CS65: Introduction to Computer Science

Graphics library
Writing more user-defined functions
Quiz 1

Drake UNIVERSITY

Md Alimoor Reza
Assistant Professor of Computer Science

Recap

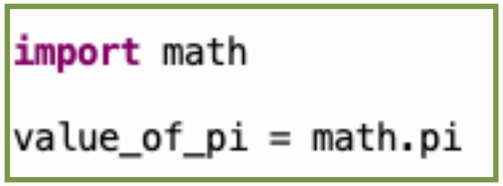
- Built-in functions in Python
  - No need to define, just call
- Control flow during function call
  - Debugging features of Thonny
  - Step-by-step execution of your program
- Scope of a variable
  - Global scope vs local scope

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 <u>module</u>

- Import the **module** before using it usually at the top of your python file

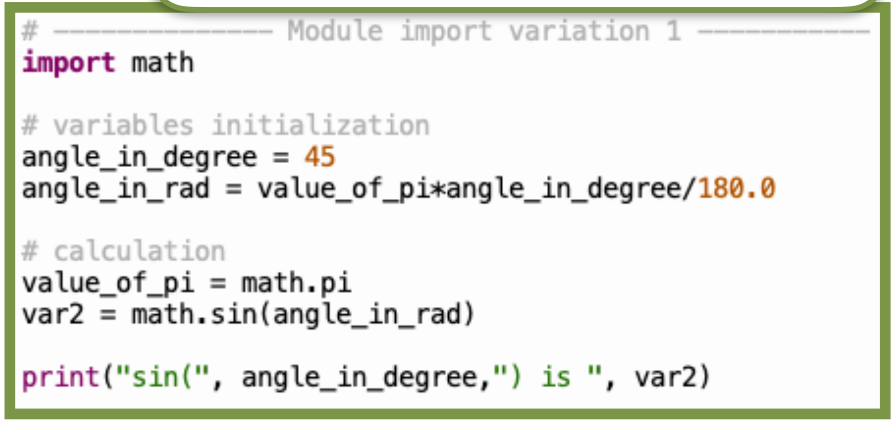
- Call function using *module_name* . *function_name*



```python
import math

value_of_pi = math.pi
```

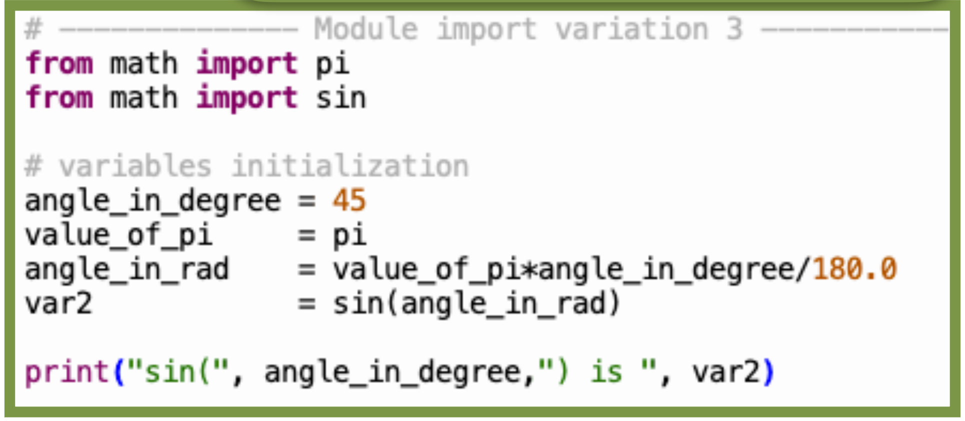
Recap: Module import variations

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

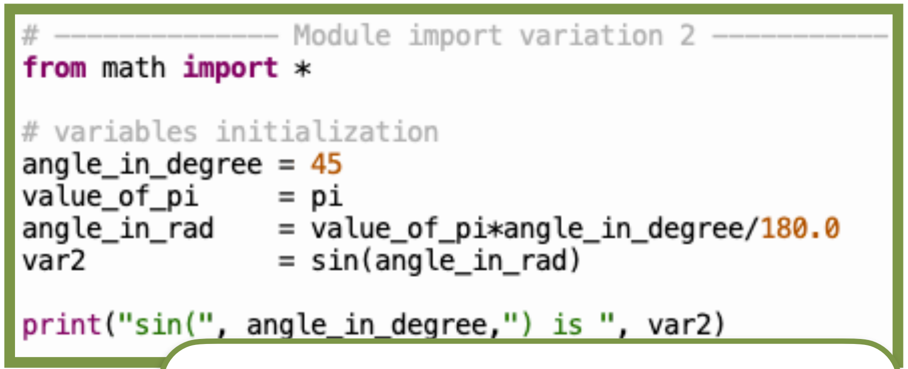


```
# -------------- Module import variation 1 -----------
import math

# 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*

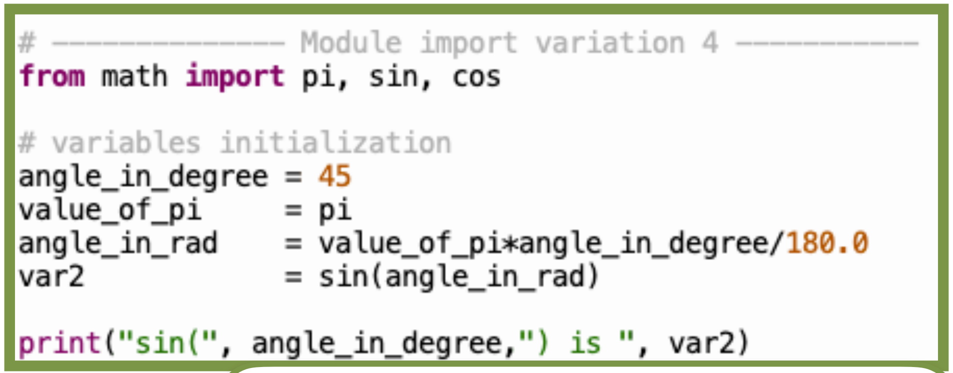


