APS106

UNIVERSITY OF TORONTO

Tutorial 2 – Week 3

We'll be starting at the 10 minute mark

if nothing else, write #cleancode

Introduction – TA

Ali Howidi

(TUT0109)

Current studies: Second year MASc BME student

Research/other interests: Signal analysis and machine learning for biosignals

Agenda

1. Lab 1 Review
   a) Review of errors
2. Lecture Review
   a) Defining and using functions
   b) Functions: Tips & Misconceptions
   c) Built-in functions
   d) Importing and Using Modules
3. Practice Questions
4. Questions?

Upon completing this tutorial, the students should be able to:

Learning Objectives

1. Understand the general concept of "function"
2. Recognize Python function calls
3. Recognize Python function definitions
4. Know how to define a function (header + body)
5. Understand the use of Python `return` statement
6. Know how to return a value from a function using a return statement
7. Understand the general concept of "variable scope"
8. Understand the general concept of built-in "function"
9. Know how to import modules
10. Know how to access a function defined in a module they imported

APS106

UNIVERSITY OF TORONTO

Review of Lab 1

Errors

if nothing else, write #cleancode

Lab 1: Review of Errors[1]

Command	Action
`alex.up()`	Lift the tail (stop drawing)
`alex.down()`	Lower the tail (start drawing)
`alex.right(d)`	Turn right by *d* degrees
`alex.left(d)`	Turn left by *d* degrees
`alex.forward(s)`	Move *s* steps in the current direction
`alex.backward(s)`	Move *s* steps backwards with the current heading
`alex.setheading(d)`	Change heading to direction *d* (0: east, 90: north, 180: west, 270: south)
`alex.goto(x, y)`	Move to coordinates *(x, y)*
`alex.circle(r, d)`	Move in circle with r radius for *d* degrees (counterclockwise if *d* > 0)

Syntax Errors: *occur when your code breaks the syntactic rules of the language (e.g., your code contains unbalanced brackets, identifiers, such as variable or function names, that do not obey the rules)*

Added comma missing between arguments

- ***Example:*** **`alex.circle(50 180)`**
  - ***Corrected version:*** **`alex.circle(50, 180)`**
  - ***Explanation:*** *a syntax rule was broken, more specifically the* ***RULE*** *that function arguments must be separated by commas*

Lab 1: Review of Errors[2]

Semantic Errors: occur when the semantic rules of the language are broken (e.g., calling a function with the wrong argument type or with the wrong number of arguments)

Command	Action
`alex.up()`	Lift the tail (stop drawing)
`alex.down()`	Lower the tail (start drawing)
`alex.right(d)`	Turn right by *d* degrees
`alex.left(d)`	Turn left by *d* degrees
`alex.forward(s)`	Move *s* steps in the current direction
`alex.backward(s)`	Move *s* steps backwards with the current heading
`alex.setheading(d)`	Change heading to direction *d* (0: east, 90: north, 180: west, 270: south)
`alex.goto(x, y)`	Move to coordinates *(x, y)*
`alex.circle(r, d)`	Move in circle with r radius for *d* degrees (counterclockwise if *d* > 0)

Removed argument

- **Example:** `alex.up(5)`
  - **Corrected version:** `alex.up()`
  - **Explanation:** a semantic rule was broken, i.e., method `up()` was called with the wrong number of arguments, more specifically `up()` accepts fewer arguments than were passed to it.

Lab 1: Review of Errors[3]

Command	Action
`alex.up()`	Lift the tail (stop drawing)
`alex.down()`	Lower the tail (start drawing)
`alex.right(d)`	Turn right by *d* degrees
`alex.left(d)`	Turn left by *d* degrees
`alex.forward(s)`	Move *s* steps in the current direction
`alex.backward(s)`	Move *s* steps backwards with the current heading
`alex.setheading(d)`	Change heading to direction *d* (0: east, 90: north, 180: west, 270: south)
`alex.goto(x, y)`	Move to coordinates *(x, y)*
`alex.circle(r, d)`	Move in circle with r radius for *d* degrees (counterclockwise if *d* > 0)

Logical Errors: occur when the meaning/semantics of a program is not the intended one (e.g., a mathematical formula is implemented incorrectly, a block of instructions is mis-indented)

- **Example:** `alex.right(2 * math.pi)`
  - **Corrected version:** `alex.right( 360 )`
  - **Explanation:** a higher level semantic rule was broken, i.e., function `right()` assumes that the values passed to it represent angles expressed in **degrees** (not **radians !)** and was passed a value that represents the value of an angle expressed in radians.

