CS65: Introduction to Computer Science

Graphics library
Writing more user-defined functions
Quiz 1

Md Alimoor Reza
Assistant Professor of Computer Science

Internship Opportunities for First Year Students @ Wellmark BlueX

3 min video announcement: “Wellmark BlueX: Technology Applied Learning Program” for first year students.

Internship Opportunities for First Year Students @ Wellmark BlueX Program

We are excited to announce a unique opportunity for *first-year students* to partner with Wellmark Inc as a part of our *BlueX* program. Selected students will partake in a 1-credit mentorship course in the Spring of 2023. Once in the program, students will be eligible for future internships and application for employment after graduation.

Requirements: First-year student at Drake University

Application Deadline: October 15, 2022

To learn more, watch the following video: https://drake.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=bdef451f-91a0-479c-a90c-aabd00d8ed0d

To apply, student will need to briefly answer a few questions and submit a resume to **Prof. Timothy Urness**. The application information is available here: https://www.drake.edu/cs/internships/wellmarkbluextechnologyappliedlearningprogram/ We also would like to encourage students to also contact Annette Watson annette.watson@drake.edu (College of Business and Public Administration) or Chrystal Stanley chrystal.stanley@drake.edu (Arts and Sciences) for any assistance with resume preparation.

Recap

- Built-in functions in Python
  - No need to define, just call

- Control flow during function call
  - Debugging features of Thonny
  - Step-by-step execution of your program

- Scope of a variable
  - Global scope vs local scope

Recap: Built-in functions in Python

- If you want to use not so commonly available built-in functions, those built-in functions need to be imported using import keyword from a library
  - library also called a module
- Import the **module** before using it usually at the top of your python file
- Call function using *module_name* . *function_name*

```
import math

value_of_pi = math.pi
```

Recap: Module import variations

Explicitly need to use ***math.pi*** or ***math.sin***

```
# ------------- Module import variation 1 ----------
import math

# variables initialization
angle_in_degree = 45
angle_in_rad = value_of_pi*angle_in_degree/180.0

# calculation
value_of_pi = math.pi
var2 = math.sin(angle_in_rad)

print("sin(", angle_in_degree,") is ", var2)
```

Directly access ***pi*** and ***sin*** *but nothing else*

```
# ------------- Module import variation 3 ----------
from math import pi
from math import sin

# variables initialization
angle_in_degree = 45
value_of_pi     = pi
angle_in_rad    = value_of_pi*angle_in_degree/180.0
var2            = sin(angle_in_rad)

print("sin(", angle_in_degree,") is ", var2)
```

```
# ------------- Module import variation 2 ----------
from math import *

# variables initialization
angle_in_degree = 45
value_of_pi     = pi
angle_in_rad    = value_of_pi*angle_in_degree/180.0
var2            = sin(angle_in_rad)

print("sin(", angle_in_degree,") is ", var2)
```

Directly access ***pi*** or ***sin***

```
# ------------- Module import variation 4 ----------
from math import pi, sin, cos

# variables initialization
angle_in_degree = 45
value_of_pi     = pi
angle_in_rad    = value_of_pi*angle_in_degree/180.0
var2            = sin(angle_in_rad)

print("sin(", angle_in_degree,") is ", var2)
```

Directly access ***pi*** ***sin*** *and* ***cos*** *(in a single import line) but nothing else*

https://docs.python.org/3/tutorial/modules.html

Recap: local and global variables

- Local variables:
  - Variables declared 1) inside function 2) function parameters
  - Only visible to the defined function
- Global variables:
  - Variables that are defined outside of user defined functions
  - Can be accessed by any function after creation
  - Global variable can be replaced/hidden by local variable if declared with the same name

```
# declare the global variables and all the functions below can see these
num1 = 100
num2 = 25

# this user defined function adds two numbers
def add_numbers(num1, num2):
    var = num1 + num2
    return var

# this user defined function subtracts two numbers
def subtract_numbers(var1, var2):
    result2 = var1 - var2
    return result2

def main():
    res1 = add_numbers(num1, num2)
    res2 = subtract_numbers(num1, num2)
    print("add_numbers()      function: called with num1=",num1, ", num2=", num2, " and result is ", res1)
    print("subtract_numbers() function: called with num1=",num1, ", num2=", num2, " and result is ", res2)


main()
```

