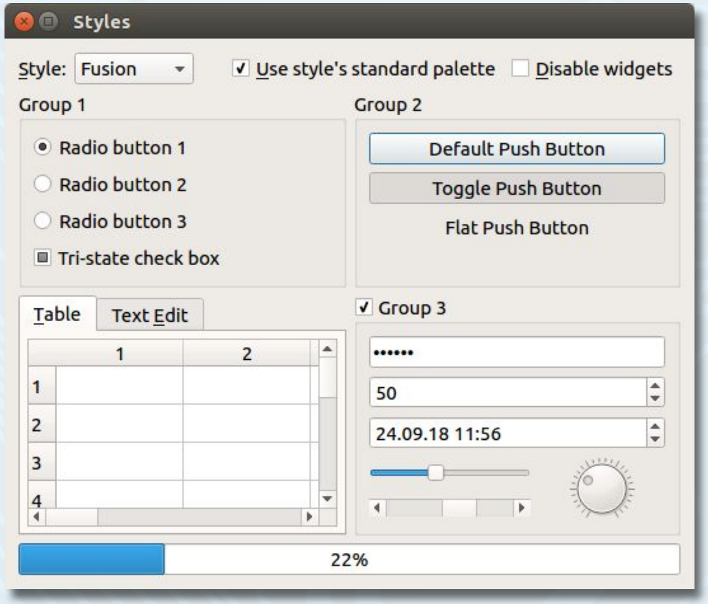
User Interfaces with Python (Tkinter)



11/12/2020

Getting Set Up

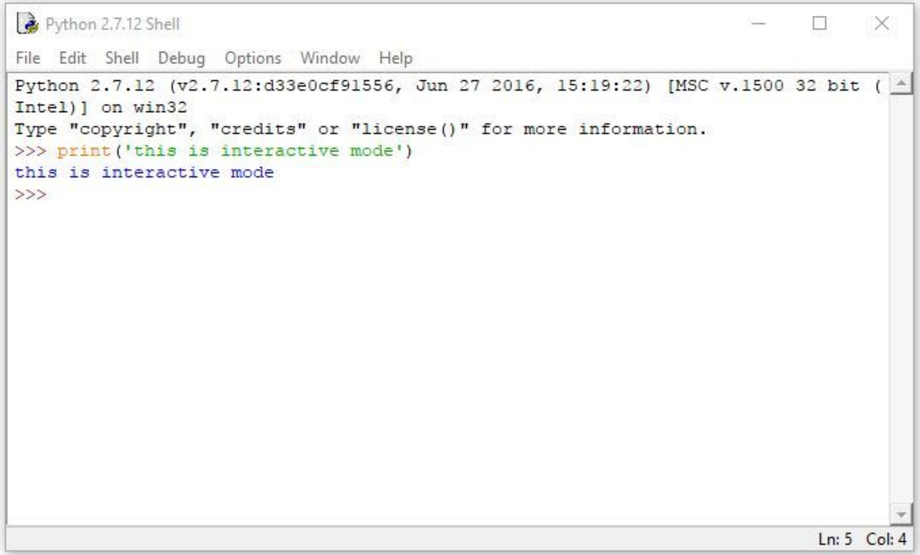
Install link: ***https://www.python.org/***

- *Make sure to select “Add to system PATH”*

Python Overview

Version 3.9

- Python is an object-oriented language
- Easy to learn and read
- Interactive mode
  - Command Line
- Large variety of libraries
  - Image processing
  - Plotting
  - GUI Programming
- Built-In IDE (IDLE)

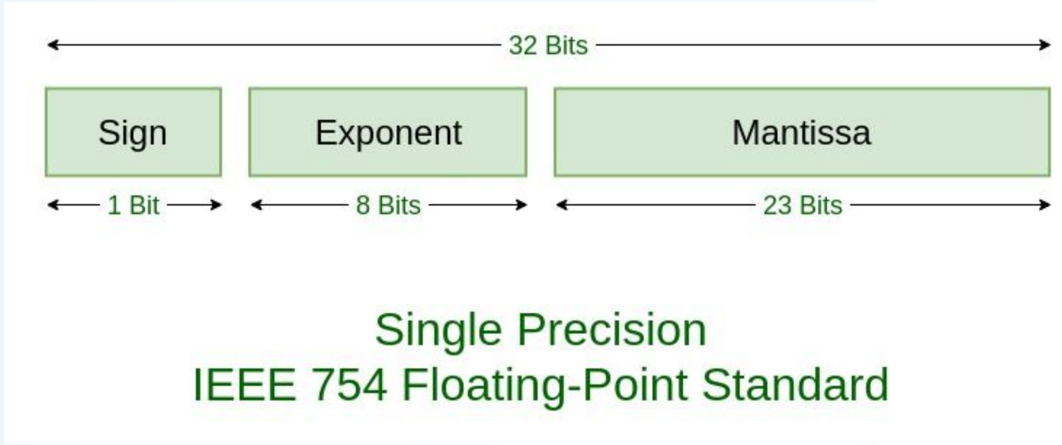


```
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (
Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print('this is interactive mode')
this is interactive mode
>>>
```

