

PYTHON LANGUAGE For Humans

by

Christopher Topalian

Copyright 2021
All Rights Reserved

**DEDICATED
TO
GOD THE FATHER**

Search Google for: Python

A screenshot of a web browser window. The title bar shows "File Edit View History Bookmarks Tools Help". The address bar shows "python - Google Search" and the URL "https://www.google.com/search?client=". The search bar contains "python". Below the search bar is a button with a yellow border and white text that says "Search for Python". A yellow arrow points from the text "Search for Python" to the search bar. The search results page for "python" is displayed, showing the Google logo, the search term "python", a search button, and a link to "www.python.org". A yellow arrow points from the text "Left Click Downloads" to the "Downloads" link on the Python.org page. The Python.org page itself has a header with "Welcome to Python.org" and "The official home of the Python Programming Language.", a search bar, a "Downloads" link, and links for "Python 3.9.1 - Python 3.8.7 - Mac OS X" and "OS X - Python 3.9.0 - Python 3.7.9". To the right, there are links for "1. Whetting Your Appetite - 5. Dat Structures - 9. Classes - ...". The bottom of the browser window shows the URL "https://www.python.org/downloads/".

Download PYTHON

Download Python | Python.org X +

https://www.python.org/downloads/ ... ⚡ 3 ⌂

python™

Donate Search GO Socialize

About Downloads Documentation News Events

Left Click Download Python 3.9.1

Download the latest version of Python

Download Python 3.9.1

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [Mac OS X](#)

Want to help? [images](#)

Opening python-3.9.1-amd64.exe

You have chosen to open:

python-3.9.1-amd64.exe
which is: exe File (26.9 MB)
from: https://www.python.org

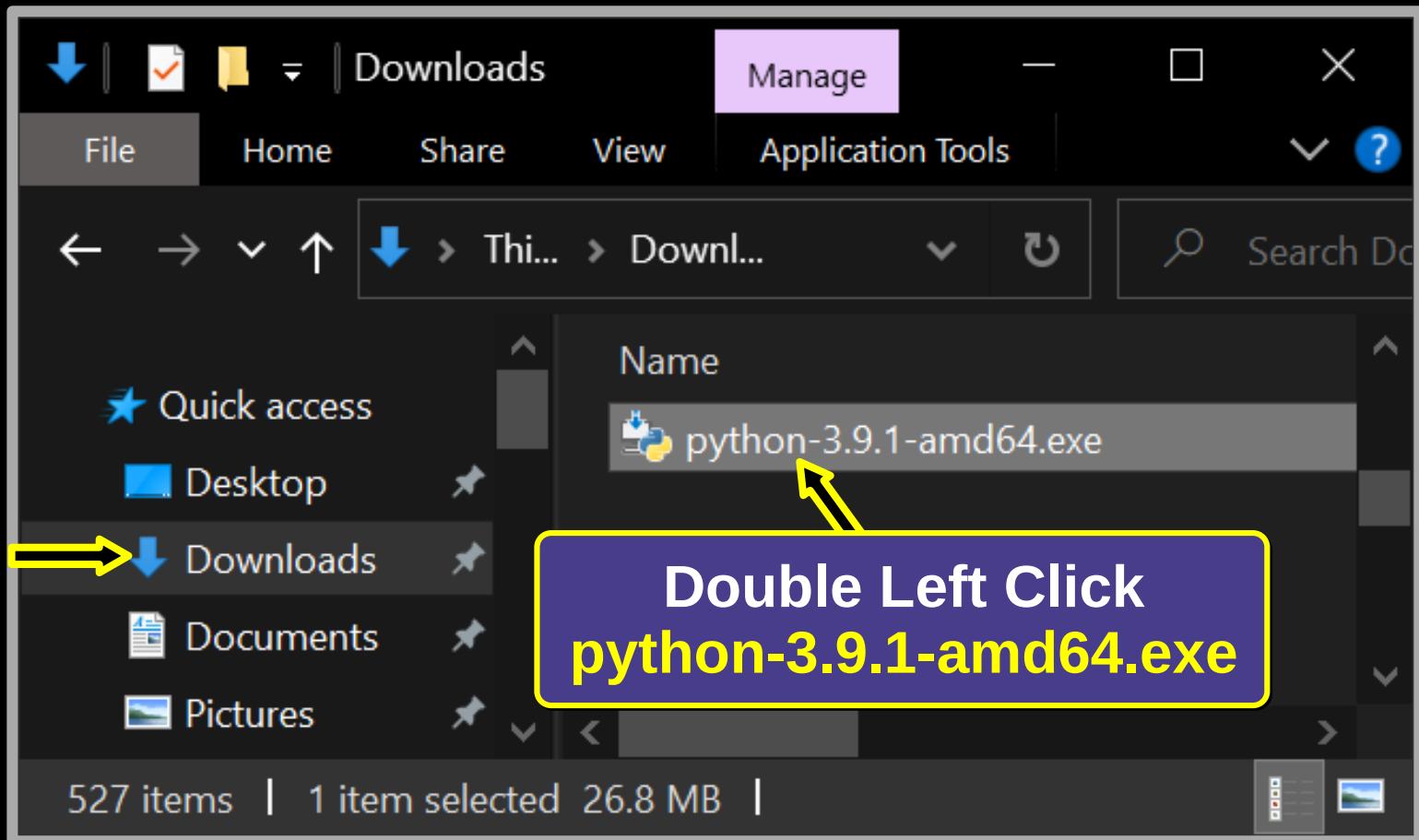
Would you like to save this file?

Left Click Save File

Save File **Cancel**

COLLEGE OF SCRIPTING MUSIC & SCIENCE

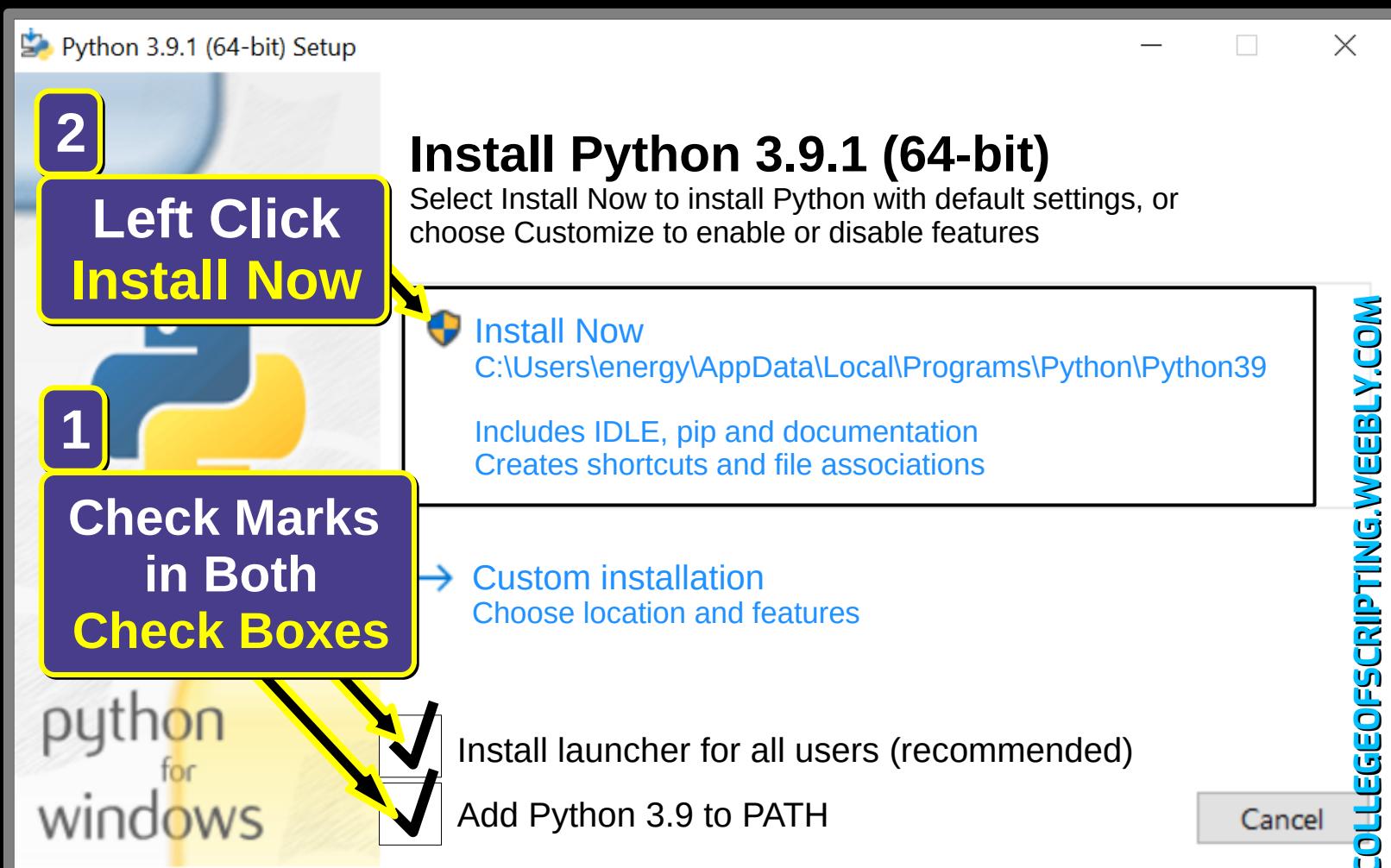
Go to Your Downloads Folder



Go to your **Downloads Folder** and then
Double Left Click on the file named
python-3.9.1-amd64.exe

This will begin the installation process.

Install Python



**Make sure to put a CHECK MARK
in BOTH Boxes!**

**These options are very important
to all of our projects!**

Setup was successful

Python 3.9.1 (64-bit) Setup

Setup was successful

New to Python? Start with the [online tutorial](#) and [documentation](#). At your terminal, type "py" to launch Python, or search for Python in your Start menu.