```
# -------------- Module import variation 3 -----------
from math import pi
from math import sin

# 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)
```

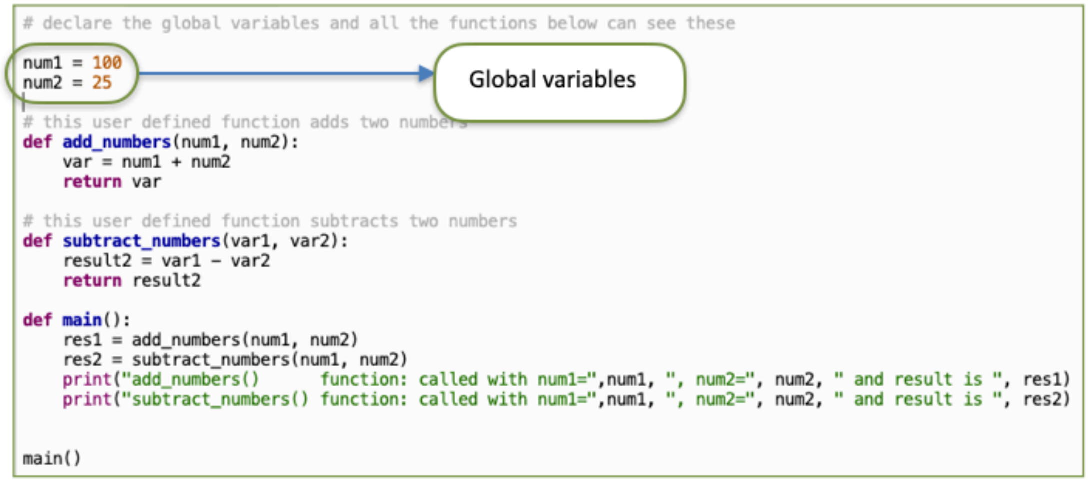
Directly access ***pi sin** and **cos*** *(in a single import line) but nothing else*

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

Drake UNIVERSITY

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()
```

Topics for today

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

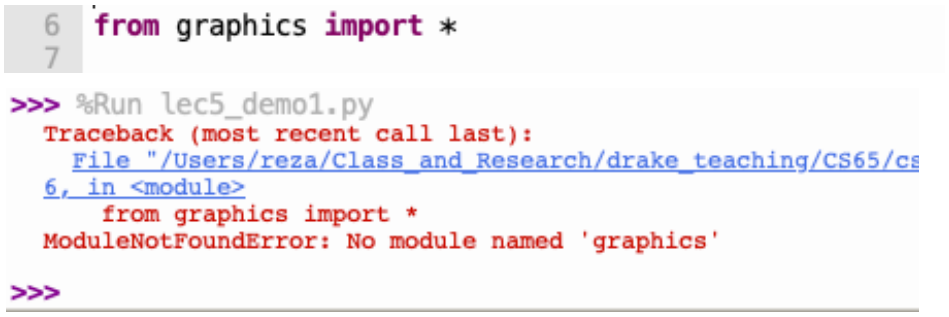
- Quiz 1

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!