Basic Data Types

Numerical

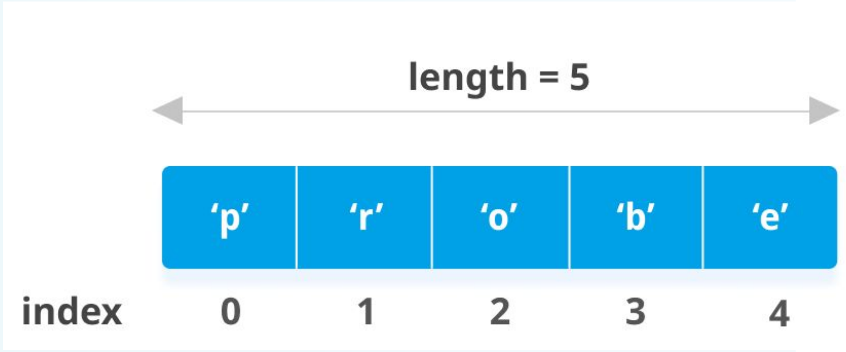
- Integers
  - Non-fractional numbers
  - -2, -1, 0, 1, 2
- Float
  - Floating point numbers
  - Numbers with a fractional component
  - If you’ve taken CMPEN 270 you already hate them



Basic Data Types

Sequences

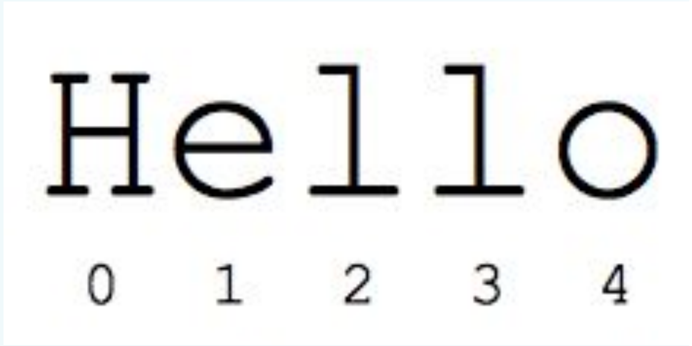
- List
  - Sequence of elements
  - Denoted with []
  - Get elements with indices
- Tuple
  - Less common
  - Denoted with ()
  - Like a list, but you can't change it after it has been declared



Basic Data Types

Other important types

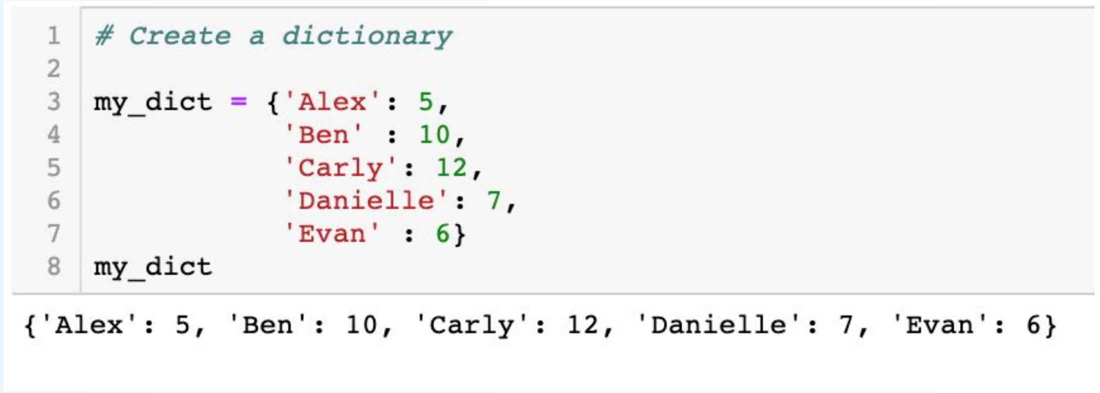
- String
  - Collection of characters
  - Denoted with “x” or ‘x’
  - Can be treated as a python list of characters
- Bool
  - True or False
  - Boolean values must be capitalized
    - “True” good
    - “true” bad