See [what's new](#) in this release, or find more info about [using Python on Windows](#).

Disable path length limit

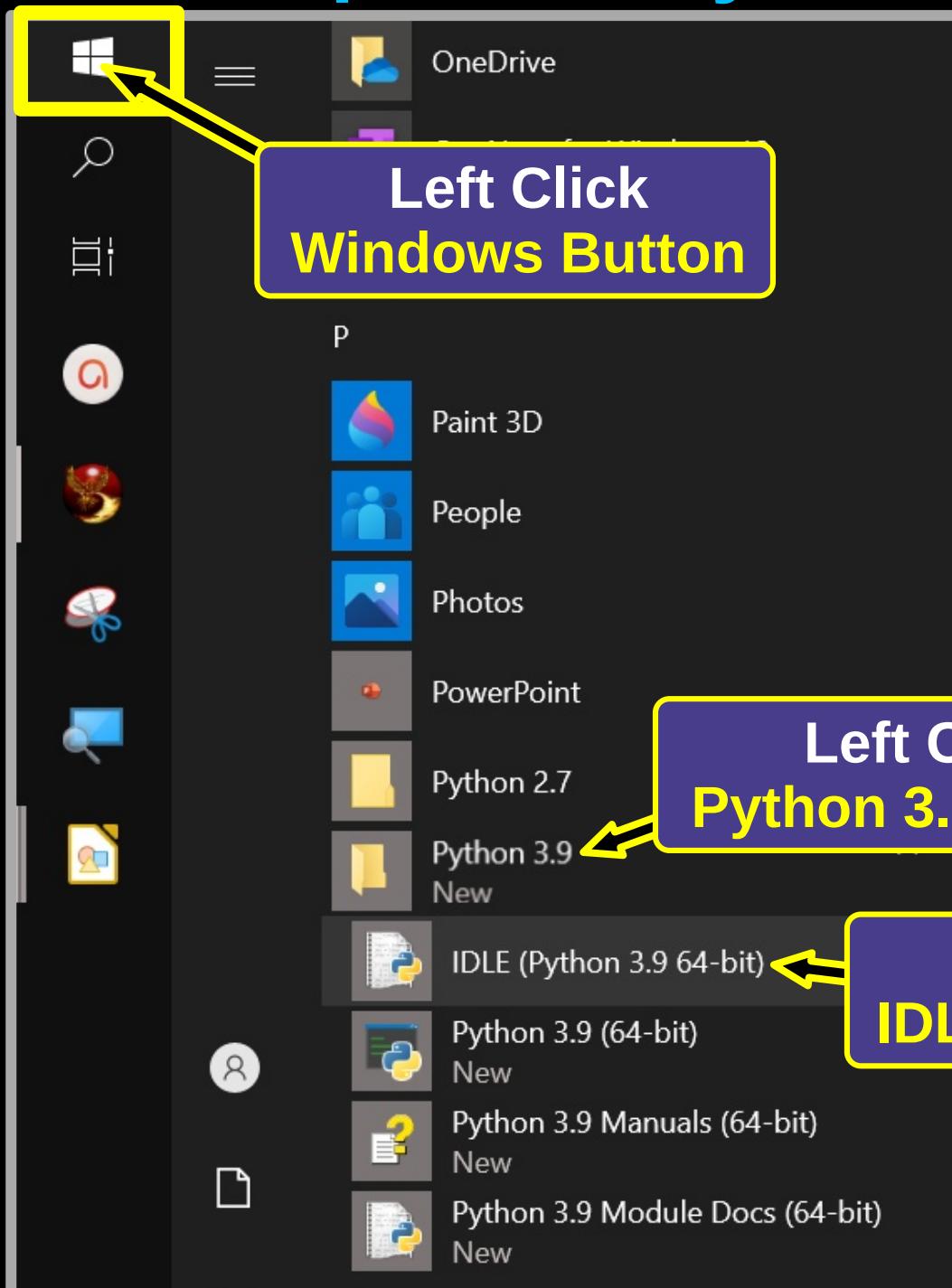
Changes your machine configuration to allow programs, including Python, to bypass the 260 character "MAX_PATH" limitation.

Should I Disable path length limit?

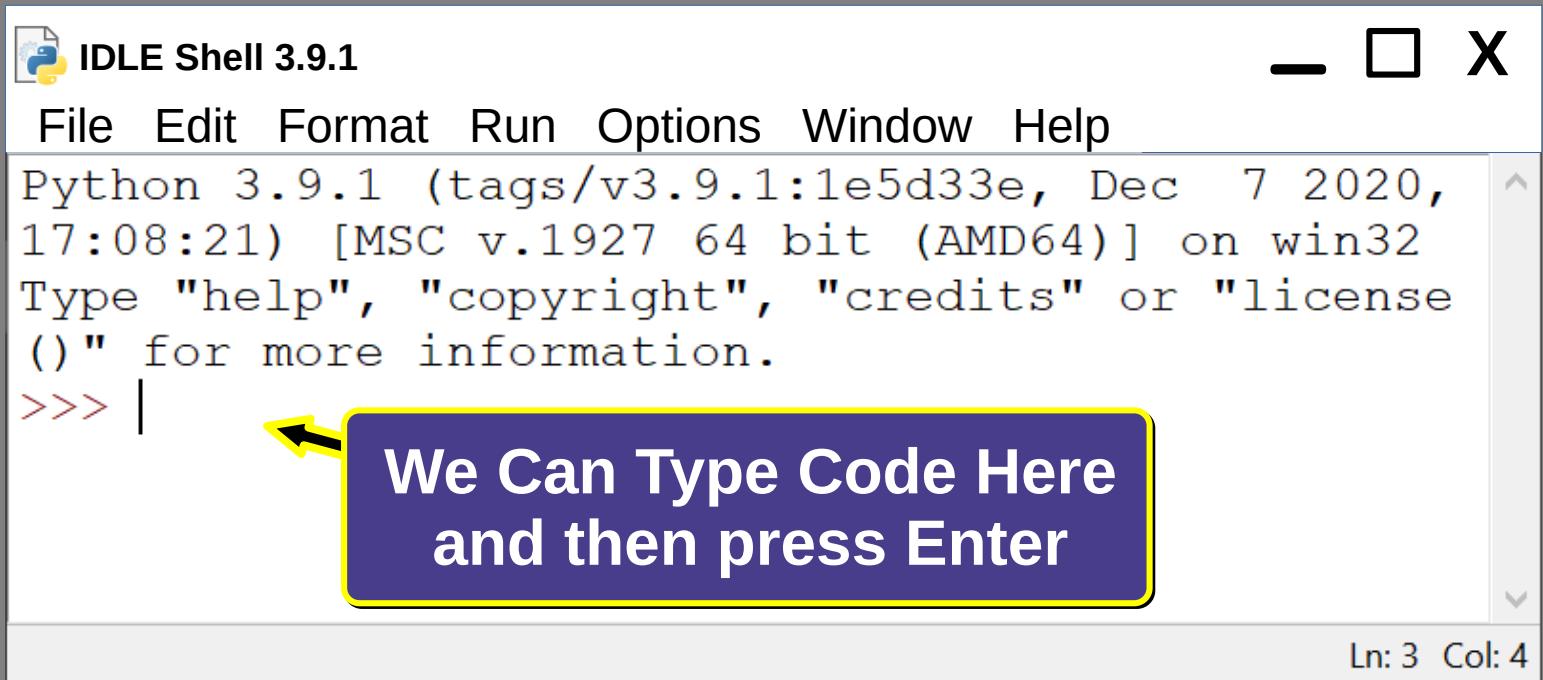
Close

We don't need to disable the path length limit. Just make sure to keep our folder names and file names less than 260 characters long, which is easy. But, if you plan on using long names, disable the limit.

Open the Python IDLE



Python IDLE Shell Opened



IDLE Shell 3.9.1

File Edit Format Run Options Window Help

Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> |

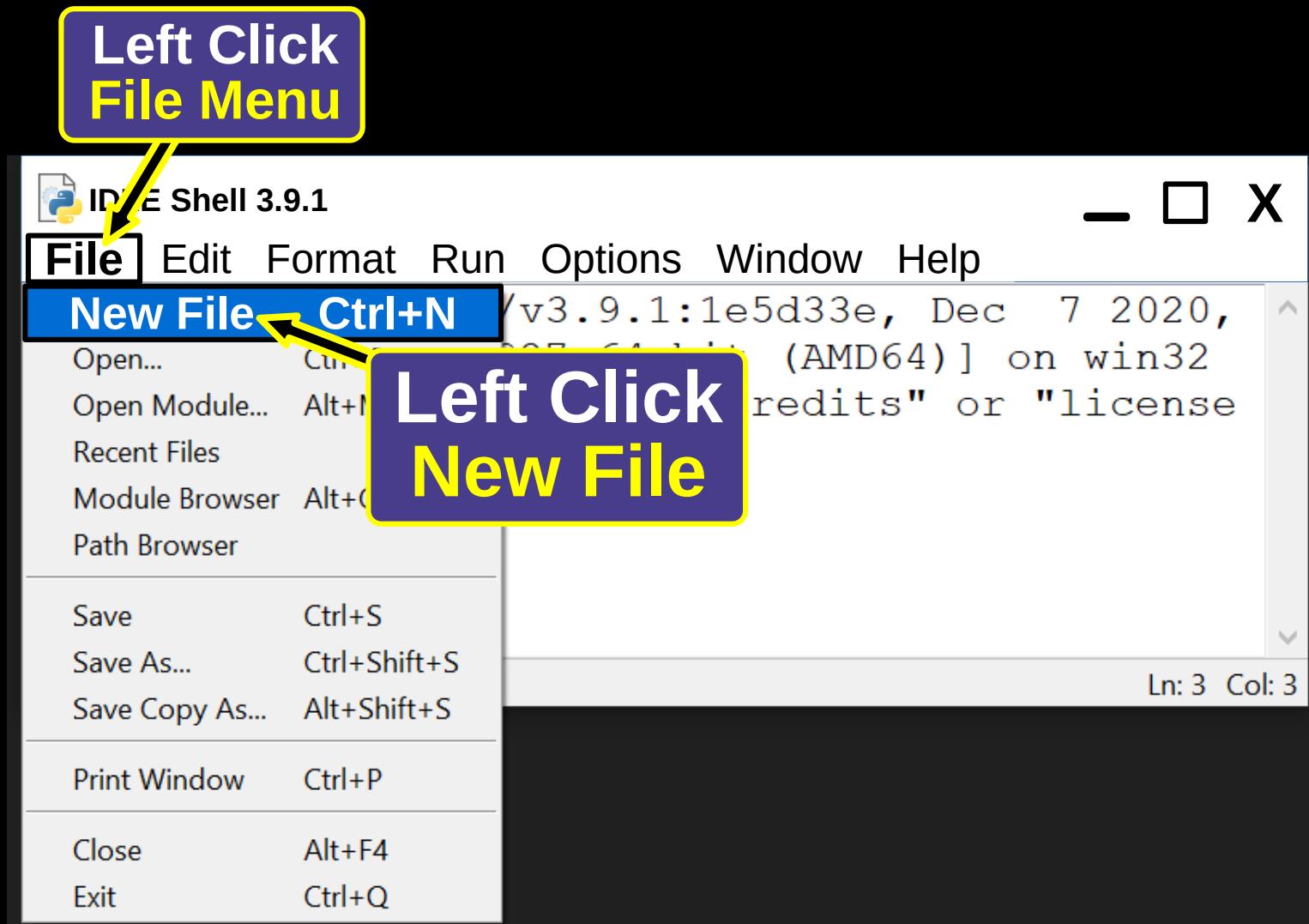
We Can Type Code Here
and then press Enter