Scope: local and global variables

- Global variables:
  - Variables that are defined outside of user defined functions
  - Can be accessed by any function after creation
  - Global variable can be replaced/hidden by local variable if declared with the same name

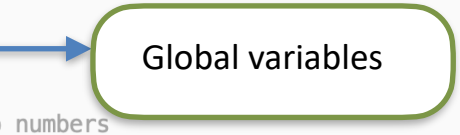
```
num1 = 1


# defining user defined functions
def dummy_function1():
    num1 = 2
    print("Inside function dummy_function1: num1 is local variable ", num1)

print("Before callling dummy_function1() value of num1 = ", num1)

dummy_function1()

print("After callling dummy_function1() value of num1 = ", num1)
```

```
Before callling dummy_function1() value of num1 =  1
Inside function dummy_function1: num1 is local variable  2
After callling dummy_function1() value of num1 =  1
```

Scope: local and global variables

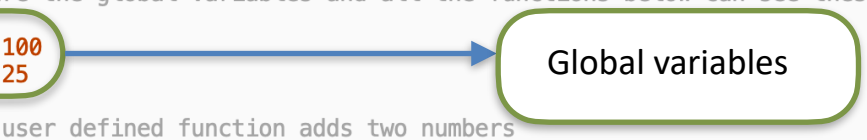
- Global variables:
  - Variables that are defined outside of user defined functions
  - Can be accessed by any function
  - Here values of global variables are copied to the parameters during function call

```
1  # declare the global variables and all the functions below can see these
2
3  num1 = 100
4  num2 = 25
5
6  # this user defined function adds two numbers
7  #             parameters are: num_a and num_b
8  def add_numbers(num_a, num_b):
9      var = num_a + num_b
10     return var
11
12 # this user defined function subtracts two numbers
13 #             parameters are: var1 and var2
14 def subtract_numbers(var1, var2):
15     result2 = var1 - var2
16     return result2
17
18 def main():
19     res1 = add_numbers(num1, num2)
20     res2 = subtract_numbers(num1, num2)
21     print("add_numbers()      function: called with num1=",num1, ", num2=", num2, " and result is ", res1)
22     print("subtract_numbers() function: called with num1=",num1, ", num2=", num2, " and result is ", res2)
23
24
25 main()
```



Demo

```
1  # declare the global variables and all the functions below can see these
2
3  num1 = 100
4  num2 = 25
5
6  # this user defined function adds two numbers
7  #            parameters are: num_a and num_b
8  def add_numbers(num_a, num_b):
9      var = num_a + num_b
10     return var
11
12 # this user defined function subtracts two numbers
13 #            parameters are: var1 and var2
14 def subtract_numbers(var1, var2):
15     result2 = var1 - var2
16     return result2
17
18 def main():
19     res1 = add_numbers(num1, num2)
20     res2 = subtract_numbers(num1, num2)
21     print("add_numbers()      function: called with num1=",num1, ", num2=", num2, " and result is ", res1)
22     print("subtract_numbers() function: called with num1=",num1, ", num2=", num2, " and result is ", res2)
23
24
25 main()
```



Scope: local and global variables

- Scope resolution: Mechanism of searching for a name, e.g., variable or function

  - **Step 1:** search the referenced name in the local scope. If not found, then go to step 2
  - **Step 2:** search the referenced name in the global scope. If not found, then go to step 3
  - **Step 3:** If searched name is not found in either step 1 or step 2, then search in the built-in scope
  - **Step 4:** If not found in the above steps, then interpreter generates an Error message

Topics for today

- Graphics library
  - installation in Thonny
  - drawing shapes using graphics library