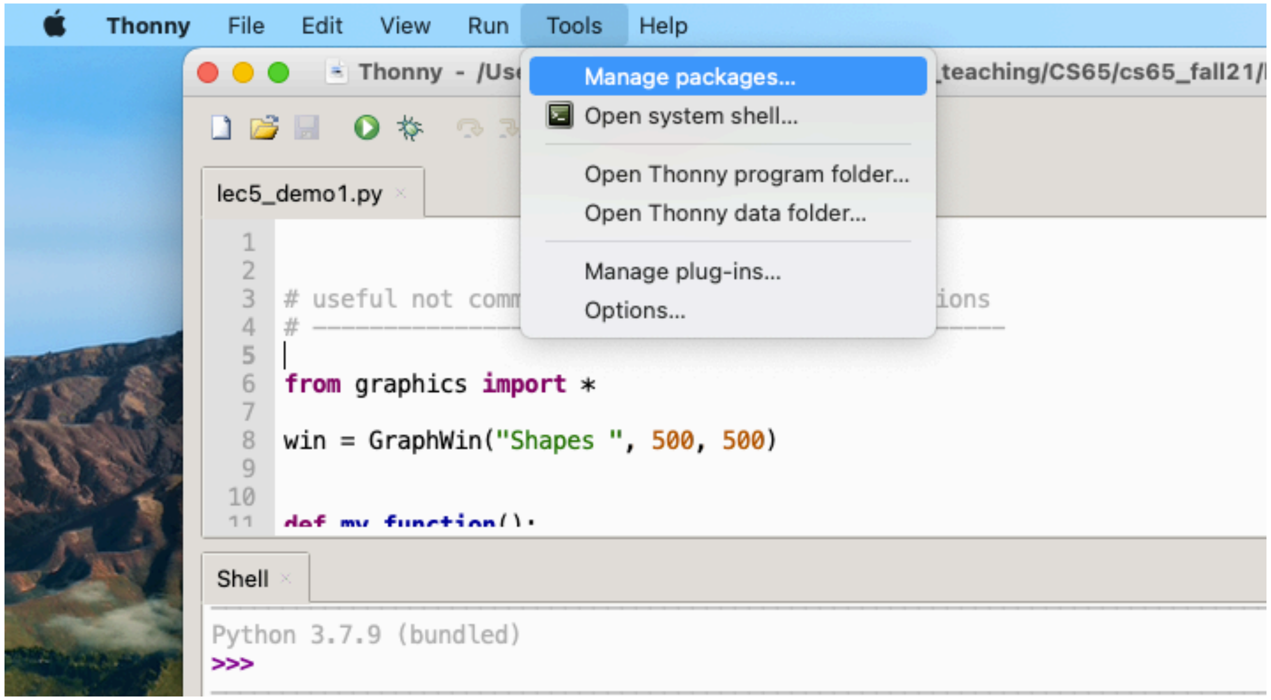


```
from graphics import *
```

```
>>> %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

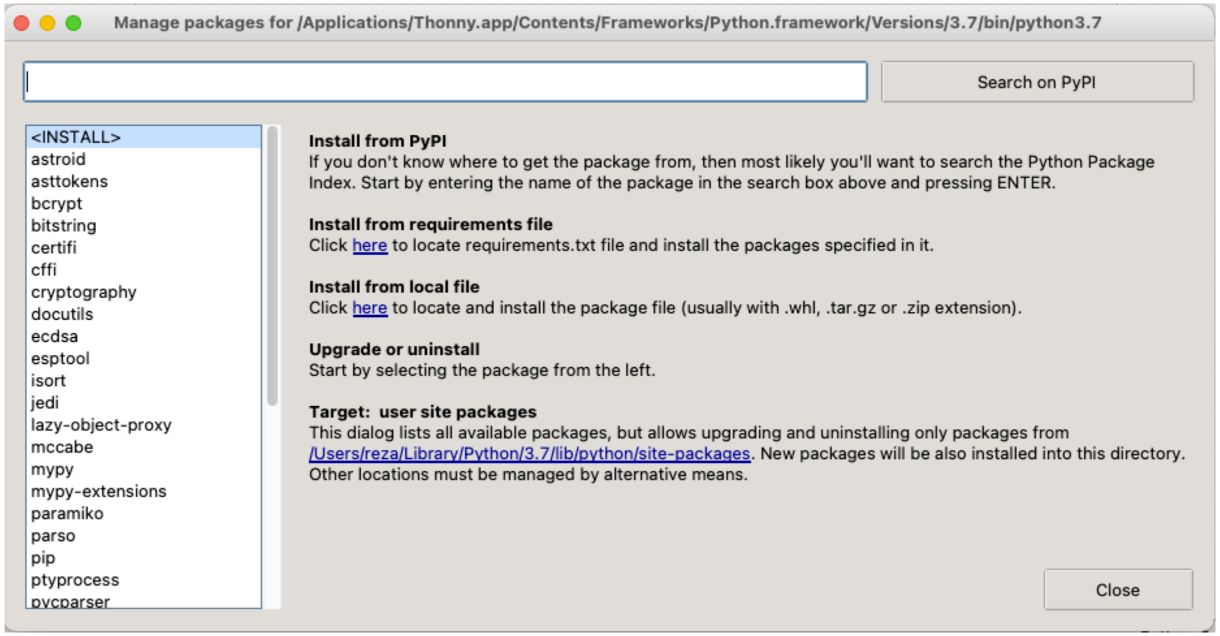


```
# useful not comm                                ions
# ----------------                               -----

from graphics import *

win = GraphWin("Shapes ", 500, 500)