Ln: 3 Col: 4

We can type our code here in the Shell,
but there is a much better way to program!

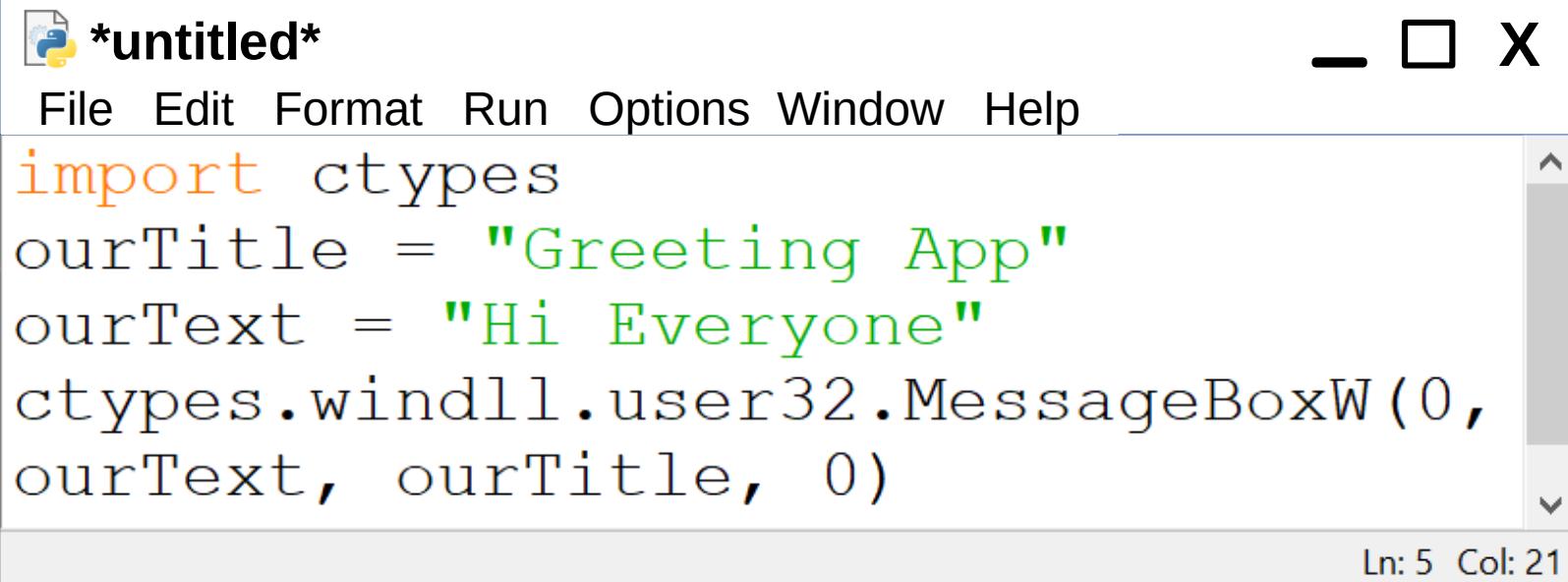
Instead of writing code one line at a time,
and pressing the Enter Button,
we will instead make a **New File**,
as shown on the Next Tutorial.

Python IDLE Create a New File



We make a **New File** to write our code in.
We use the **File Menu** and choose **New File**.
The Shortcut is **Ctrl + N**

We Type Our Code in Our New File



untitled

File Edit Format Run Options Window Help

```
import ctypes
ourTitle = "Greeting App"
ourText = "Hi Everyone"
ctypes.windll.user32.MessageBoxW(0,
ourText, ourTitle, 0)
```

Ln: 5 Col: 21

We type our code
in the New File we made.

TYPE the code
as you see it **above**.

On the Next Tutorial page,
we show this code much bigger.

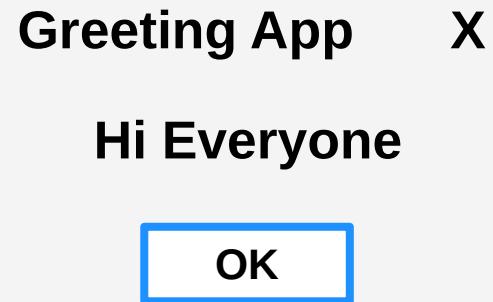
user32 MessageBoxW

```
import ctypes
```

```
ourTitle = "Greeting App"
```

```
ourText = "Hi Everyone"
```

```
ctypes.windll.user32.MessageBoxW(0,  
ourText, ourTitle, 0)
```



We want to create a **Message box**,
but we first have to import the module **ctypes**,
to give us access to **user32** functions from **windll**

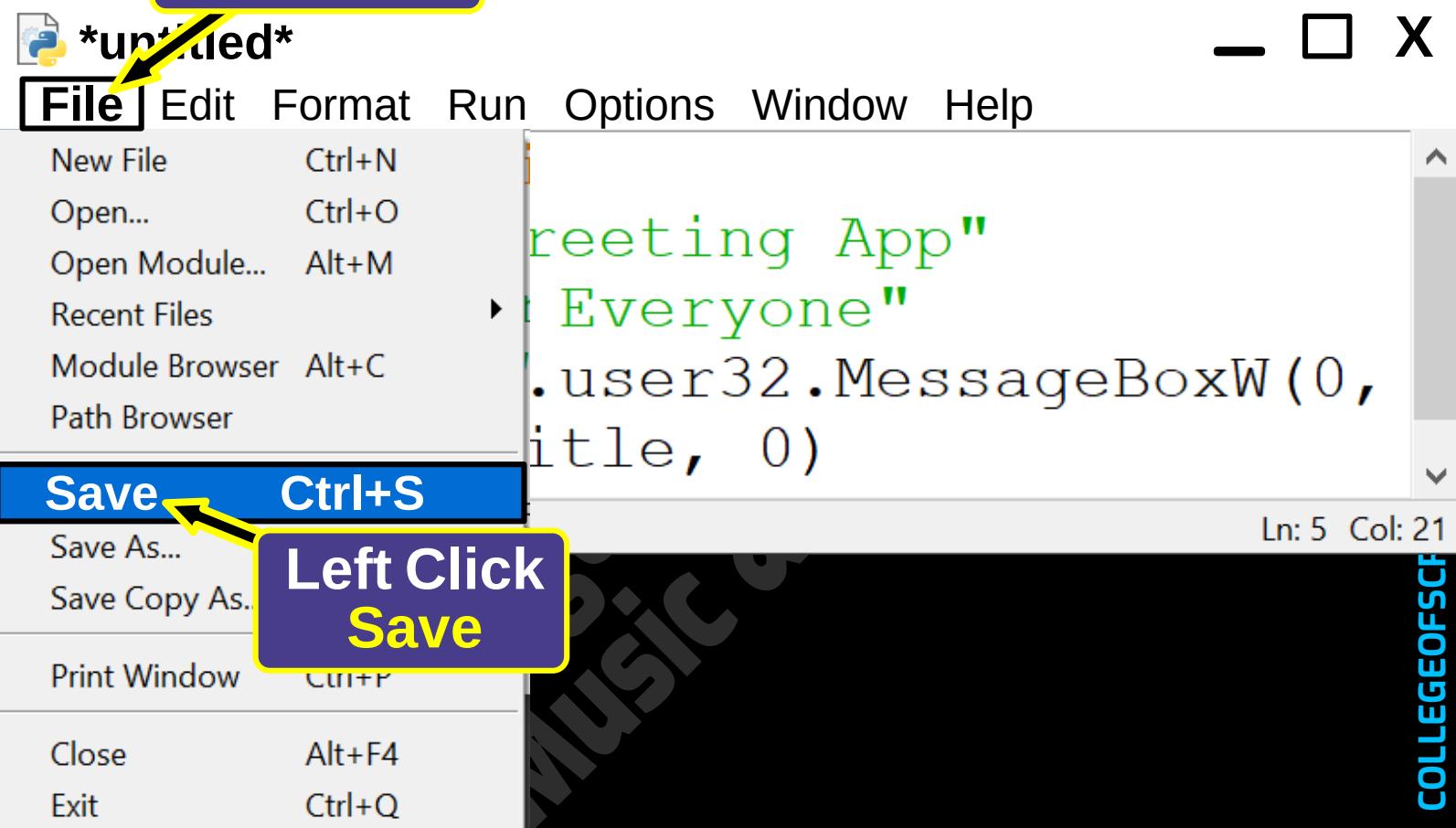
two lowercase L

We import the **ctypes** module,
which is a native built-in standard Python module.

