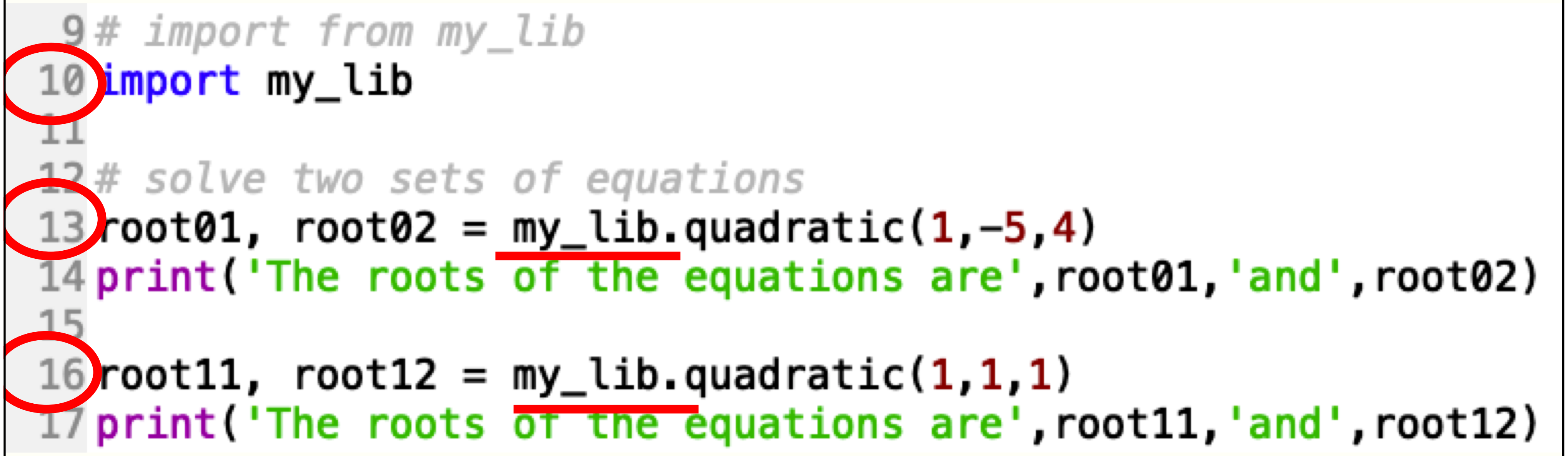
- Open the file use_import_prelim.py
- The editor complains about Lines 13 and 16 because the function quadratic() cannot be found



```
 12 # solve two sets of equations
⚠13 root01, root02 = quadratic(1,-5,4)
 14 print('The roots of the equations are',root01,'and',root02)
 15
⚠16 root11, root12 = quadratic(1,1,1)
 17 print('The roots of the equations are',root11,'and',root12)
```

Getting to use import (2)

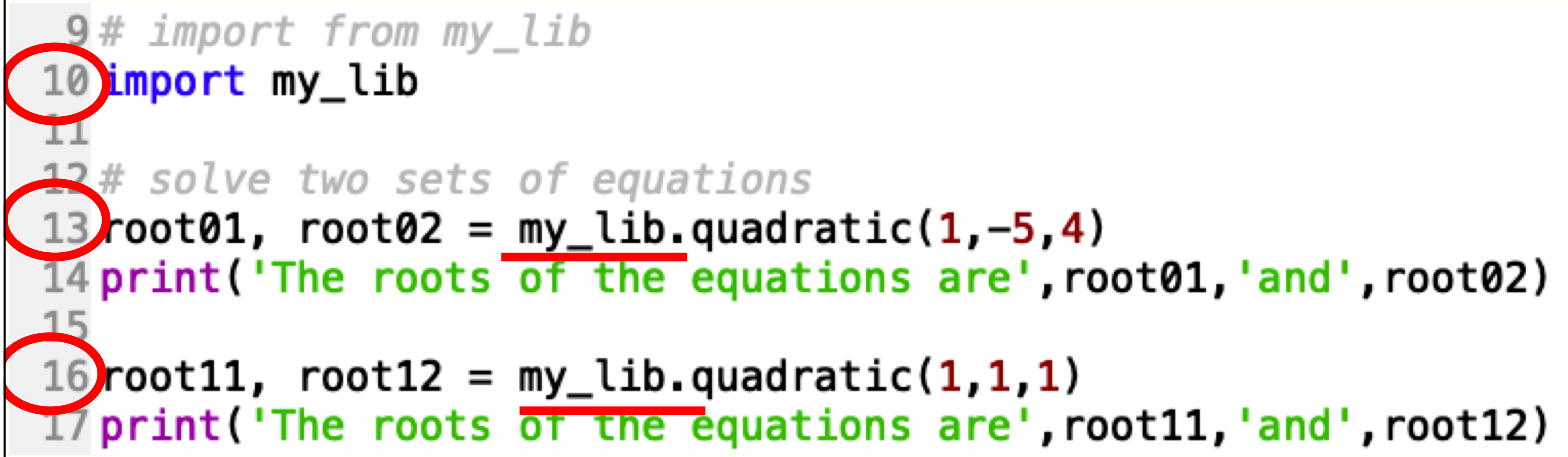
- Add Line 10
- Modify Lines 13 and 16 as follows
- Save and run the program



```
 9 # import from my_lib
10 import my_lib
11
12 # solve two sets of equations
13 root01, root02 = my_lib.quadratic(1,-5,4)
14 print('The roots of the equations are',root01,'and',root02)
15
16 root11, root12 = my_lib.quadratic(1,1,1)
17 print('The roots of the equations are',root11,'and',root12)
```