def my_function():
```

```
Python 3.7.9 (bundled)
>>>
```

Quick installation of graphics in Thonny

- Type in ‘graphics’ in the empty textbox and then **hit** ‘Search on PyPI’



Install from PyPI
If you don't know where to get the package from, then most likely you'll want to search the Python Package Index. Start by entering the name of the package in the search box above and pressing ENTER.

Install from requirements file
Click here to locate requirements.txt file and install the packages specified in it.

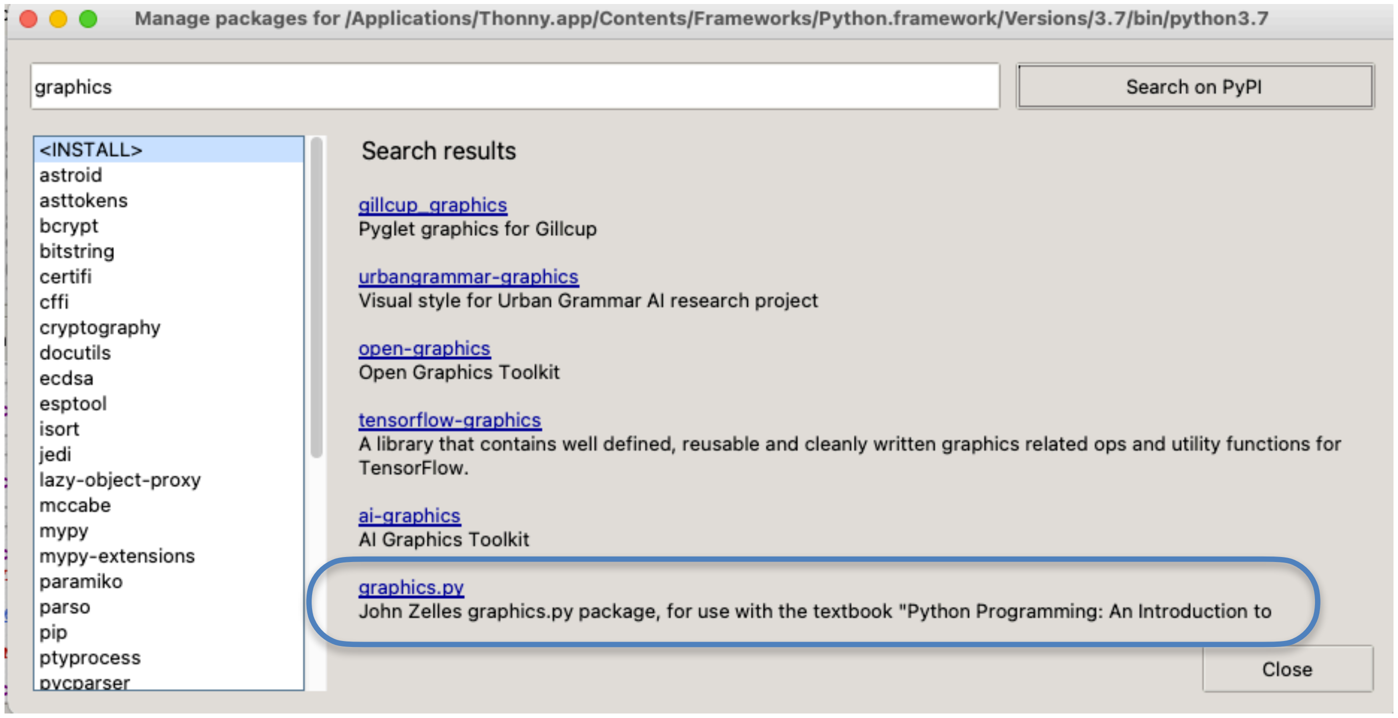
Install from local file
Click here to locate and install the package file (usually with .whl, .tar.gz or .zip extension).

Upgrade or uninstall
Start by selecting the package from the left.

Target: user site packages
This dialog lists all available packages, but allows upgrading and uninstalling only packages from /Users/reza/Library/Python/3.7/lib/python/site-packages. New packages will be also installed into this directory. Other locations must be managed by alternative means.

Quick installation of graphics in Thonny

- Select the graphics.py from the bottom



Manage packages for /Applications/Thonny.app/Contents/Frameworks/Python.framework/Versions/3.7/bin/python3.7