The **ctypes** module allows us to use the function
MessageBoxW, from the **user32** library of **windll**

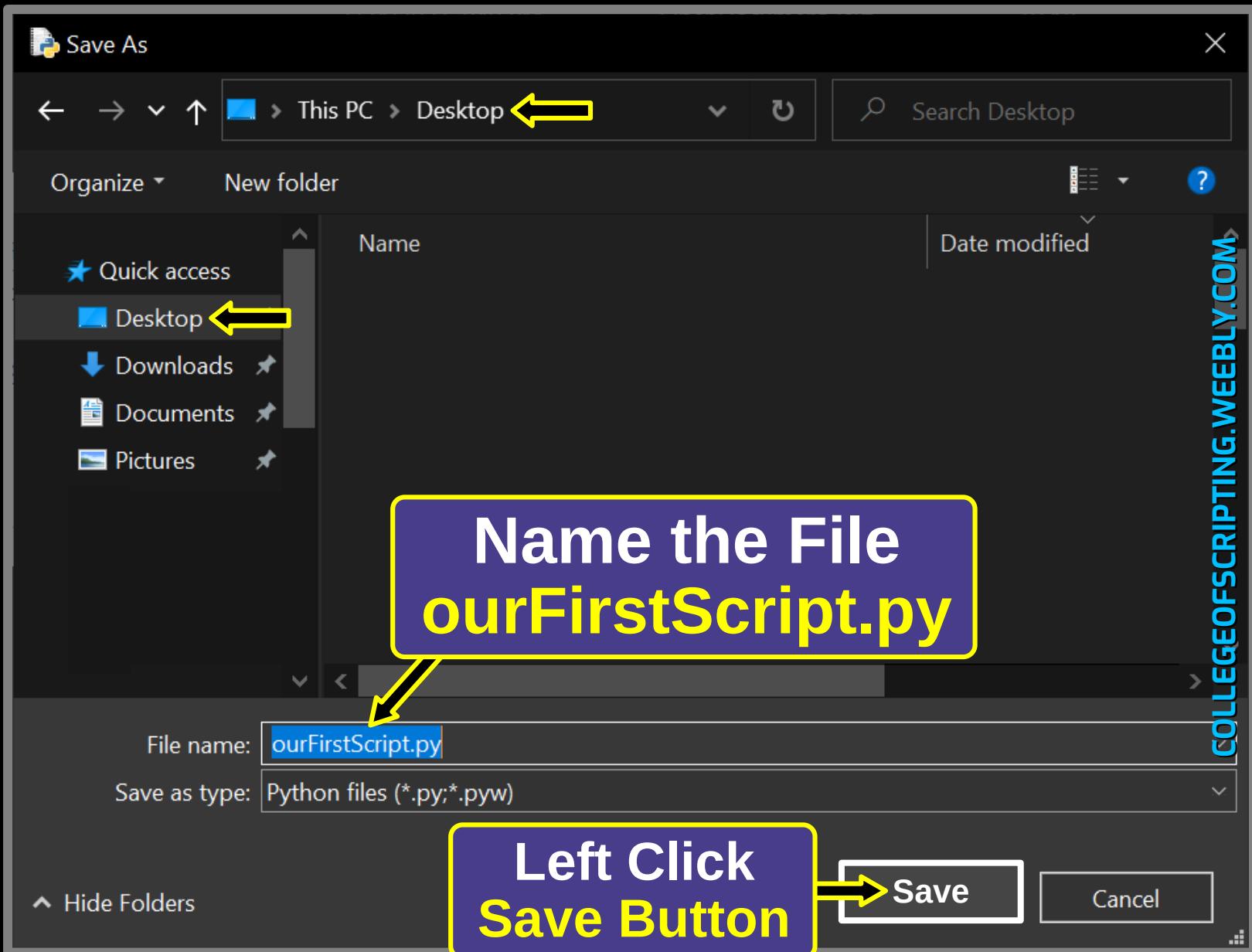
Save Our New File

Left Click
File Menu



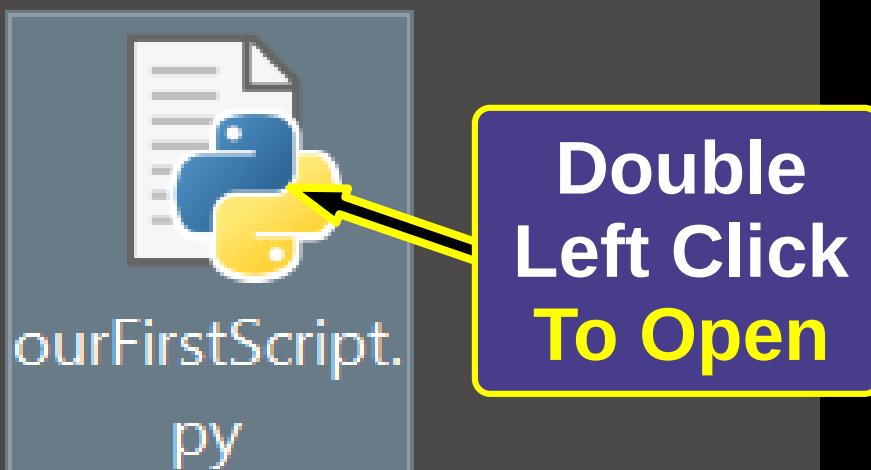
We Save our File using the File Menu.
We use File Menu and choose Save File.
The Shortcut is Ctrl + S

Naming & Saving Our Python File



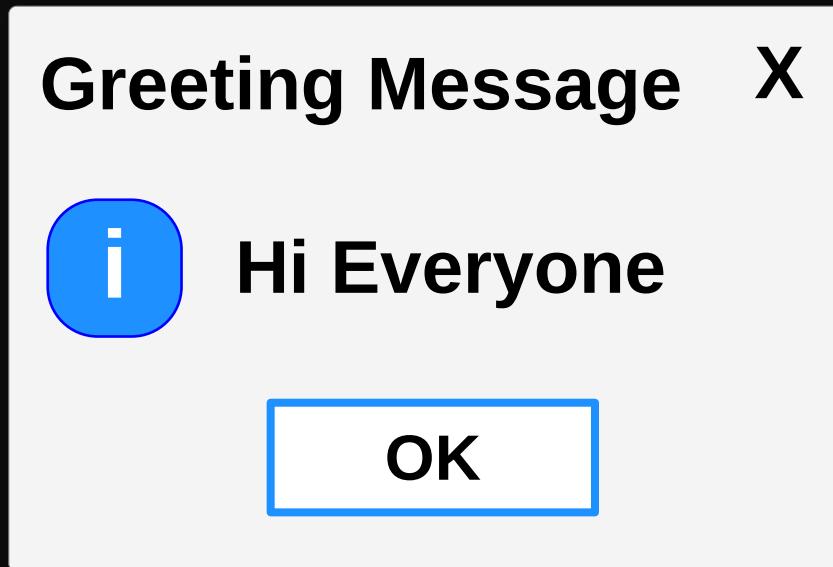
Open Our File `ourFirstScript.py`

In the previous tutorial, we **saved** our file named `ourFirstScript.py` on our **Desktop**. Look on the Desktop for `ourFirstScript.py`



Our Working App! `ourFirstScript.py`

 C:\WINDOWS\py.exe



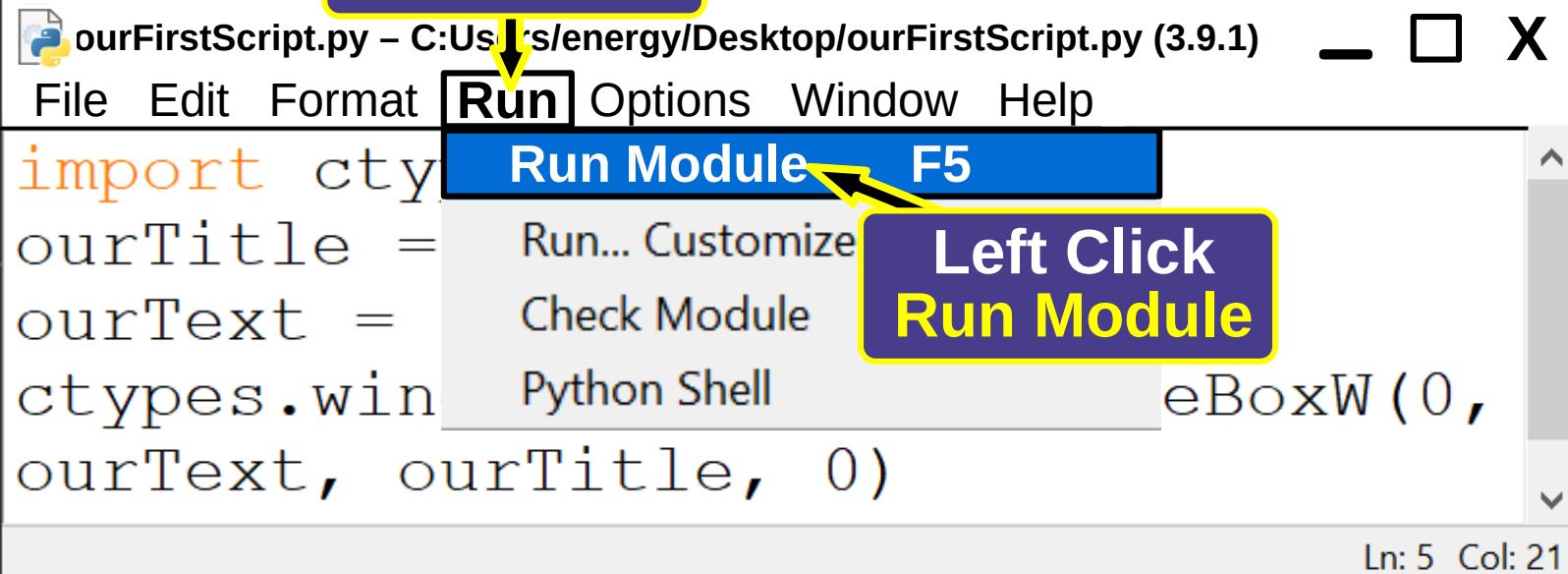
As you can see above, our Python Script works! We created a message box, with the title of **Greeting Message**, and with the text of **Hi Everyone**.