- Quiz 1

Graphics library

- A simple library (containing other python codes) that makes it easy to experiment with graphics components
- You will learn how to draw stuffs (shapes, text, etc) on a window using Python programming
- Graphics library: https://mcsp.wartburg.edu/zelle/python/graphics/graphics/index.html

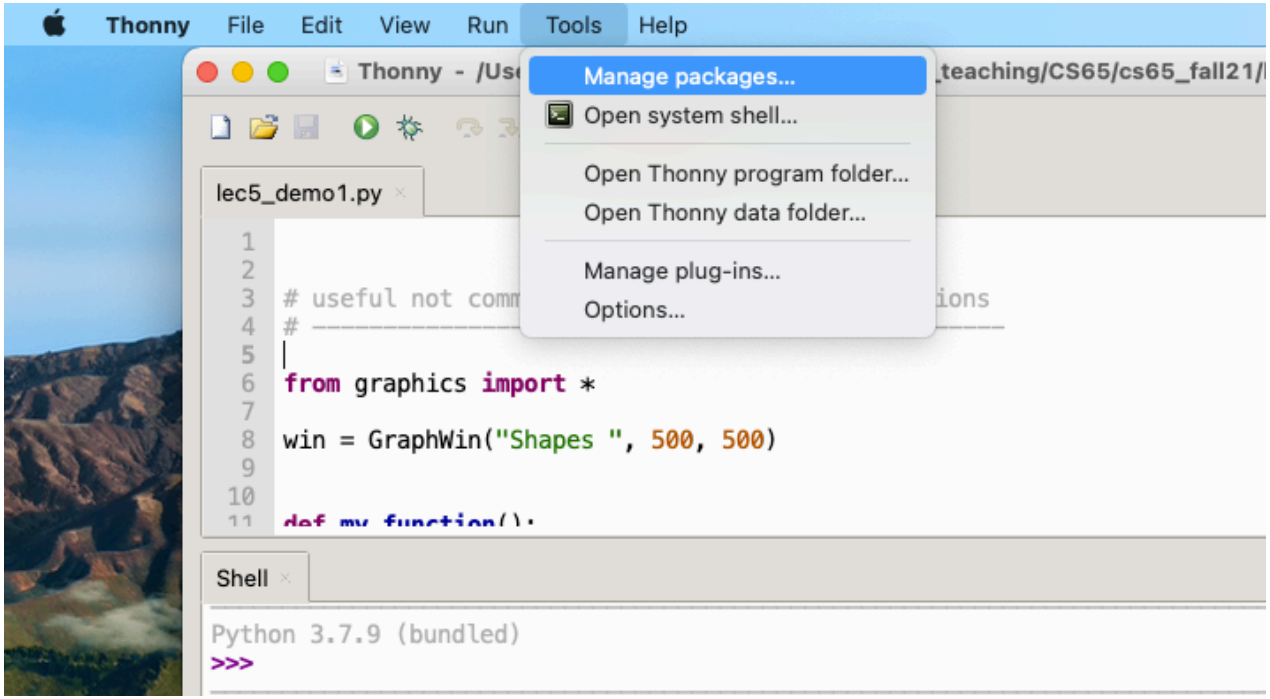
Graphics library

- The graphics library might not be installed in your Thonny
  - ERROR!

```
  6 from graphics import *
  7
>>> %Run lec5_demo1.py
  Traceback (most recent call last):
    File "/Users/reza/Class_and_Research/drake_teaching/CS65/cs
  6, in <module>
      from graphics import *
  ModuleNotFoundError: No module named 'graphics'

>>>
```