graphics

Search results

gillcup_graphics
Pyglet graphics for Gillcup

urbangrammar-graphics
Visual style for Urban Grammar AI research project

open-graphics
Open Graphics Toolkit

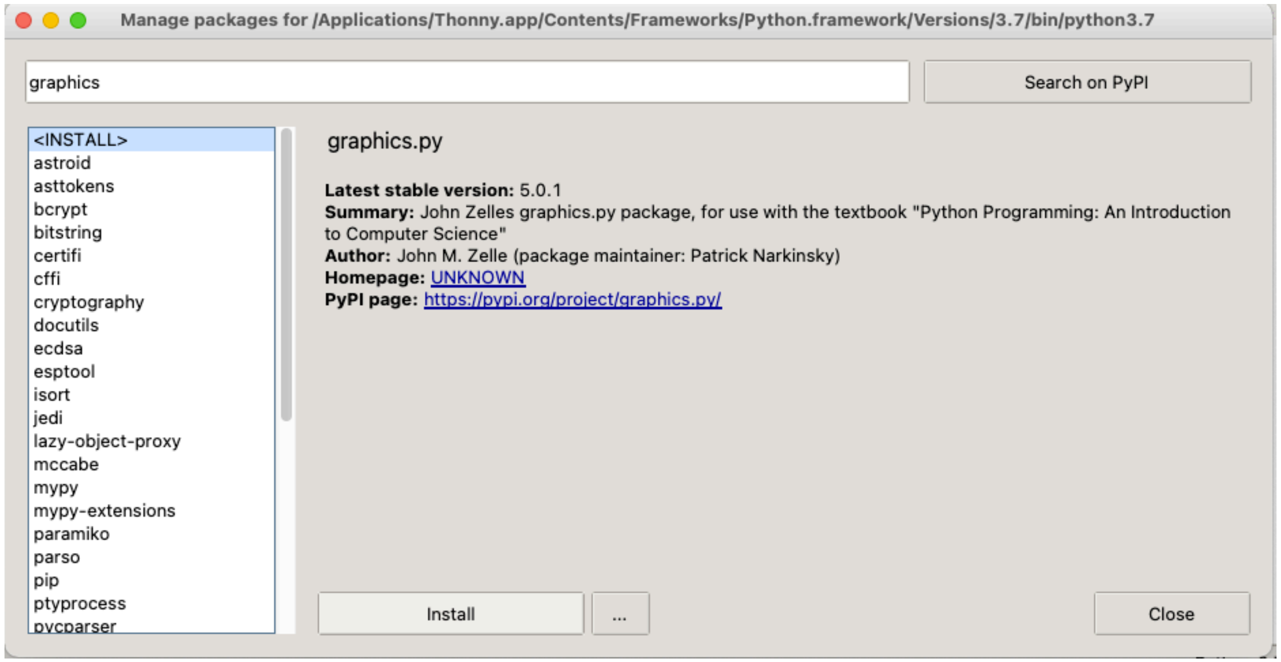
tensorflow-graphics
A library that contains well defined, reusable and cleanly written graphics related ops and utility functions for TensorFlow.

ai-graphics
AI Graphics Toolkit

graphics.py
John Zelles graphics.py package, for use with the textbook "Python Programming: An Introduction to

Quick installation of graphics in Thonny

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


Manage packages for /Applications/Thonny.app/Contents/Frameworks/Python.framework/Versions/3.7/bin/python3.7

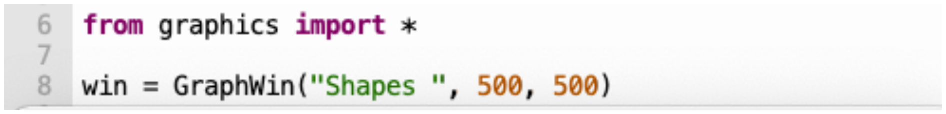
graphics

graphics.py

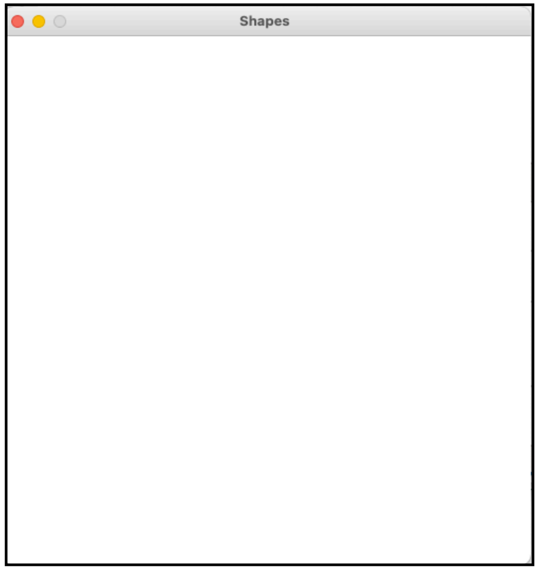
Latest stable version: 5.0.1
Summary: John Zelles graphics.py package, for use with the textbook "Python Programming: An Introduction to Computer Science"
Author: John M. Zelle (package maintainer: Patrick Narkinsky)
Homepage: UNKNOWN
PyPI page: https://pypi.org/project/graphics.py/

A simple program using graphics

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

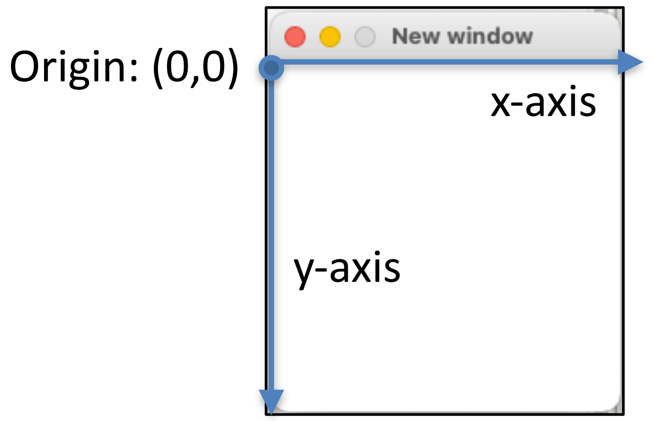