Next, we learn to **RUN** our code using **F5**

Run Our Script **ourFirstScript.py**

Christopher Topalian

**Left Click
Run Menu**

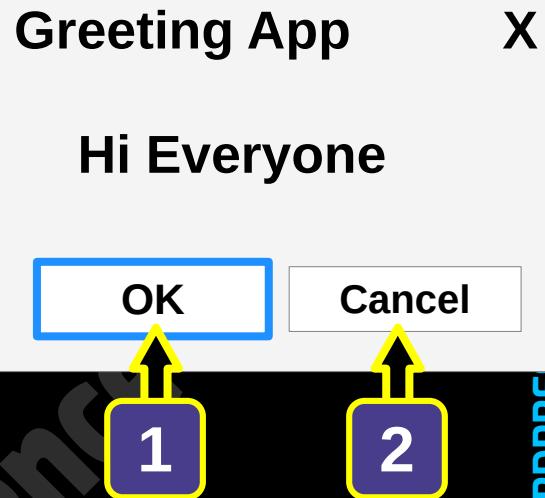


We can Run our **Script** using the **Run Menu**.
We **Left Click** the **Run Menu** and choose
Run Module. The Shortcut is **F5**

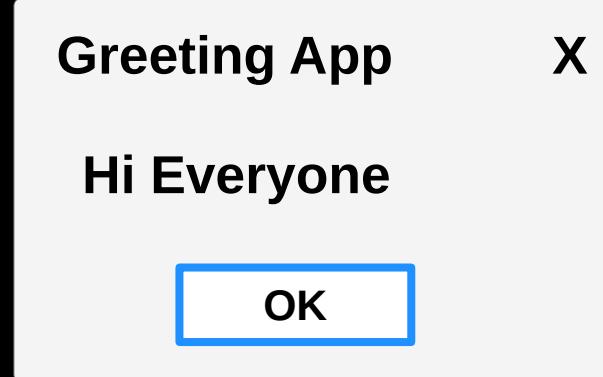
The Run module makes testing code easy,
but remember, the Run Module is NOT as
reliable as Double clicking to check for issues.
We Always Double Click our scripts to
ensure that they work, before sharing them
with other people.

Message Box 2 Button Choices

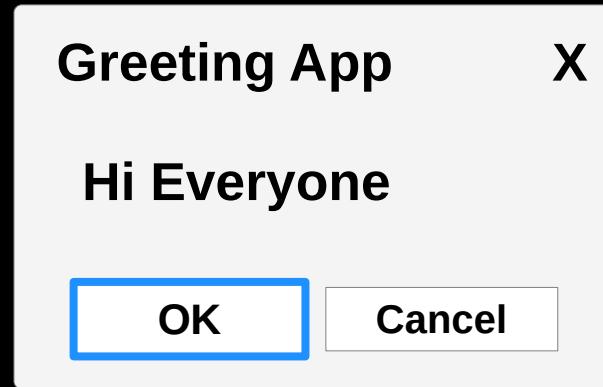
```
import ctypes  
  
ourTitle = "Greeting App"  
  
def ourFunction():  
    ourText = "Ready?"  
    return ourText  
  
choice = ctypes.windll.user32.MessageBoxW(0,  
ourFunction(), ourTitle, 1)  
  
if (choice == 1):  
    print("pressed ok")  
  
if (choice == 2):  
    print("pressed cancel")  
  
input('Press Enter to Exit')
```



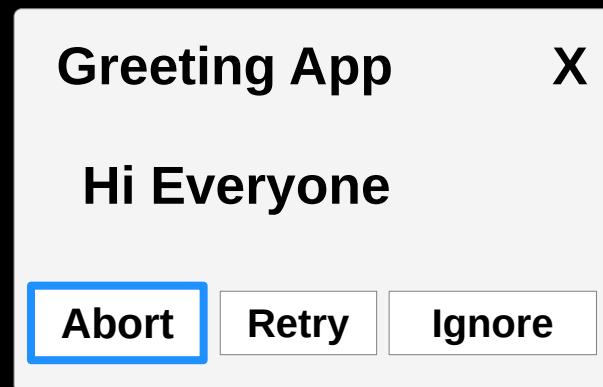
Stays open,
until user
presses Enter



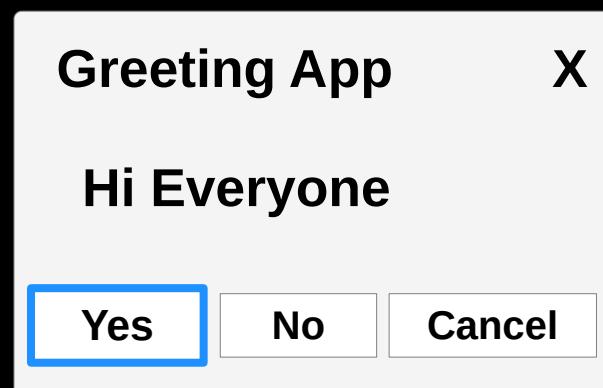
**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 0)**



**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 1)**

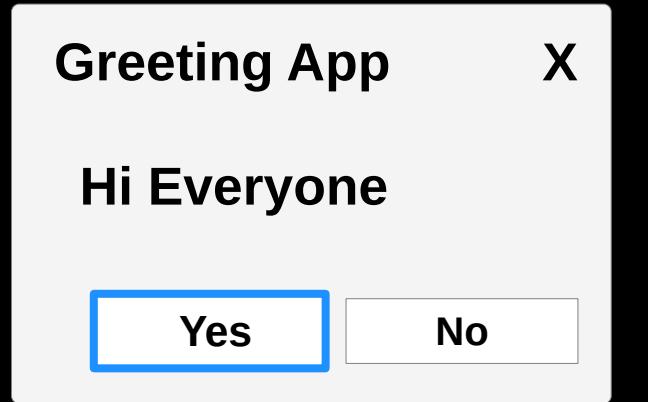


**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 2)**

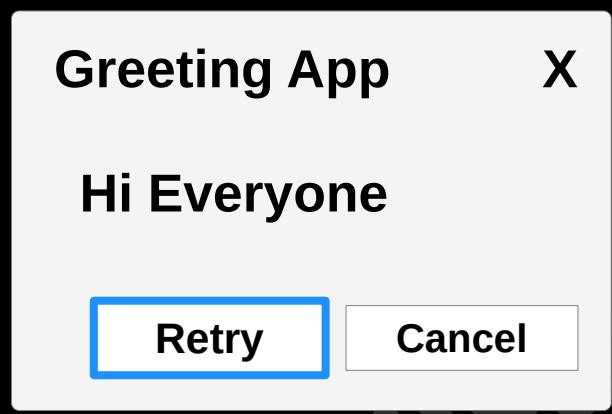


**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 3)**

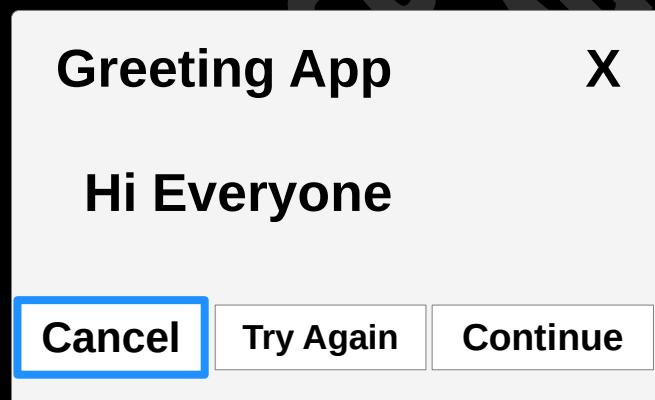




**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 4)**



**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 5)**

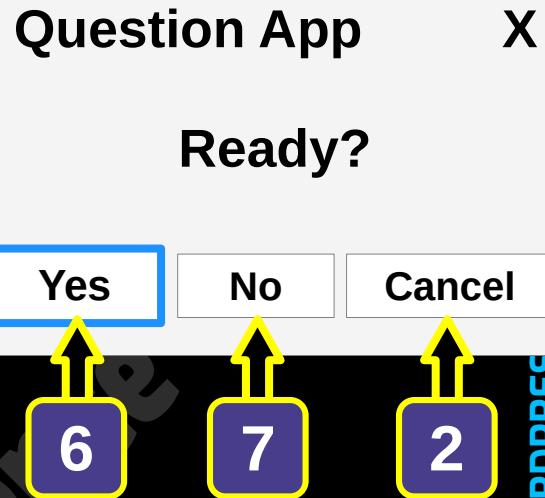


**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 6)**



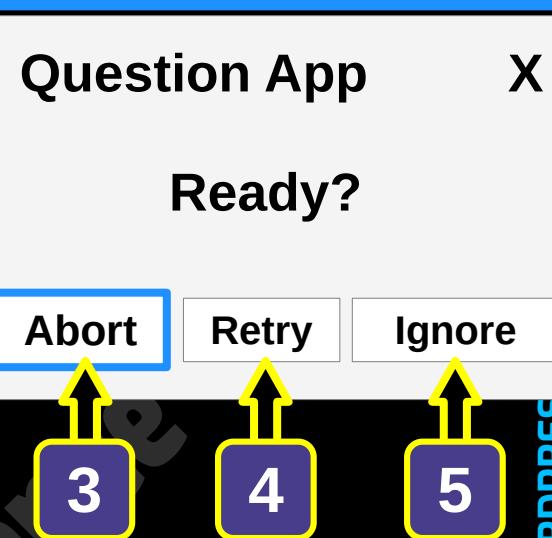
MessageBox 3 Button Choices

```
import ctypes  
  
ourTitle = "Question App"  
  
def ourFunction():  
    ourText = "Ready?"  
    return ourText  
  
choice = ctypes.windll.user32.MessageBoxW(0,  
ourFunction(), ourTitle, 3)  
  
if (choice == 6):  
    print("pressed Yes")  
  
if (choice == 7):  
    print("pressed No")  
  
if (choice == 2):  
    print("pressed Cancel")  
  
input('Press Enter to Exit')
```