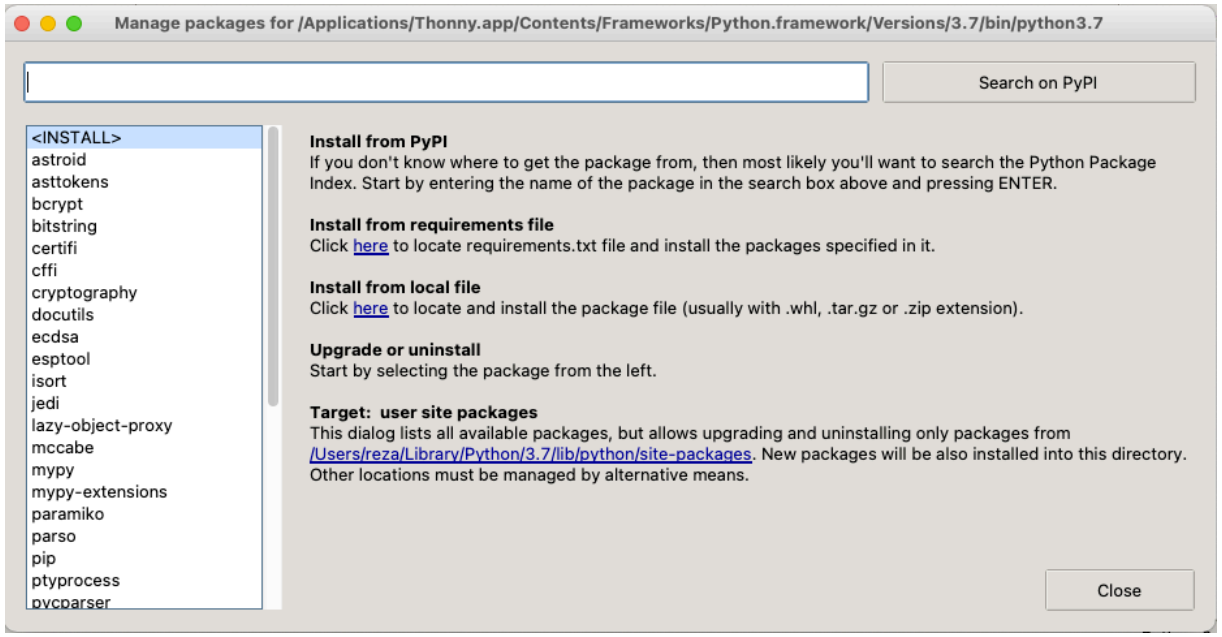
Quick installation of graphics in Thonny

- Find the **Tools** option from the list of menus



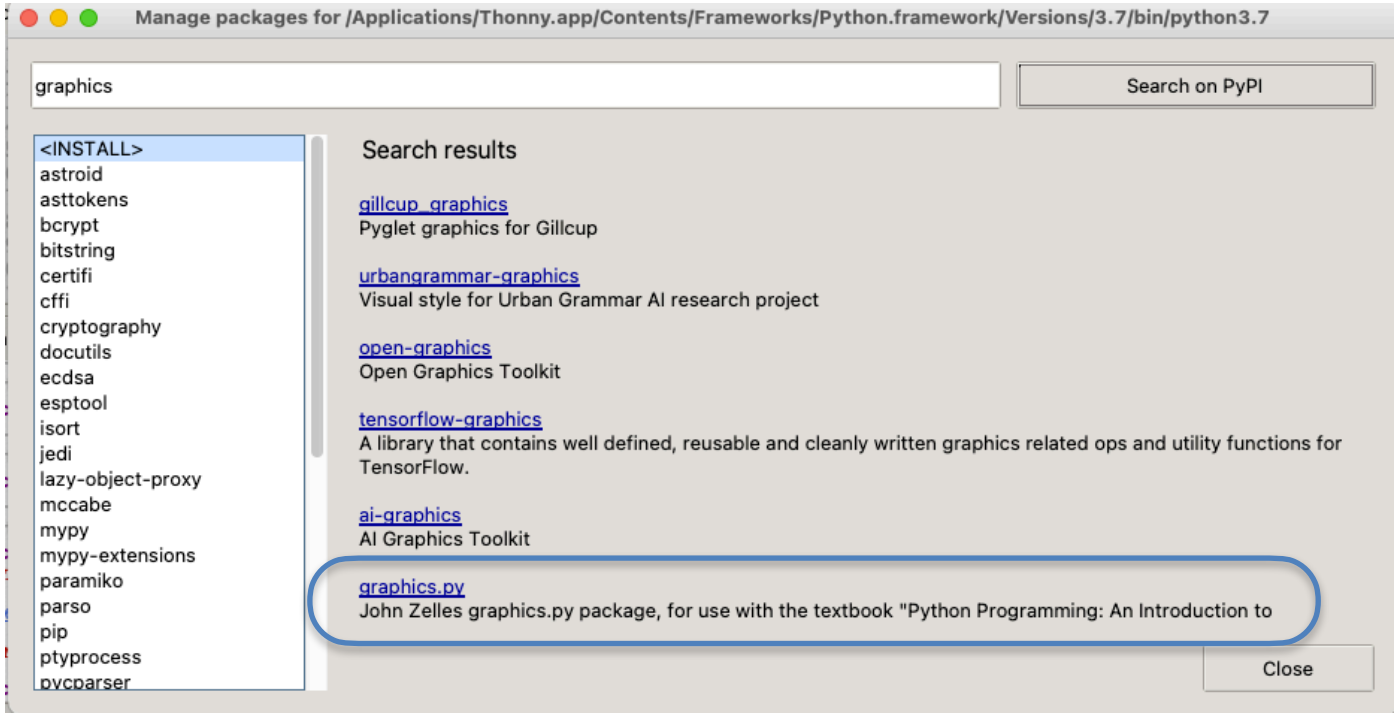
Quick installation of graphics in Thonny