Corrected the value of the argument

APS106

UNIVERSITY OF TORONTO

Review of Lecture

Defining and Using functions

if nothing else, write #cleancode

Defining Functions

Function Elements:

1. Header
2. Body
   - Docstring
   - Statements implementing the purpose (intended semantics) of the function

Practice question:

I implement a function that calculates the area of a rectangle.

Week3.py

function_name

```
#function syntax

def function_name (parameters): #function heading
    #docstring
    '''
    Type contract (ie all input parameter types -> return type
    Describe your function here
    '''

    # function body goes here


    # return statement at the end to exit the function
```

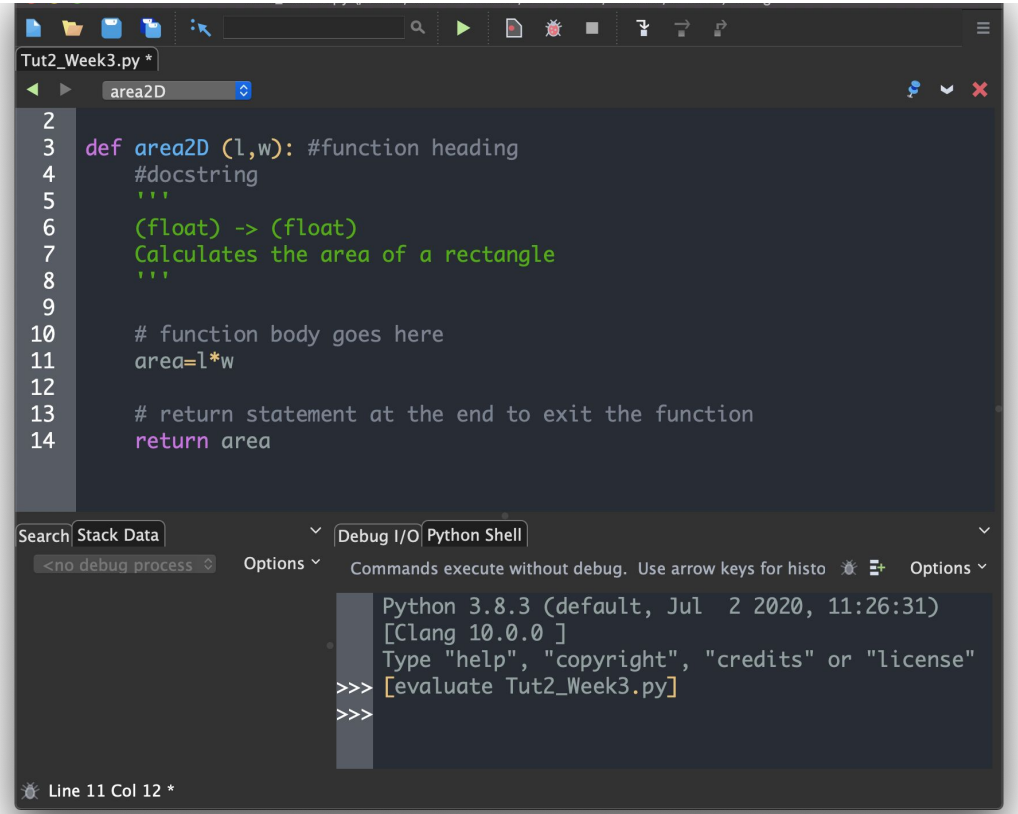
Example of a function

Function Elements:

1. Header
2. Body
   - Docstring
   - Statements implementing the purpose (intended semantics) of the function

Questions:

- What happens when I run this code?
- How do/can I use this function ?