MessageBox 3 Button Choices

```
import ctypes  
  
ourTitle = "Question App"  
  
def ourFunction():  
    ourText = "Ready?"  
    return ourText  
  
choice = ctypes.windll.user32.MessageBoxW(0,  
ourFunction(), ourTitle, 2)  
  
if (choice == 3):  
    print("pressed Abort")  
  
if (choice == 4):  
    print("pressed Retry")  
  
if (choice == 5):  
    print("pressed Ignore")  
  
input('Press Enter to Exit')
```



MessageBox 3 Button Choices

```
import ctypes
```

```
ourTitle = "Question App"
```

```
def ourFunction():
    ourText = "Ready?"
    return ourText
```

```
choice = ctypes.windll.user32.MessageBoxW(0,
    ourFunction(), ourTitle, 6)
```

```
if (choice == 2):
    print("pressed Cancel")
```

```
if (choice == 10):
    print("pressed Try Again")
```

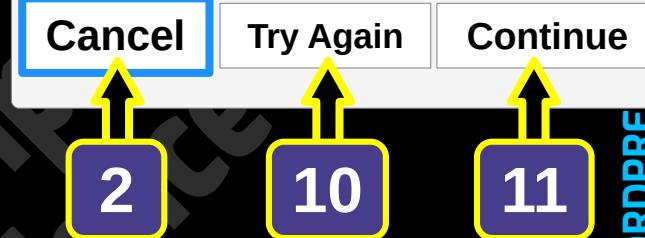
```
if (choice == 11):
    print("pressed Continue")
```

```
input('Press Enter to Exit')
```

Question App

X

Ready?



Learn more about
message box in the
reference section at
the end of this book

Python **Keywords**

and	del	import	return
as	elif	in	True
assert	else	is	try
async	except	lambda	while
await	False	None	with
break	finally	nonlocal	yield
class	for	not	
continue	from	or	
def	global	pass	
	if	raise	

Do NOT use these **keywords as names for your variables, functions, or classes**

Christopher Topalian
vars()
zip()
__import__()

Python Built-In Func- -tions

abs()
all()
any()
ascii()
bin()
bool()
breakpoint()
bytearray()
bytes()
callable()
chr()
classmethod()
compile()
complex()
delattr()
dict()
dir()
divmod()
enumerate()
eval()
exec()
filter()
float()
format()
frozenset()
getattr()
globals()
hasattr()
hash()
help()
hex()
id()
input()
int()
isinstance()
issubclass()
iter()
len()
list()
locals()
map()
max()
memoryview()
min()
next()
object()
oct()
open()
ord()
pow()
print()
property()
range()
repr()
reversed()
round()
set()
setattr()
slice()
sorted()
staticmethod()
str()
sum()
super()
tuple()
type()

Python IDLE Editor Bigger Font Size & Dark Theme

Configure IDLE

Left Click Options

ourFirstScript.py – C:\Users\ener\PycharmProjects\PythonForMusicians\ourFirstScript.py (3.9.1) _ X

File Edit Format Run Options Window Help

import ctypes
ourTitle = "Our First Script"
ourText = "Hello World!"
ctypes.windll.user32.MessageBoxW(0,
ourText, ourTitle, 0)