- Type in '<u>graphics</u>' in the empty textbox and then **hit** '<u>Search on PyPI</u>'



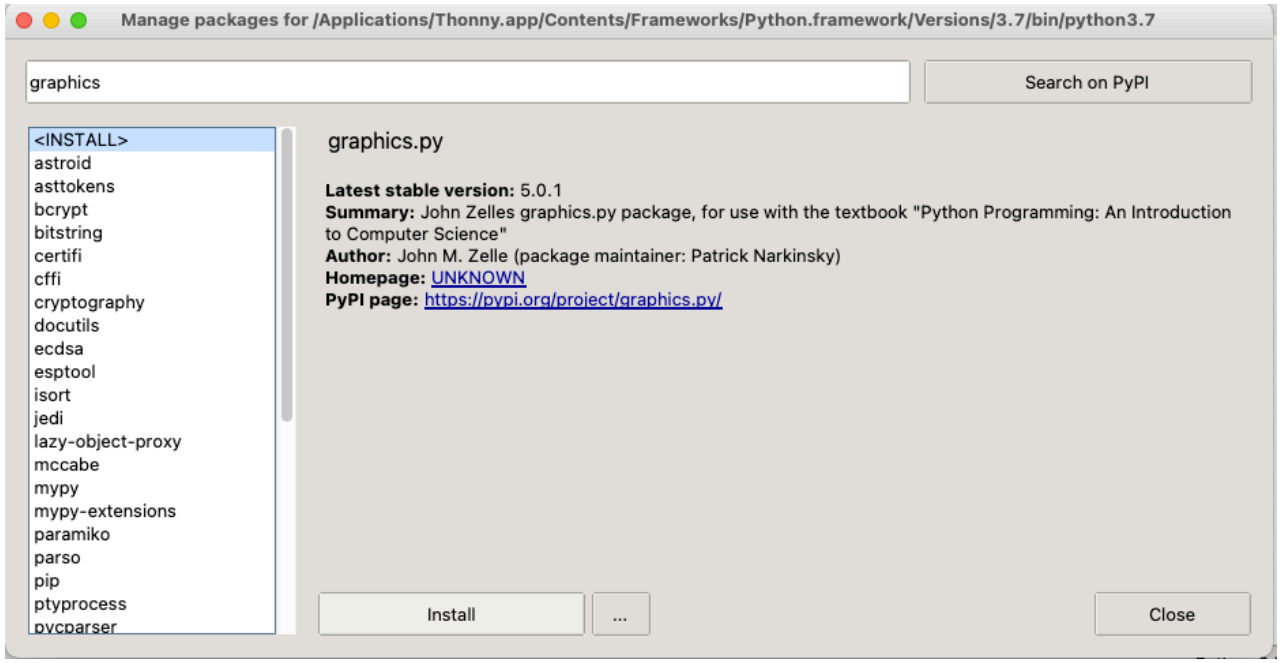
Quick installation of graphics in Thonny