Calling functions

Note: variables `length` and `width` are **parameters** of the `area2D` function.

Questions:

- Q1: Why isn't the area of the rectangle with length 5 and with 7 displayed in the Python shell?
- Q2. How can I display the value returned by the function `area2D` ?

(bottom)

```
#function syntax
def area2D(l,w): #function heading
    #docstring
    '''
    (float), (float) -> (float)
    Calculates the area of a rectangle
    '''

    #function body goes here
    area=l*w

    #return statement at the end to exit the function
    return area

length=5
width=7
area2D(length,width) #calling your funciton
```

Functions and variables

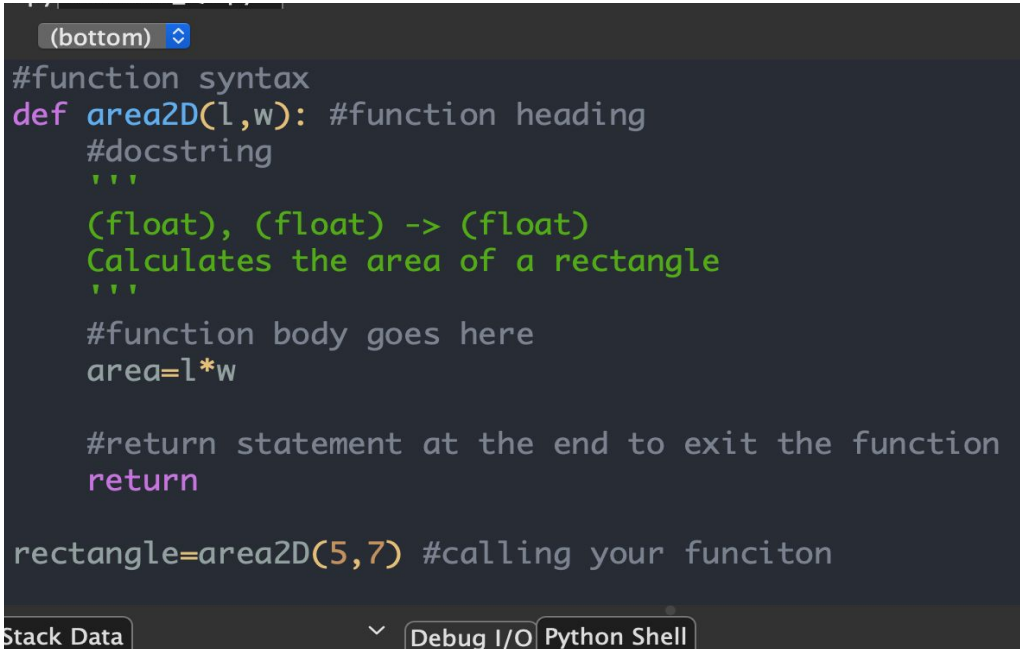
- **Q1 Answer:** the value returned by the function **`area2D`** is "stored" in variable **`rectangle`**, but the program does not contain an instruction for displaying the value of variable **`rectangle`**.

- Try typing the following statements in the python shell to see what happens.

```
>>> print(rectangle)
>>> rectangle
```

- Try adding the following statements to the Python script:

```
print(rectangle)
rectangle
```