```
from graphics import *

win = GraphWin("Shapes ", 500, 500)
```



Changing window size

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

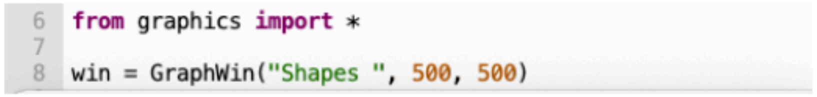


- Coordinate system
  - x: top-left —> top-right
  - y: top-left —> bottom-left

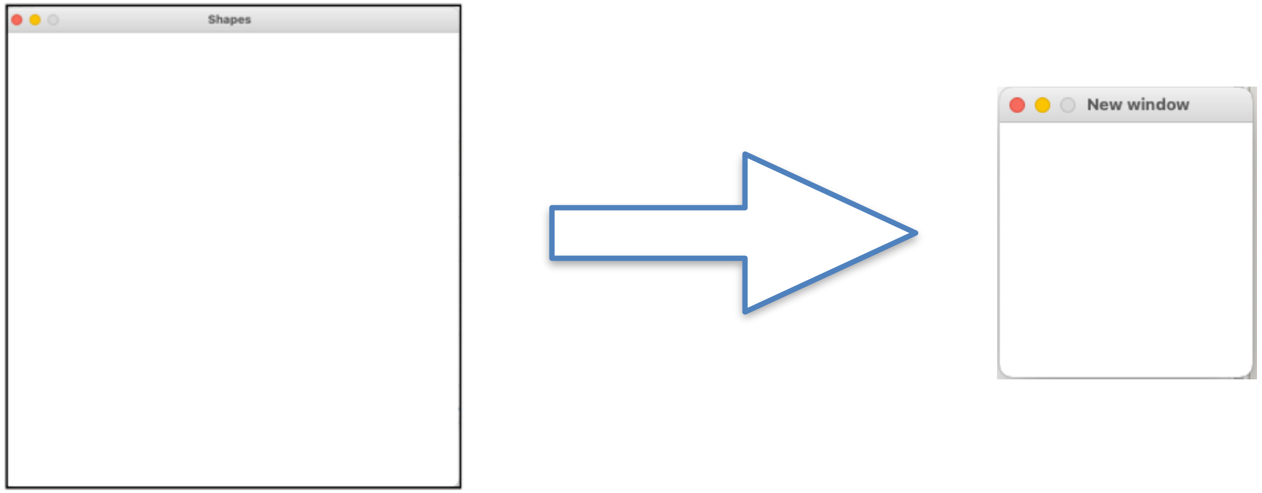
- You can set the dimensions of the window by mentioning the <u>width</u> and <u>height</u> (in pixel units)
  - x-axis ———> width
  - y-axis ———> height

Changing window size

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