- Select the graphics.py from the bottom



Quick installation of graphics in Thonny

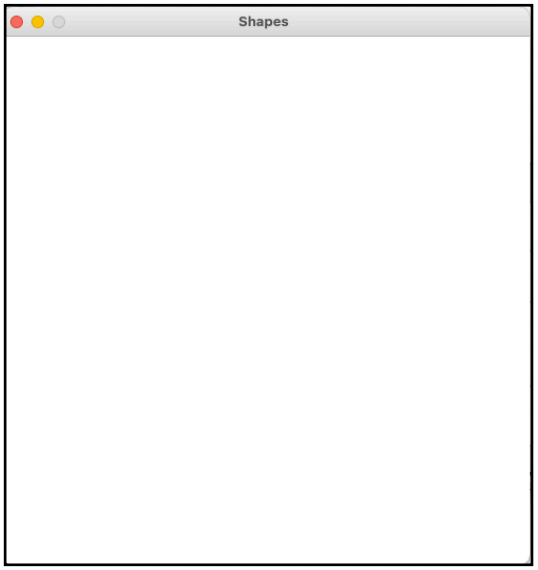
- Finish the installation! Now you are ready to access graphics library components



A simple program using graphics

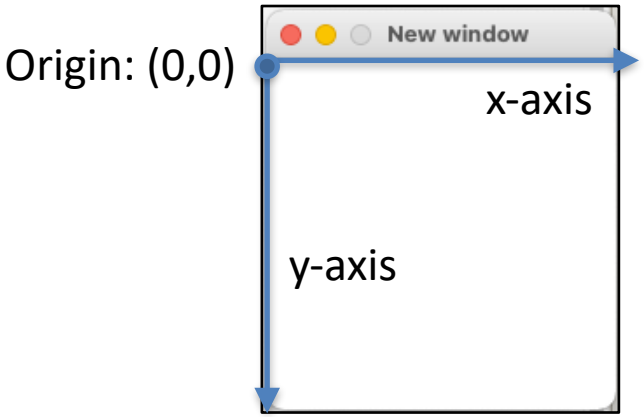
- *GraphWin(…)*: creates the **canvas** or **panel** where everything will be drawn

```
6  from graphics import *
7
8  win = GraphWin("Shapes ", 500, 500)
```



Changing window size

- *GraphWin(…)*: creates the **canvas** or **panel** where everything will be drawn



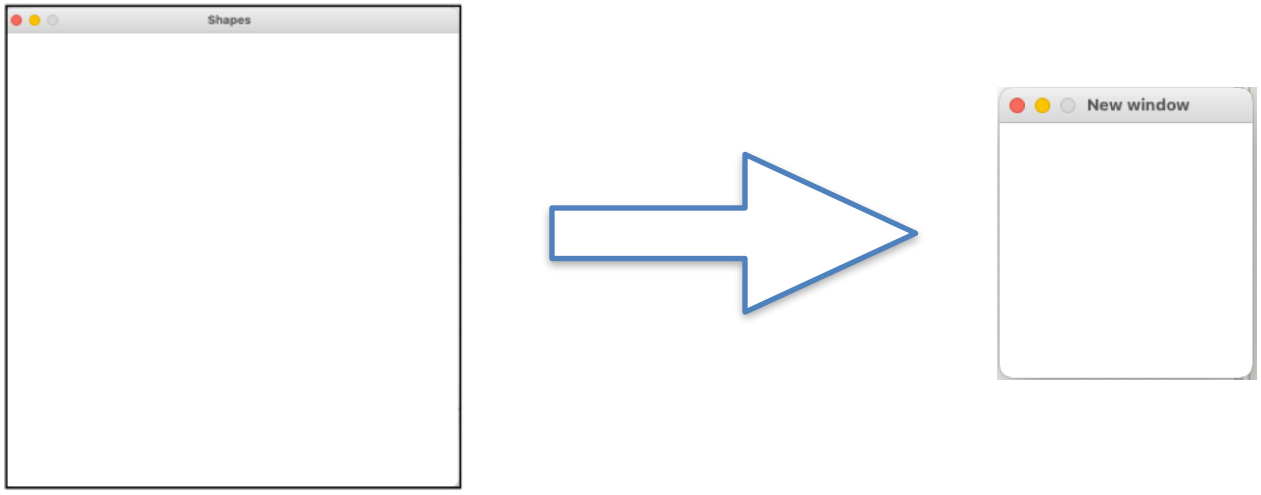
- Coordinate system
  - x: top-left —> top-right
  - y: top-left —> bottom-left