Basic Data Types

Some other python data types

- Dict
- Complex
- Range
- Byte
- And more!



```
# Create a dictionary

my_dict = {'Alex': 5,
           'Ben' : 10,
           'Carly': 12,
           'Danielle': 7,
           'Evan' : 6}
my_dict
```

```
{'Alex': 5, 'Ben': 10, 'Carly': 12, 'Danielle': 7, 'Evan': 6}
```

Python Syntax

Basics

- Comments
  - Python will skip any line that starts with a #
  - Triple quotes can be used for multi-line
- Variables
  - Python is not statically typed
  - Declare variables with =
  - Naming
    - Cannot start with a number
    - Can only contain A-z, 0-9, and _

```
#declare x as a float
x = 3.14

#declare x as a string
x = “hello world”

#declare x as a list
x = [3.14, “hello world”, True]

#and so on...
```

Python Syntax

Conditionals

- Python can compare values with <, >, <=, >=, !=, and ==
  - Returns True or False
- if-elif-else
  - Python does not use {}
- The code that you want the statement to run should be indented on the next lines

```
x = 5
y = 10

if x > y:
        print(“x is bigger”)
elif 2 != 3:
        print(“2 is not 3”)
else:
        #something else...
```

Python Syntax

Loops

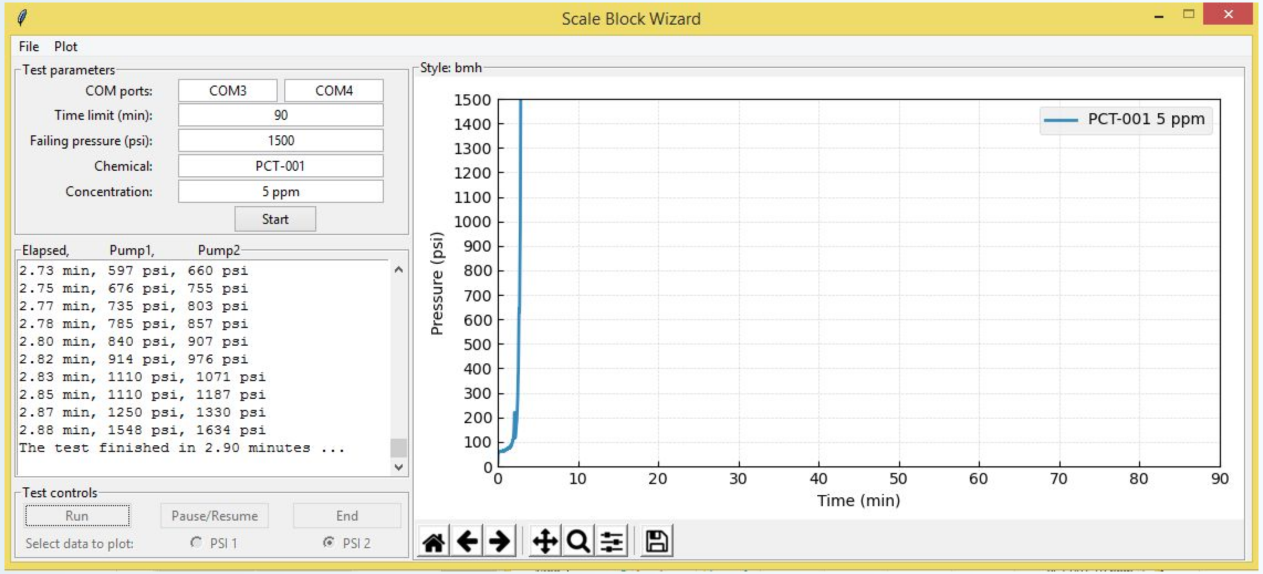
- While
  - while (condition):
- For
  - for (variable) in (iterable):

```
#do something forever
while True:
	print(“python”)

#print the numbers 0-9
for i in range(10):
	print(i)

#print the elements in a list x
x = [“python”, “java”, “c++”]
for j in x:
	print(j)
```