APS106

UNIVERSITY OF TORONTO

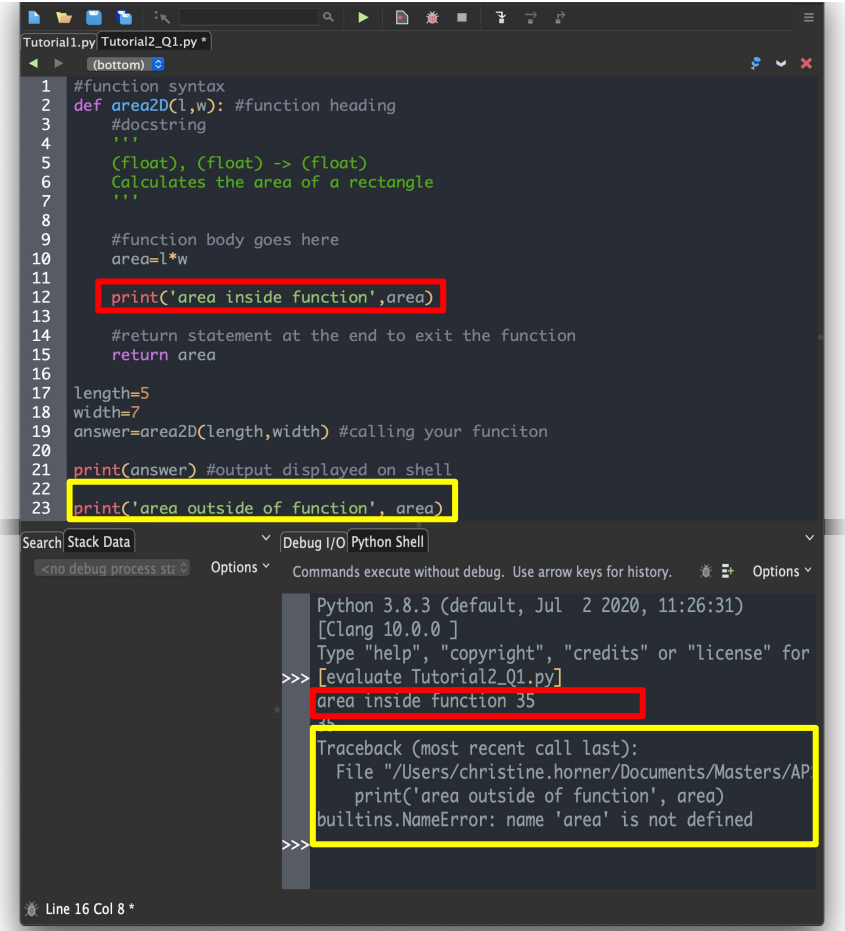
Review of Lecture

Functions: Tips & Misconceptions

Variable Scope

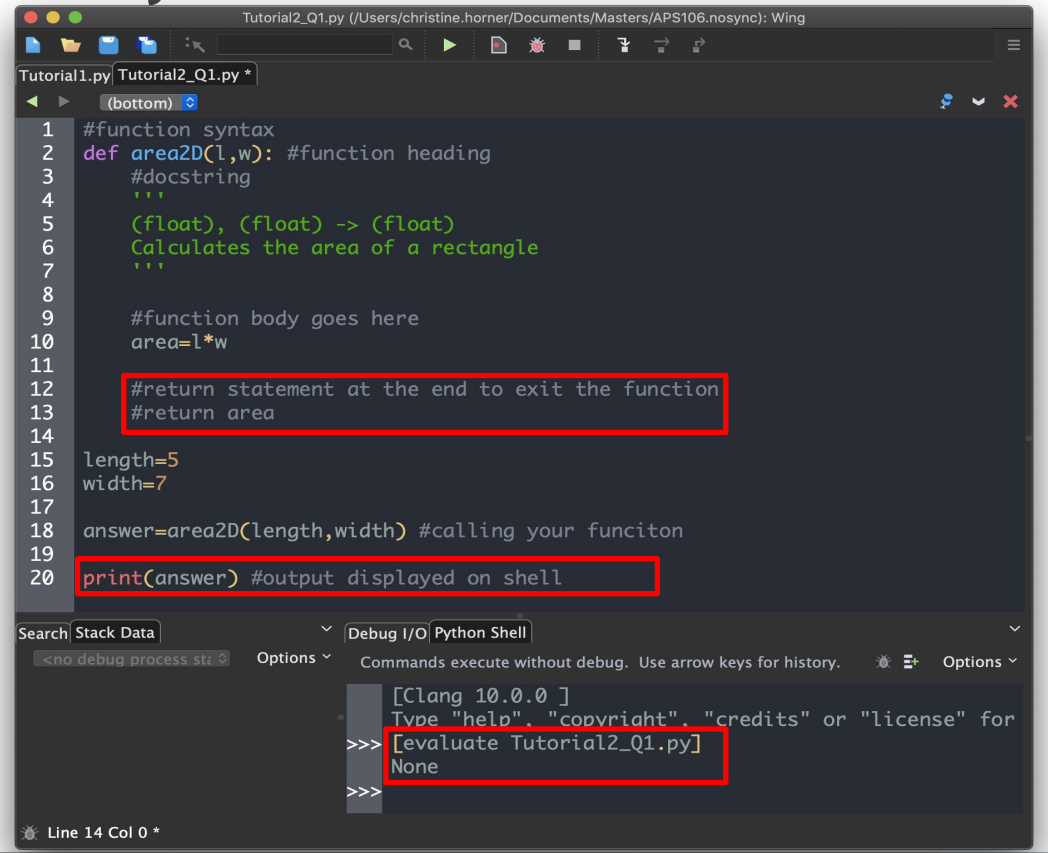
- Variables declared and initialized **inside** a function are only accessible, aka visible, within the body of the function.