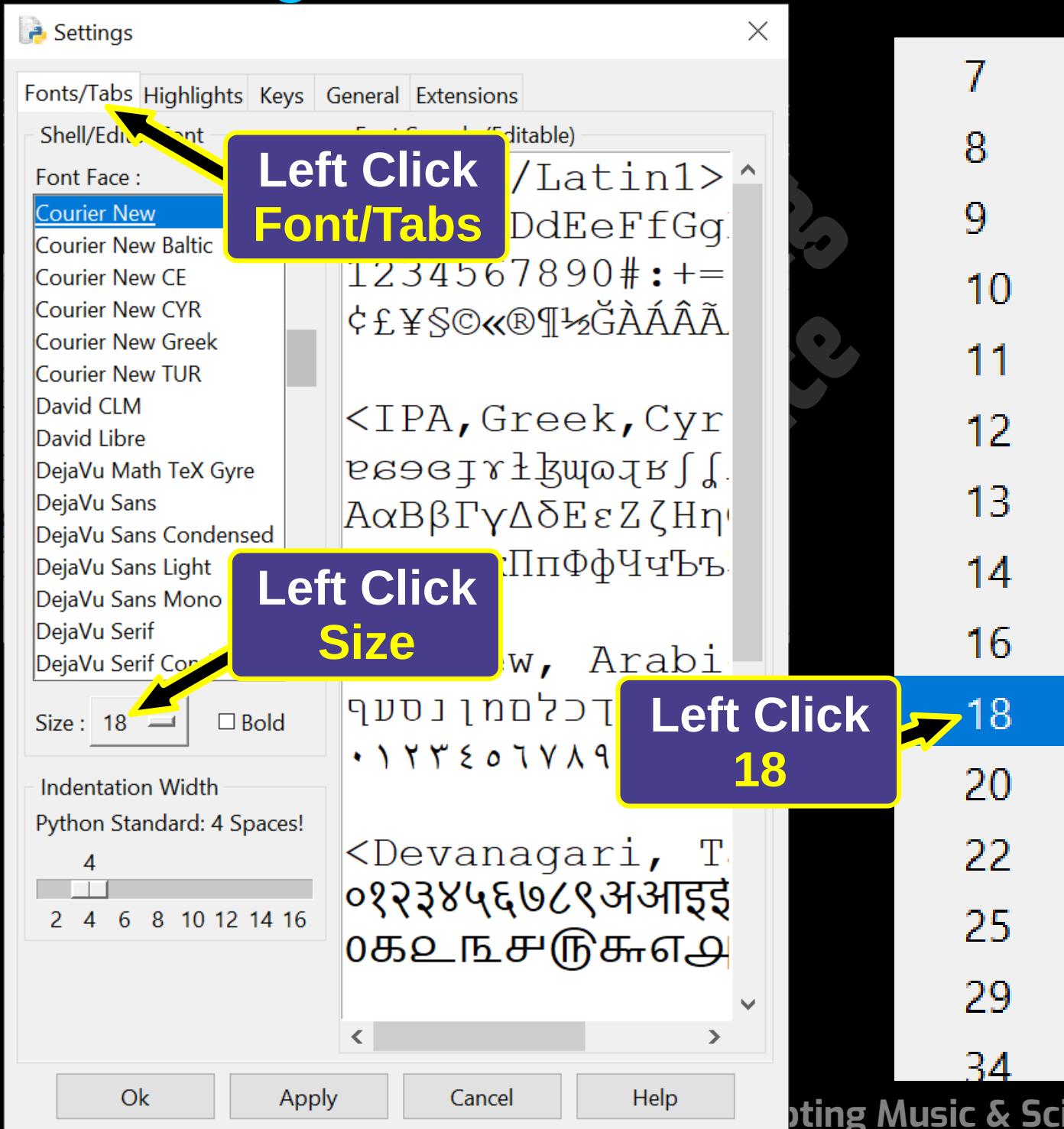
Ln: 5 Col: 21

Left Click Configure IDLE

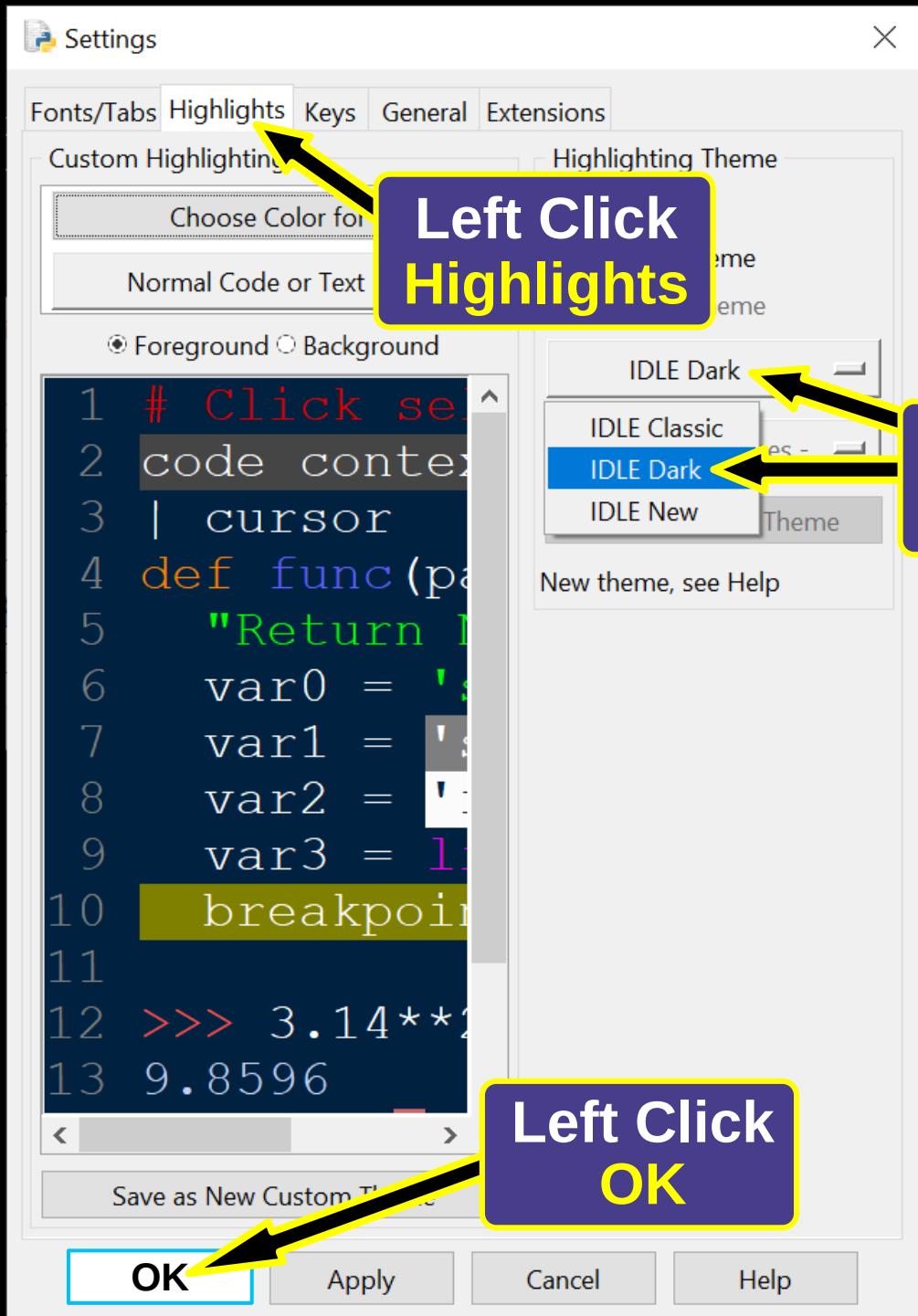
We will Change the THEME of our Editor.
First, we Open the Options Menu.
Then we choose Configure IDLE.

We will change the Size of the Font and
we will change the Theme to DARK,
as shown on the next tutorial pages.

Configure IDLE – Font Size



Configure IDLE – DARK Theme



DARK THEME Shown

```
import ctypes
ourTitle = "Greeting App"
ourMessage = "Hi Everyone"
choice=ctypes.windll.user32.MessageBoxW(0,
ourMessage, ourTitle, 0)|
```

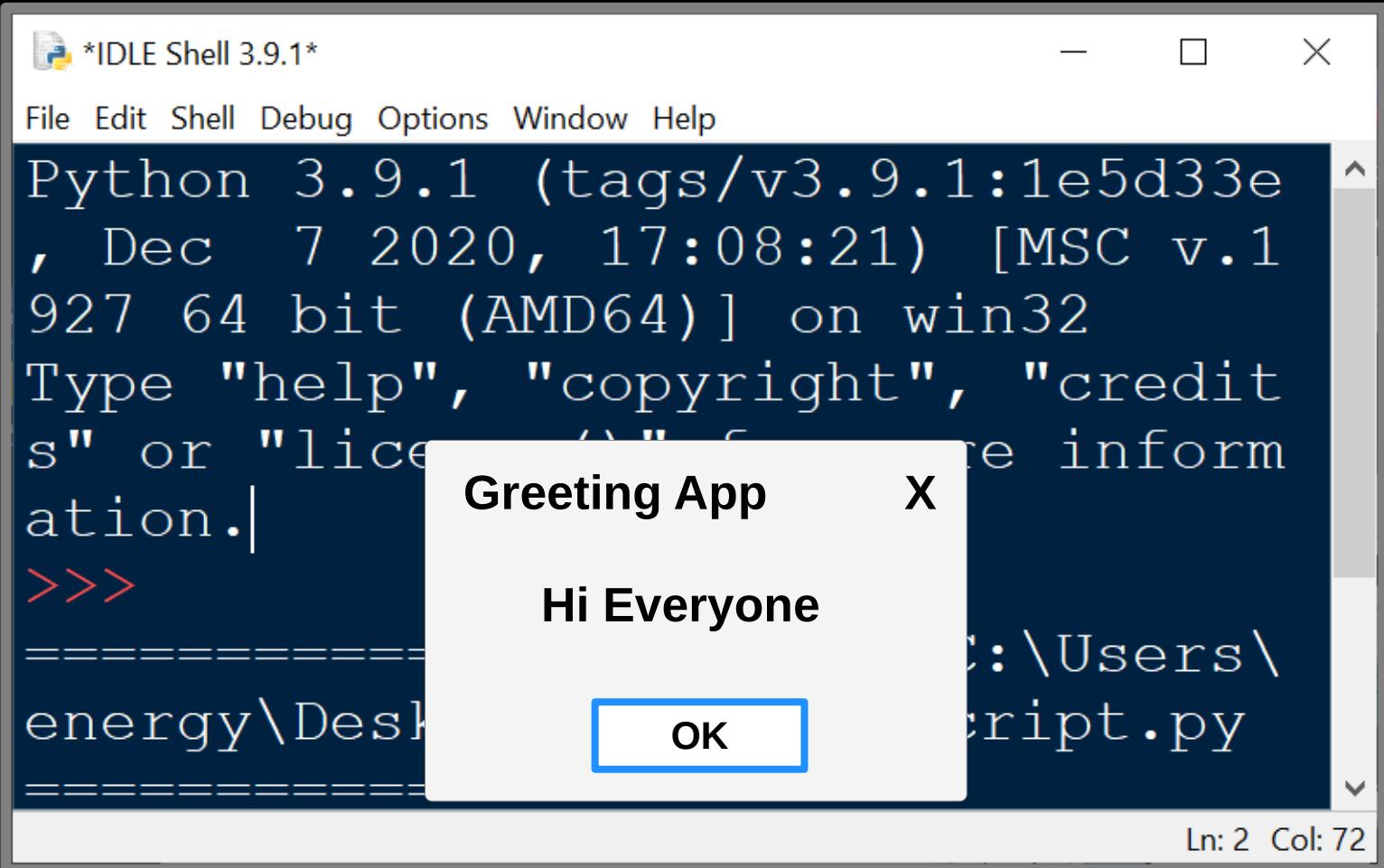
Ln: 5 Col: 24

The Dark Theme is very useful for long duration coding. We recommend using the dark theme for an easier time coding.

A bright white screen can be far too bright on many computer monitors and TV sets.

This Dark theme also changes the Python Shell Background, as shown on the next page.

IDLE Shell shown as Dark Theme



The Dark Theme is much kinder to Eyes!
As you can see here, it is a calmer setting.
It's very nice when we run our scripts, that
there is **not** a sudden bright white screen,
that appears as the script runs.

MATH

College of Scripting
Music & Science

math.pi

```
import ctypes  
import math
```

```
ourTitle = "Value of Pi"
```

```
def ourFunction():  
    ourText = math.pi  
    answer = str(ourText)  
    return answer
```

```
ctypes.windll.user32.MessageBoxW(0,  
ourFunction(), ourTitle, 0)
```

Value of Pi

X

3.141592653589793

OK

3.141592653589793

math.floor

```
import ctypes  
import math
```

```
ourTitle = "Floor App"
```

```
def ourFunction():  
    ourText = math.floor(4.45)  
    answer = str(ourText)  
    return answer
```

```
ctypes.windll.user32.MessageBoxW(0,  
ourFunction(), ourTitle, 0)
```

Floor App

X

4

OK

4.45 becomes 4
4.55 becomes 4
4.95 becomes 4

math.ceil

```
import ctypes
import math

ourTitle = "Ceil App"

def ourFunction():
    ourText = math.ceil(4.25)
    answer = str(ourText)
    return answer

ctypes.windll.user32.MessageBoxW(0,
ourFunction(), ourTitle, 0)
```

Ceil App

X

5

OK

4.25 becomes 5
4.45 becomes 5
4.55 becomes 5

round

```
import ctypes
```

```
ourText = round(24.34, 1)
```

```
ourTitle = "Rounding App"
```

```
ourText = str(ourText) <-- Convert to String
```

```
ctypes.windll.user32.MessageBoxW(0,  
ourText, ourTitle, 0)
```

round to 1 place
after decimal

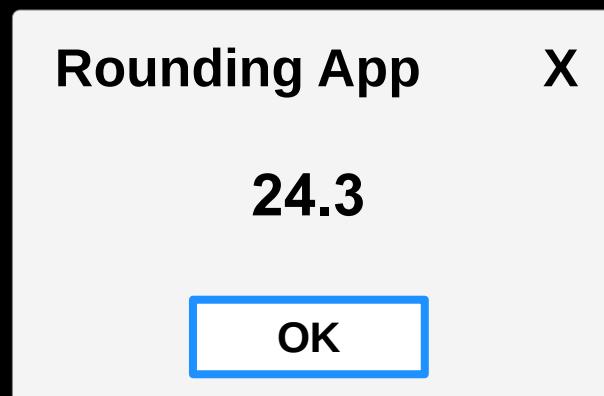
We round the decimal number 24.34
to 1 place after the decimal point.

23.34, rounds to 23.3

23.35, rounds to 23.4

23.347 rounds to 23.3

23.351 rounds to 23.4



pow

```
import ctypes
```

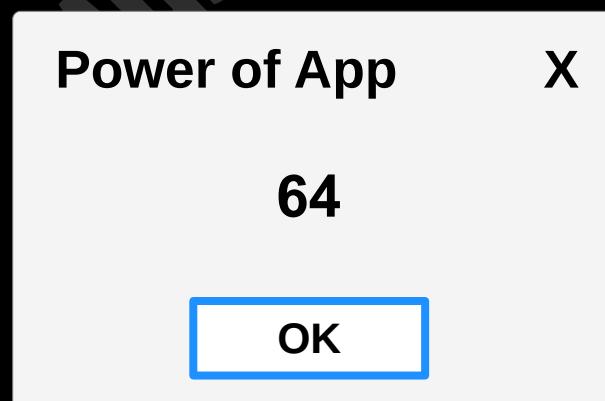
```
ourText = pow(8, 2)
```

```
ourTitle = "Power of App"
```

```
ourText = str(ourText) ← Convert to String
```

```
ctypes.windll.user32.MessageBoxW(0,  
ourText, ourTitle, 0)
```

pow means to RAISE TO THE POWER OF N



Exponents

**

Christopher Topalian

```
import ctypes
```

```
ourText = 8**2
```

```
ourTitle = "Power of App"
```

```
ourText = str(ourText) ← Convert to String
```

```
ctypes.windll.user32.MessageBoxW(0,  
ourText, ourTitle, 0)
```

Two ** symbols are another way
to write exponents.