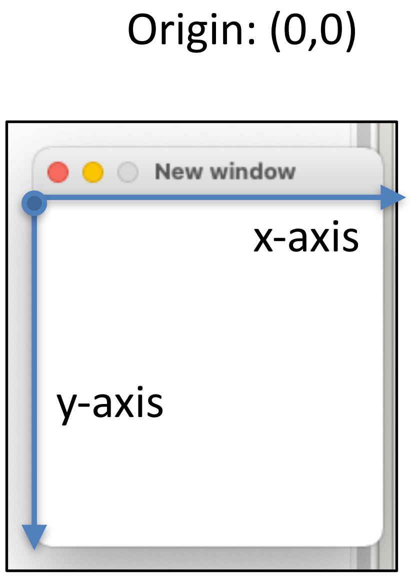
```python
from graphics import *

win = GraphWin("Shapes ", 500, 500)
```



Changing window size

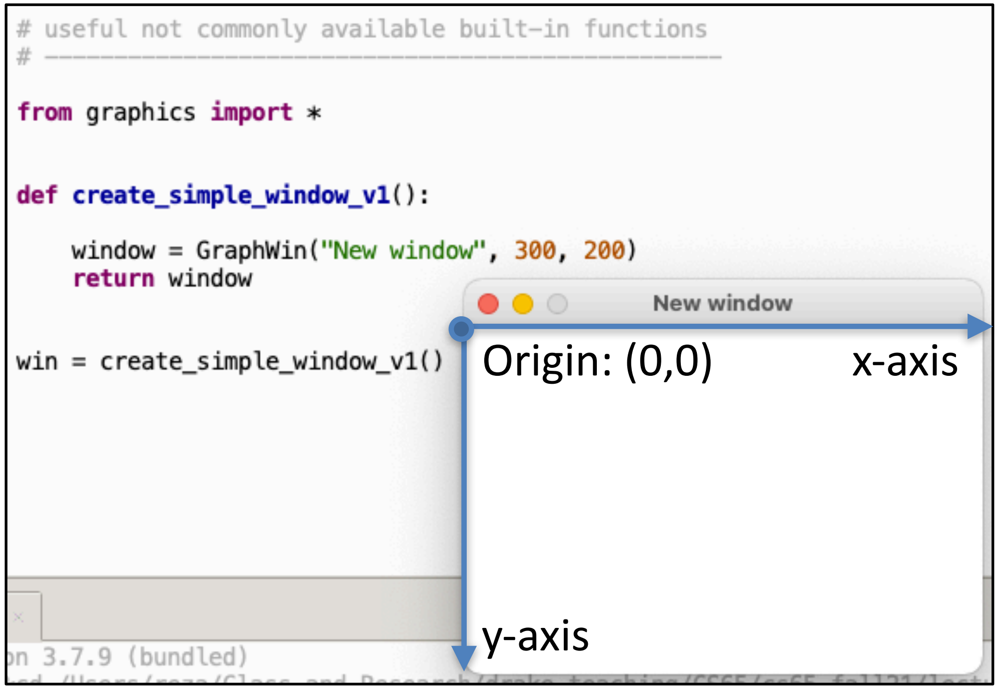
- Changing the shape of the window of size (500, 500), just need to change the values inside *GraphWin()*

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



Drawing rectangular window

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



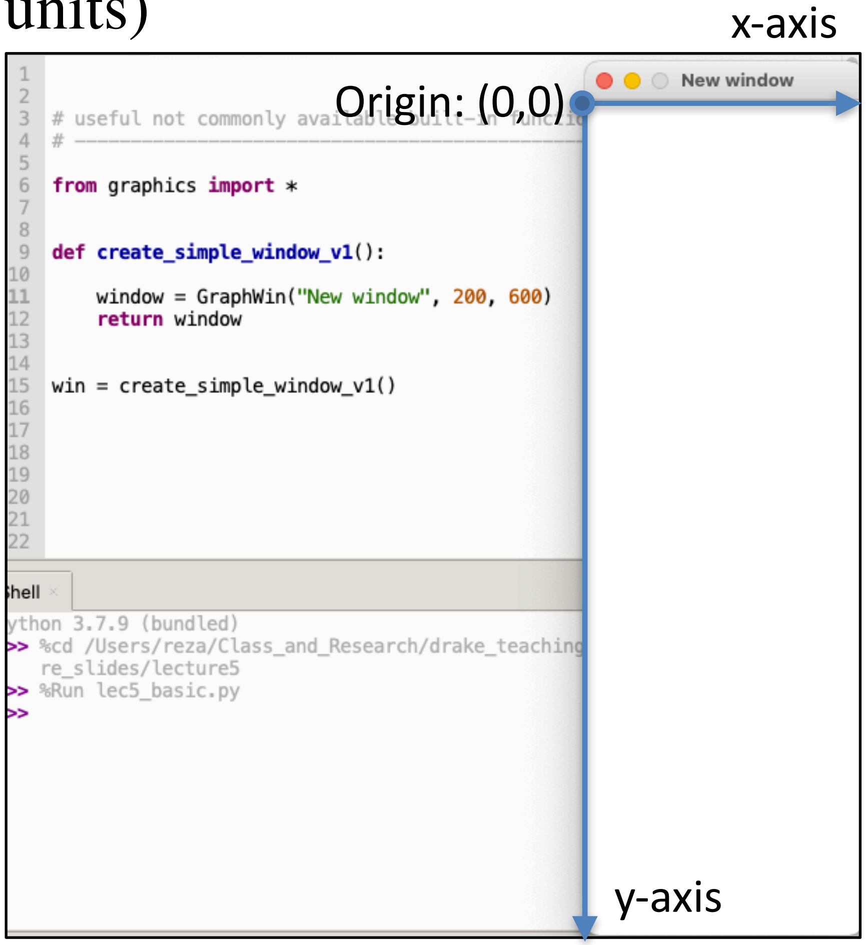
```
# useful not commonly available built-in functions
# ------------------------------------------------