- Their values can be accessed from inside the function, e.g., they can be displayed in the shell by passing them as arguments to `print()` (red case), but cannot be accessed from outside the function, e.g., from the main program body (yellow case)



Python Functions Always Return a Value ![1]

A Python function whose body does not contain a `return` statement will return by default the value **`None`**.

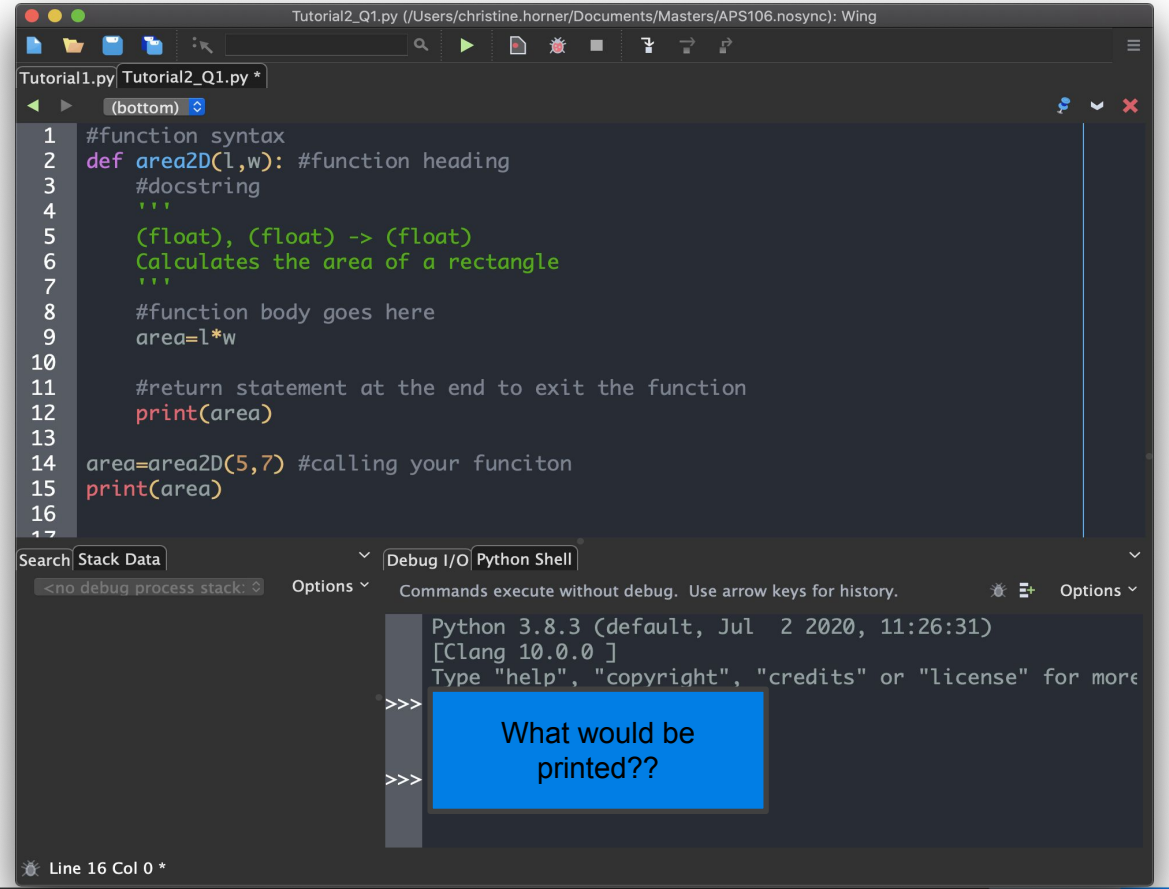


Python Functions Always Return a Value !^{2}

What would be printed in the Python shell ?

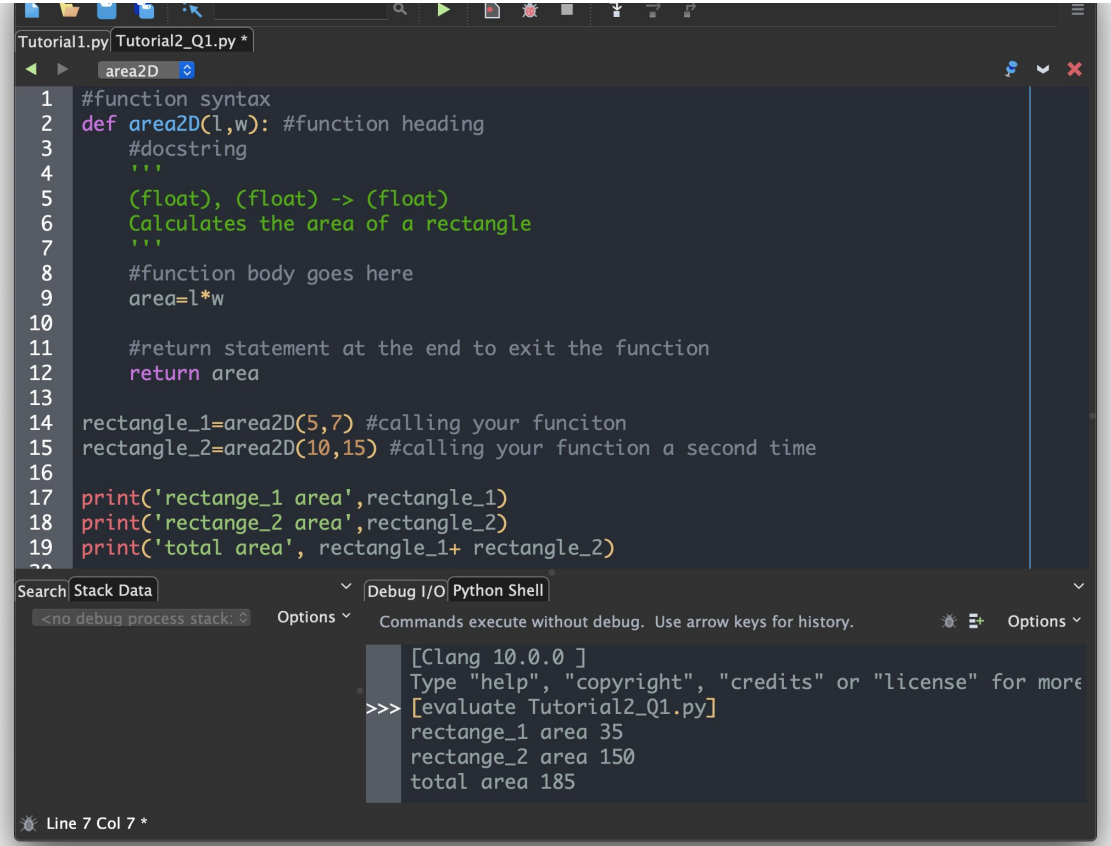
If you get this, you understand how functions work ☺️

If not, ask more questions !



When Do We Need to Use `return` Statements?

What if you wanted to find the total area of two rectangles ?