What can you do with Tkinter



Scale Block Wizard

File Plot

Test parameters

COM ports:	COM3	COM4
Time limit (min):	90	
Failing pressure (psi):	1500	
Chemical:	PCT-001	
Concentration:	5 ppm	

Start

Elapsed, Pump1, Pump2

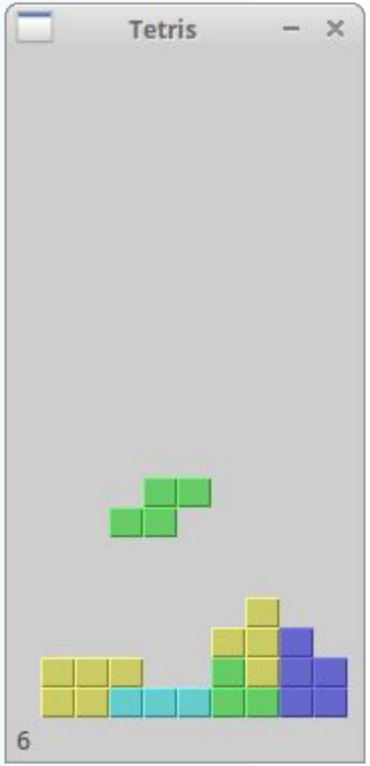
```
2.73 min, 597 psi, 660 psi
2.75 min, 676 psi, 755 psi
2.77 min, 735 psi, 803 psi
2.78 min, 785 psi, 857 psi
2.80 min, 840 psi, 907 psi
2.82 min, 914 psi, 976 psi
2.83 min, 1110 psi, 1071 psi
2.85 min, 1110 psi, 1187 psi
2.87 min, 1250 psi, 1330 psi
2.88 min, 1548 psi, 1634 psi
The test finished in 2.90 minutes ...
```

Test controls

Run Pause/Resume End

Select data to plot: PSI 1 [✓ PSI 2]

Style: bmh



Tkinter Terminology and Coding

- Coding in tkinter involves straightforward layering
  1. Initialize tkinter (import, set up class/constructor)
  2. Create widget
  3. Pack Widget
  4. Create main loop
- This can get more advanced if you want multiple frames
- You can also layer widgets within widgets

Widget: Standard GUI elements (buttons, menus, labels, entries, etc)

Pack: Method used to specify positioning of widgets within their container (left, right, top, bottom)

Frame: A widget used to contain other widgets, so that you can organize your GUI the way you want

Basic Tkinter Example

```
#tkinter is a native library to python, but you must import it to use it
import tkinter as tk

#instantiate tkinter class
root = tk.Tk()

#create widget
label = tk.Label(root, text = ‘Hello World’, padx=10, pady=10)

#pack widget
label.pack()

#create main loop
root.mainloop()
```

Making a GUI Aesthetic

- When making a GUI you may wish to change font color, size, etc
  - fg: foreground (text) color
  - bg: background color
  - bd: background size
  - font: font style of the text
  - size: size of text
- Lists of available colors and fonts can be found online

Example:

```
label = tk.Label(root,
          text = ‘Merry Christmas’,
          fg = ‘red’
          bd = 5
          bg = ‘green’
          font = (‘arial’, size = 10))
```

A more layered example...

```
import tkinter

#create the class and constructor
class ExampleGUI:
        def __init__(self):

                #create frames
                self.frame1 = tkinter.Frame()
                self.frame2 = tkinter.Frame()

                #frame1 widgets
                self.title = tkinter.Label(self.frame1, text = ‘Example’, bd = 15, bg = ‘blue’, fg = ‘red’, font = (‘arial’, 15))

                #pack frame1 widgets into frame1
                self.title.pack()

                #frame2 widgets
                self.prompt = tkinter.Label(self.frame2, text=’Enter your name:’)
                self.nameEntry = tkinter.Entry(self.frame2, width = 10)

                #pack frame2 widgets into frame2
                self.prompt.pack()
                self.nameEntry.pack()

                #pack frames into main window
                self.frame1.pack()
                self.frame2.pack()

                #enter main loop
                tkinter.mainloop()

#create instance of ExampleGUI
example = ExampleGUI()
```

How about something functional?

- Most programs you write should be interactive
  - Take an input → Show an output

- We’ll create a program for taking notes / ‘To-do’ list
  - Use what we’ve learned so far + advanced features