What does import do?

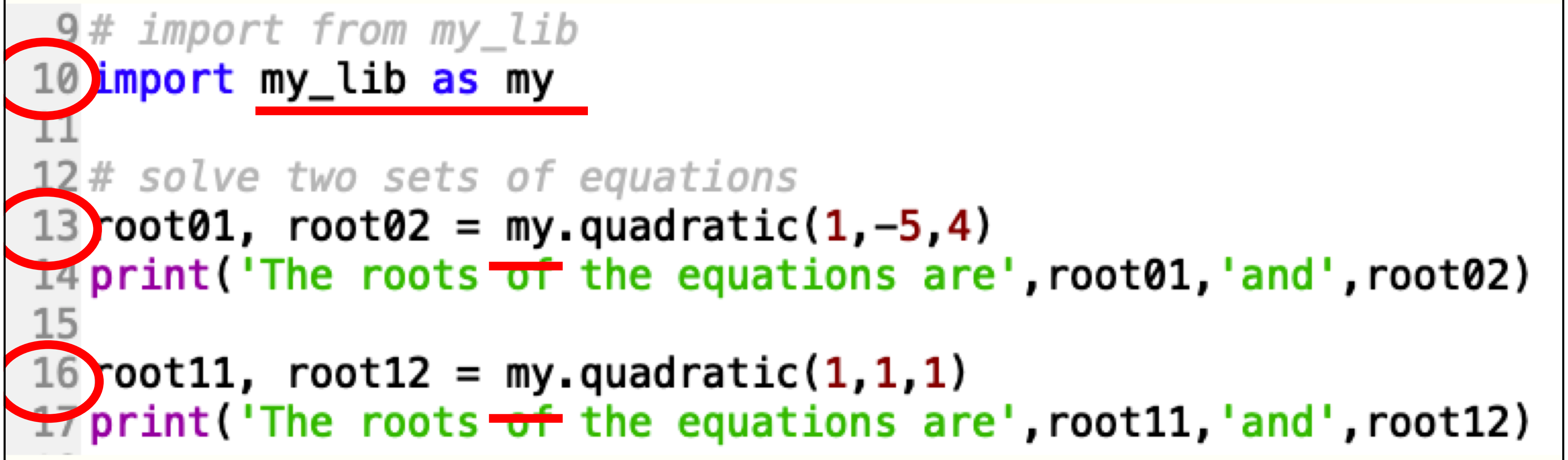
- The keyword **import** tells Python to include the functions in my_lib.py as part of this code
- You can read the code and comment to understand the flow of the program
- Good to add a comment to explain what functions you want to be imported



```python
# import from my_lib
import my_lib

# solve two sets of equations
root01, root02 = my_lib.quadratic(1,-5,4)
print('The roots of the equations are',root01,'and',root02)

root11, root12 = my_lib.quadratic(1,1,1)
print('The roots of the equations are',root11,'and',root12)
```