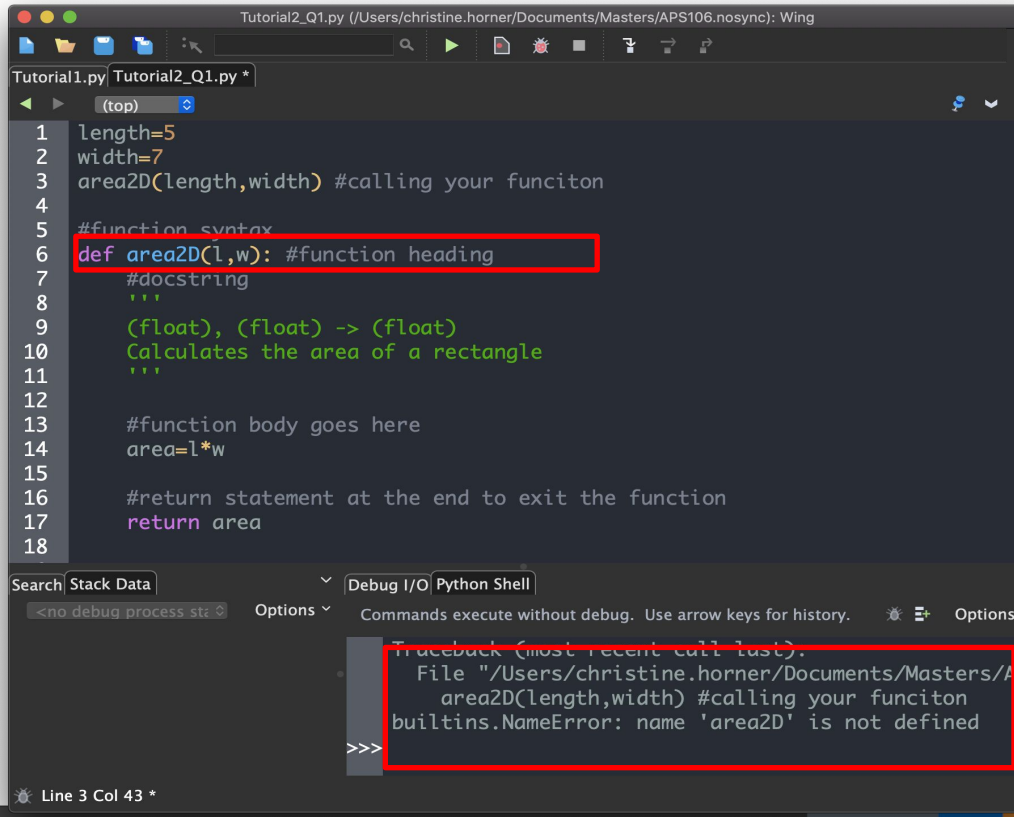
Functions must be defined before being called...

or you will get an error !

The error message is telling us that Python has no idea what `area2D` is.

This is a semantic error caused by the programmer calling a function before it was defined.

⇒ Remember that the Python interpreter runs your code line by line. The statement containing the definition of `area2D` had not been reached at the time the statement in line 3 was executed.



APS106

UNIVERSITY OF TORONTO

Review of Lecture

Built-in Functions

if nothing else, write #cleancode

Lecture Content: Built-in Functions![1]

Built-in Functions: functions that come pre-packaged with a Python implementation

```
abs(-23)
abs(56.24)
type('hello')
```

```
int(4.2)
float(8)
id(4)
```

- Other built in functions can be found here: https://docs.python.org/3.8/library/functions.html
- Use `help()` to look up the documentation for a function, e.g., `help(abs)`

Lecture Content: Functions[2]