from graphics import *


def create_simple_window_v1():

    window = GraphWin("New window", 300, 200)
    return window


win = create_simple_window_v1()
```

```
on 3.7.9 (bundled)
```

Drawing rectangular window

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

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



```
# useful not commonly available built-in functio
# ------------------------------------------------

from graphics import *

def create_simple_window_v1():
    window = GraphWin("New window", 200, 600)
    return window

win = create_simple_window_v1()
```

```
ython 3.7.9 (bundled)
>> %cd /Users/reza/Class_and_Research/drake_teaching
   re_slides/lecture5
>> %Run lec5_basic.py
>>
```

Coding demo

Graphical objects from graphics library

- Graphics library provides different shapes (graphical objects):
    - **Point**, Line, **Circle**
    - Oval, **Rectangle**, Polygon
    - Text, Image

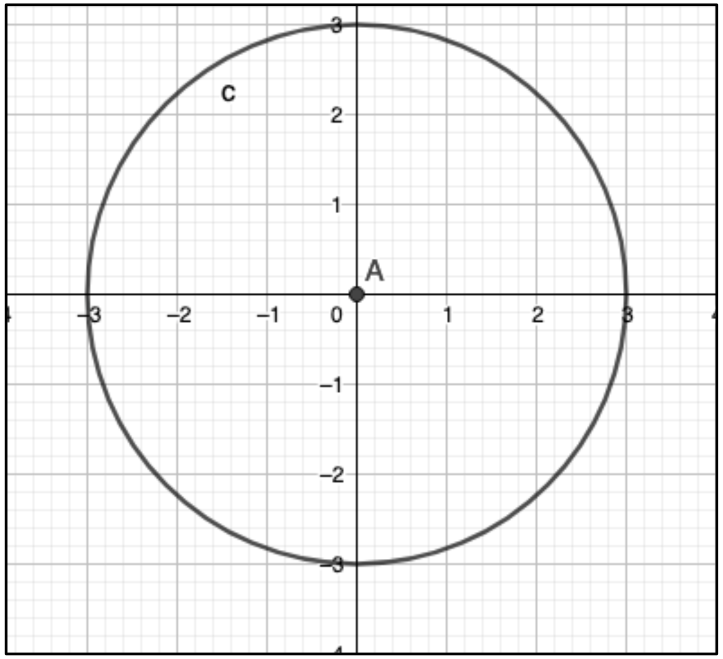
- You can manipulate properties of these shapes/objects
    - change color and sizes

- You can also move them around inside the window

Drawing inside the window

- You can draw inside the window

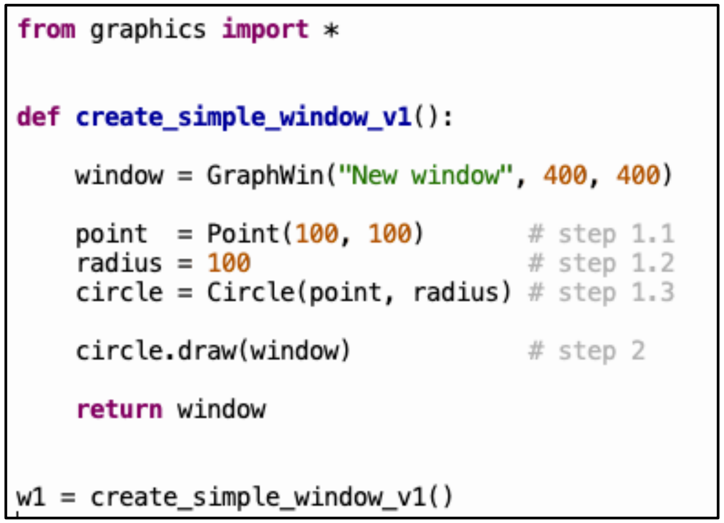
- Drawing a circle inside
  - how many variables do we need for a circle?



Drawing inside the window

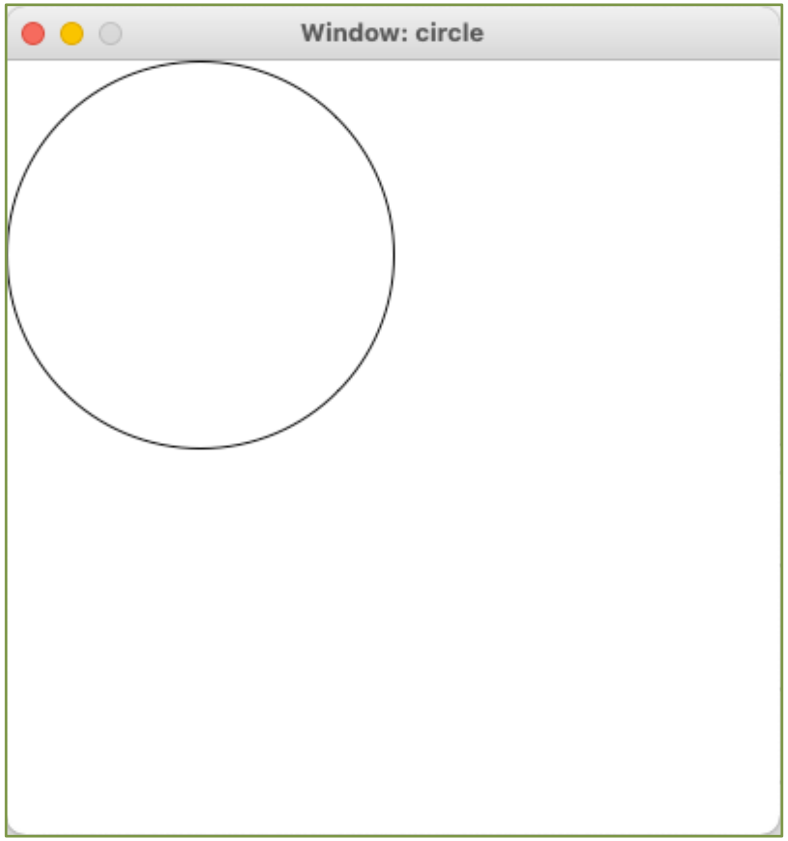
- Step 1: Construct a circle
    - Step 1.1: construct a point —> the center of the circle
    - Step 1.2: fix the radius
    - Step 1.3: put them together

- Step 2: **Draw** the newly constructed circle inside the window



```python
from graphics import *


def create_simple_window_v1():

    window = GraphWin("New window", 400, 400)

    point  = Point(100, 100)         # step 1.1
    radius = 100                     # step 1.2
    circle = Circle(point, radius)   # step 1.3

    circle.draw(window)              # step 2

    return window


w1 = create_simple_window_v1()
```

Coding demo

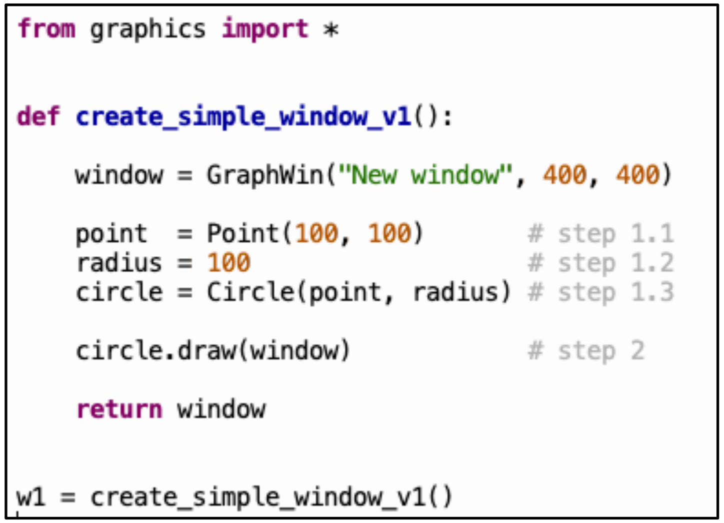


Exercise

- Write a function that draws a circle based on
  - user specified **center** (2D point)
  - user specified **radius**

- Extra credit: the size of the window can also be specified by the user

- What changes do you need to make?



```
from graphics import *


def create_simple_window_v1():

    window = GraphWin("New window", 400, 400)

    point  = Point(100, 100)        # step 1.1
    radius = 100                    # step 1.2
    circle = Circle(point, radius)  # step 1.3

    circle.draw(window)             # step 2

    return window


w1 = create_simple_window_v1()
```

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
        - Circle example discussed
        - Try Rectangle, Triangle

- **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 (02/15)