- You can set the dimensions of the window by mentioning the width and height (in pixel units)
  - x-axis — — —> width
  - y-axis — — —> height

Changing window size

- Changing the shape of the window of size (500, 500), just need to change the values inside *GraphWin()*

```
6  from graphics import *
7
8  win = GraphWin("Shapes ", 500, 500)
```

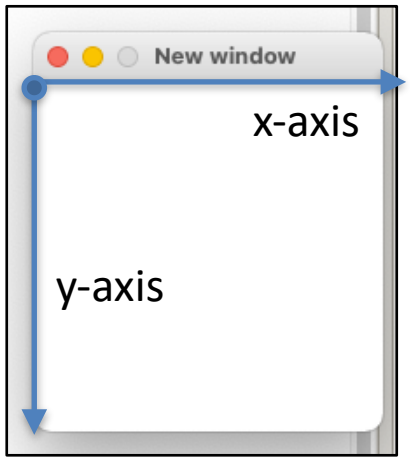


Changing window size

- Changing the shape of the window of size (500, 500), just need to change the values inside *GraphWin()*

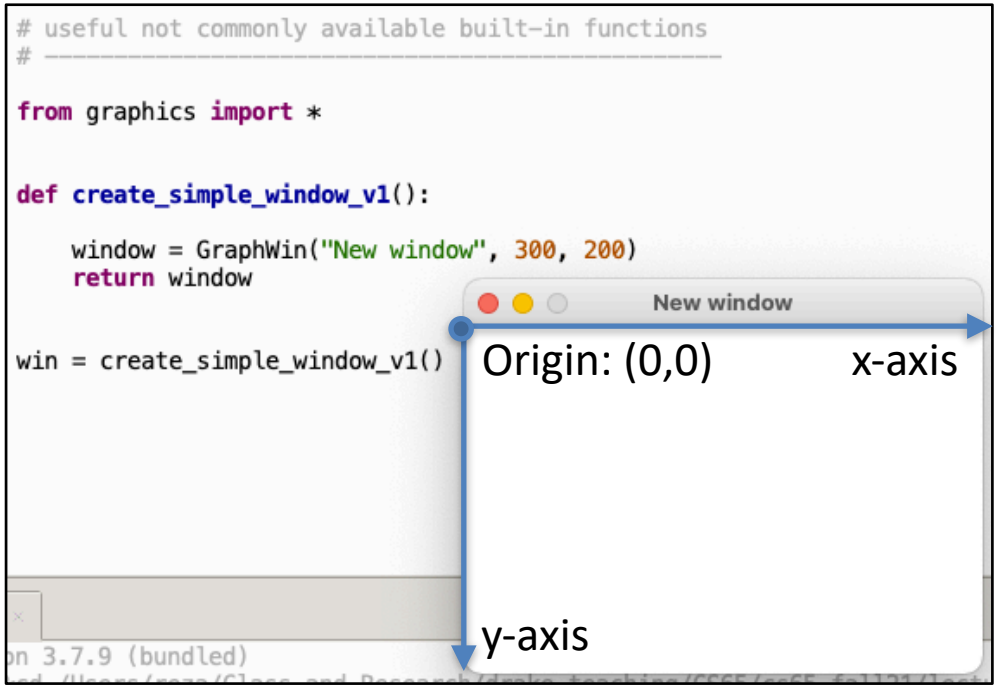
- Write a function for a simple window
  - keep writing your code inside such functions
  - make a habit