`input( [ prompt ])`

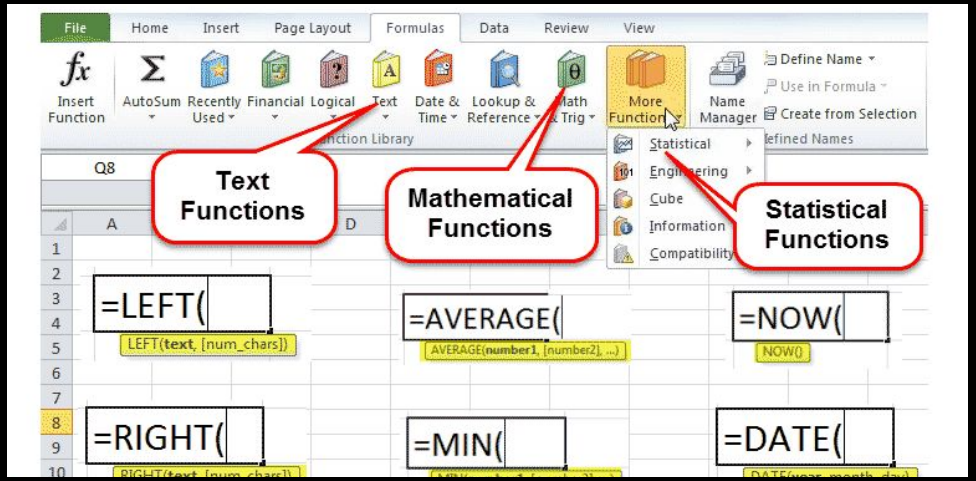
- If the *prompt* argument is present, it is written to standard output. The function then reads the line entered by the user in the command line/shell, converts it to a string (stripping a trailing newline), and returns that.
  - Depending on how we want to process the value entered by the user in our code, we may need to explicitly convert the string returned by `input()` using typecasting

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

- Print *objects* to the text stream *file*, separated by *sep* and followed by *end*.
  - *sep*, *end*, *file*, and *flush*, if present, must be given as keyword arguments.

Examples of functions you might have used...

- Excel!



- Think of your favourite apps on your phone

APS106

UNIVERSITY OF TORONTO

Review of Lecture

Importing and Using Modules

if nothing else, write #cleancode

Python Modules

- Various statements, including function and class definitions (which will be introduced in later lectures) can be stored in separate Python files, called **modules**.

- To get access to the functions defined in a module, you need to import the module.
  - Typical form of importing a module:

```
import module_name
```

  - To access a function within a module:

```
module_name.function_name
```

Python's Built-in modules Modules

- Built-in modules we use in this course: **`math`** and **`csv`**.
- Example:

```
import math

print(math.abs(-4))
```

  - There are many other cool, open source Python libraries, e.g., NumPy, Pandas, scikit-learn, but we won't cover them in this course and you are not allowed to use them in lab assignments or exams!

APS106

UNIVERSITY OF TORONTO

Practice Problems

slido

What is printed when this code is run?

ⓘ Start presenting to display the poll results on this slide.

Review Practice Problem 1

```
# Function practice problem 1
def my_func():
    x = 1
    print (x)
    x = x + 1
    return x

x = 4
print(x)
x = my_func()
print(x)
```

What is printed when this code is run?

4,5,5
4,1,4
4,1,2
4,1,1
None of the above

slido

What is the mistake in this code?

ⓘ Start presenting to display the poll results on this slide.

Review Practice Problem 2

(bottom)

```
#function syntax
def area2D (l,w): #function heading
    #docstring
    '''
    (float) -> (float)
    Calculates the area of a rectangle
    '''

    # function body goes here
    area=l*w

    # return statement at the end to exit the function
    return area


l=input('enter length: ') #input from user
w=input('enter width: ') #input from user

answer=area2D(l,w) #calling your function

print(answer) #output displayed on shell
```

What is the mistake in this code?

Coding Problem 1

Write a function to convert an angle expressed in degrees to radians.

HINT: You can import modules. Are there any built-in functions from the `math` module that can help ?

Things to consider before you start writing your code:

1. What would be the parameters of your function?
2. What should the function output?

Coding Problem 2

Write a function with 3 inputs: the minimum speed limit and maximum speed limit of that street, as well as your current speed

Return a number to the user telling them whether they are obeying speed limit rules (number itself), and if not, which speed limit they are breaking (minimum or maximum)

Hint:
There are functions that take in values and return their minimum and maximum

slido

Any questions?

ⓘ Start presenting to display the poll results on this slide.