Another way to use import

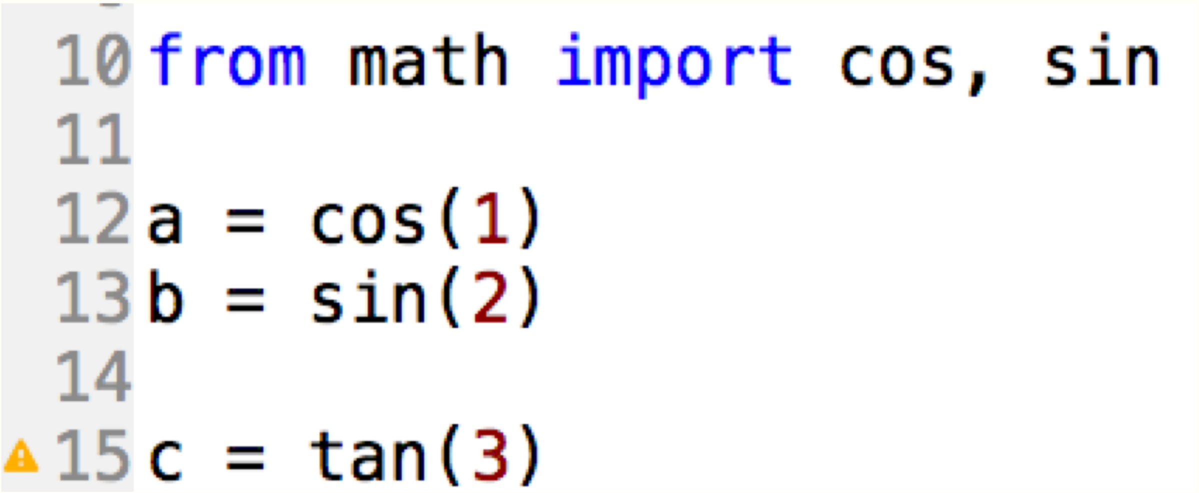
- The changes are in Lines 9 and 23
- You can define a short form to use
  - Have you seen import as before?



```
 9 # import from my_lib
10 import my_lib as my
11
12 # solve two sets of equations
13 root01, root02 = my.quadratic(1,-5,4)
14 print('The roots of the equations are',root01,'and',root02)
15
16 root11, root12 = my.quadratic(1,1,1)
17 print('The roots of the equations are',root11,'and',root12)
```

Importing selected functions

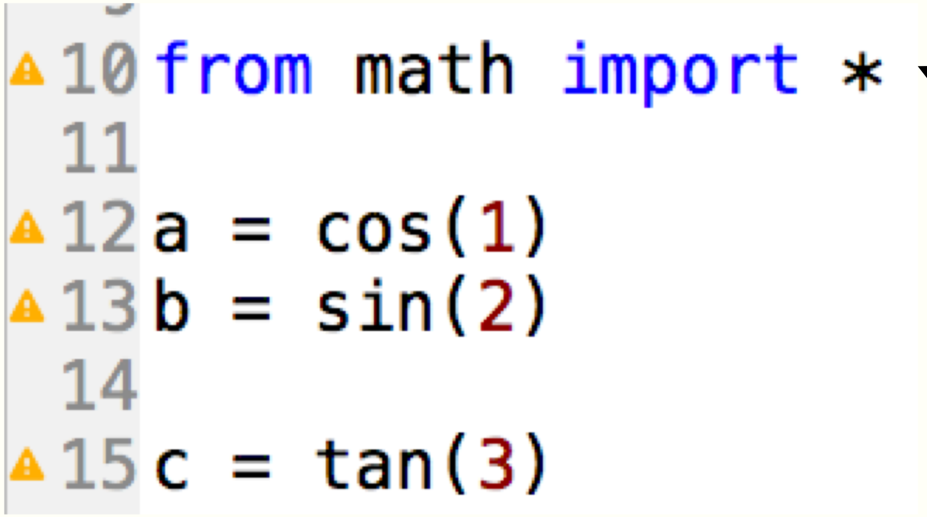
- You can import selected functions from a library
- The following code imports only cos and sin function
- Note that if you use selective import, you can simply use cos instead of math.cos
- You haven’t imported tan so there is an error in Line 15



```
 10 from math import cos, sin
 11
 12 a = cos(1)
 13 b = sin(2)
 14
⚠15 c = tan(3)
```

Bad way to use import

- The following code runs but the editor complains



```python
from math import *

a = cos(1)
b = sin(2)

c = tan(3)
```

This method of importing is BAD. DON'T USE.

- This is because
  - It is no longer possible to keep track of where the functions are coming from
  - Multiple libraries may have functions with the same name. This can lead to a name clash.
- We consider this poor coding practice. DON'T USE.

Summary

- List comprehension
- Lists
  - Indexing and slicing
- Import

End

Week 3b: list comprehension, list indexing and slicing, import