Origin: (0,0)



Drawing rectangular window

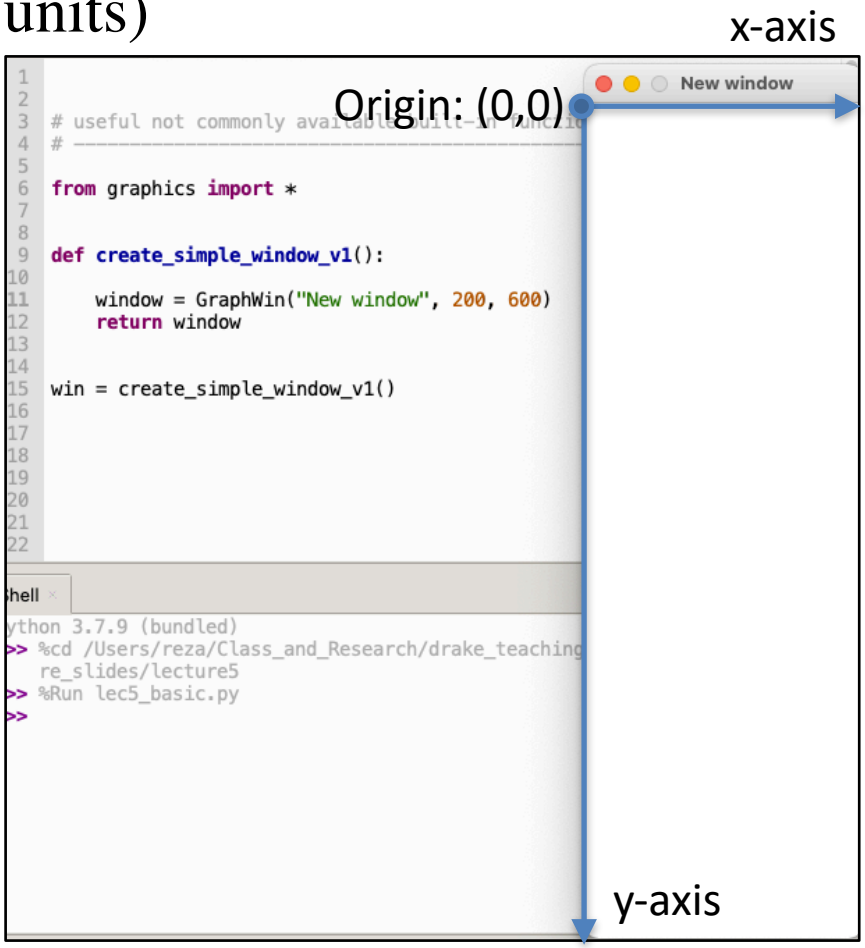
- You can set the dimensions of the window by mentioning the width and height (in pixel units)
  - x-axis — — —> width
  - y-axis — — —> height



Drawing rectangular window

- You can set the dimensions of the window by mentioning the width and height (in pixel units)
  - x-axis ——> width
  - y-axis ——> height

- Coordinate system
  - x: top-left —> top-right
  - y: top-left —> bottom-left



Coding demo

Topics

- Application Programming Interface (API)
  - Graphics API
    - installation in Thonny
    - drawing shapes using graphics components

- Quiz 1

Summary

- **Takeaway from this lecture**
  - Graphics library allows us to draw stuffs
  - Basics shapes are already defined, you just need to draw them according to a specific way
    - Drawing Circle, Rectangle, Triangle (next week)

- **To do:**
  - Read: https://mcsp.wartburg.edu/zelle/python/graphics/graphics/index.html

- **Announcements:**
  - Assignment 1 will be out soon! It will be due in 2 weeks.
  - Quiz 1 grades will be out by next Tuesday (09/20)