Power of App

X

64

OK

math.sqrt

```
import ctypes, math
```

```
ourText = math.sqrt(4)
```

```
ourTitle = "Square Root App"
```

```
ourText = str(ourText) ← Convert to String
```

```
ctypes.windll.user32.MessageBoxW(0,  
ourText, ourTitle, 0)
```

comma
to import
multiple
modules

Convert to String

sqrt means SQUARE ROOT

Square Root App X

2.0

OK

CONVERT kmh to mph

```
import ctypes

kmh = int(input("Enter Kilometers Per Hour"))

mph = kmh * 0.6214

print(kmh, "Kilometers Per Hour is",
      mph,"Miles Per Hour")

ourText = str(mph) + " Miles Per Hour"
ourTitle = "Kilometers per hour to mph"

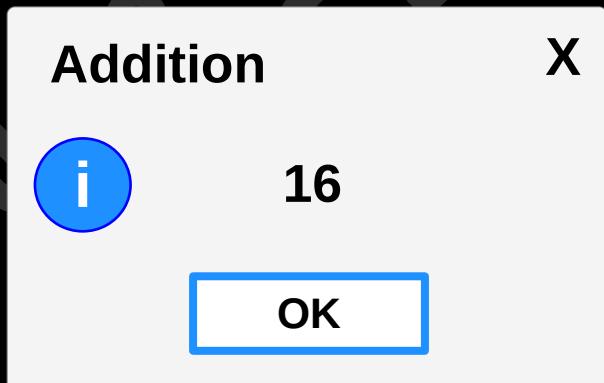
ctypes.windll.user32.MessageBoxW(0,
ourText, ourTitle, 0)

input('Press Enter to Exit')
```

For when the user Double Clicks to open:
The last line of code, `input('Press Enter to Exit')`
keeps the python launcher window open,
so that the user can view the result of the choice
that they made.

Make a Function for Addition

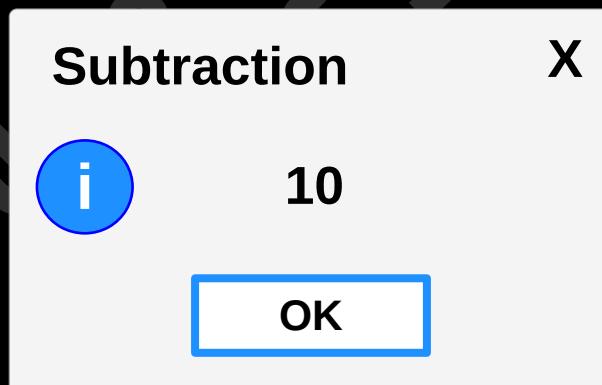
```
import ctypes  
  
def ourFunction(a, b):  
    c = a + b  
    return c  
  
answer = str(ourFunction(8,8))  
  
result = ctypes.windll.user32.MessageBoxW(0,  
answer,"Addition", 0)
```



As you can see above, the messagebox appears, with an info icon, and shows the result of the addition of the two numbers $8 + 8$, which equals 16, as the result, shown as the message.

Make a Function for Subtraction

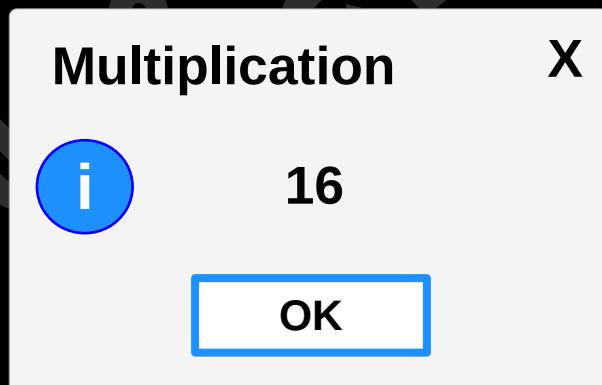
```
import ctypes  
  
def ourFunction(a, b):  
    c = a - b  
    return c  
  
answer = str(ourFunction(30,20))  
  
result = ctypes.windll.user32.MessageBoxW(0,  
answer,"Subtraction", 0)
```



As you can see above, the messagebox appears, with an info icon, and shows the result of the subtraction of the two numbers $30 - 20$, which equals 10, as the result, shown as the message.

Make a Function for Multiplication

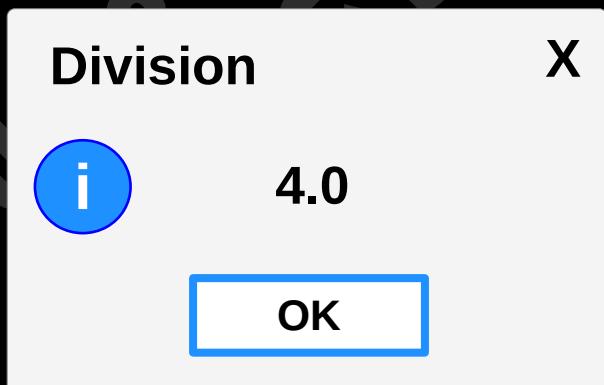
```
import ctypes  
  
def ourFunction(a, b):  
    c = a * b  
    return c  
  
answer = str(ourFunction(4,4))  
  
result = ctypes.windll.user32.MessageBoxW(0,  
answer,"Multiplication", 0)
```



As you can see above, the messagebox appears, with an info icon, and shows the result of the multiplication of the two numbers $4 * 4$, which equals 16, as the result, shown as the message.

Make a Function for Division

```
import ctypes  
  
def ourFunction(a, b):  
    c = a / b  
    return c  
  
answer = str(ourFunction(16,4))  
  
result = ctypes.windll.user32.MessageBoxW(0,  
answer,"Division", 0)
```



As you can see above, the messagebox appears, with an info icon, and shows the result of the division of the two numbers 16 / 4, which equals 4.0, as the result, shown as the message.

MAKE AN APP WINDOW

New Blank Window

```
from tkinter import *
```

```
ourWindow = Tk()
```

```
ourWindow.geometry('500x300+300+200')
```

```
ourWindow.title('A Simple Window')
```

```
ourWindow.mainloop()
```

Make Window Object

height

width

y pos

x pos

Run Event Loop

We can make a window and populate it with graphical user interface elements, such as **labels** and **buttons**, and we can fill our window with **text**, **images**, and even **videos**.

geometry specifies the window **size** and **location** on the screen.

mainloop() handles all events.

New Window, Button Changes Color

```
from tkinter import *
def ourFunction():
    print('Blue')
    ourWindow.configure(bg='blue')
ourWindow = Tk()
ourWindow.geometry('300x200+300+200')
ourWindow.title('Here is Our Window')
ourButton = Button(ourWindow,
text = 'Change Color',
command = ourFunction,
fg = 'white', bg = 'black').place(x = 100, y = 50)
ourWindow.mainloop()
```

This script makes a **Window**, with a **Button** that changes the Background **Color** of the Window.

New Window, font size, font type

```
from tkinter import *
```

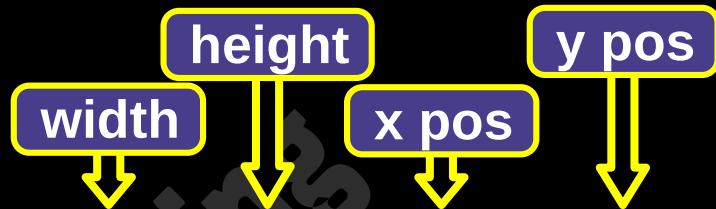
```
ourWindow = Tk()
```

```
ourWindow.geometry('700x400+300+150')
```

```
ourWindow.title('Our Window')
```

```
ourLabel = Label(ourWindow,  
text = 'This is Fun', width = '200',  
font=("Courier", 25), fg = 'aqua',  
bg = 'black').pack()
```

```
ourWindow.mainloop()
```

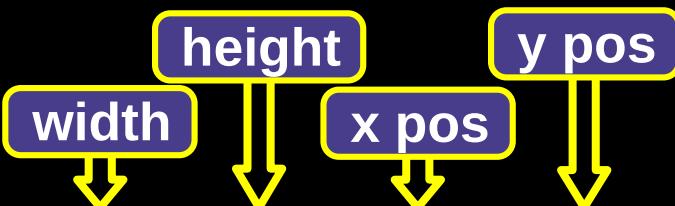


We make a Window with a Label width of 200, with a black background, aqua font color and Courier font face that is 25 for the font size.

New Window, font size/type variable

```
from tkinter import *
```

```
ourWindow = Tk()
```



```
ourWindow.geometry('500x400+300+150')
```

```
ourWindow.title('Our New Window')
```

```
ourText = 'Hi Everyone'
```

```
ourFontSize = 22
```

```
ourFontStyle = "Arial"
```

```
ourLabel = Label(ourWindow,
```

```
text = ourText, width = '10',
```

```
font=(ourFontStyle, ourFontSize),
```

```
fg = 'aqua', bg = 'black').pack(side = LEFT)
```

```
ourWindow.mainloop()
```

We make a Window with a Label width of 10, a black background, aqua font color, Arial font face, 22 for the font size, placed with pack LEFT, places object on left side of window.

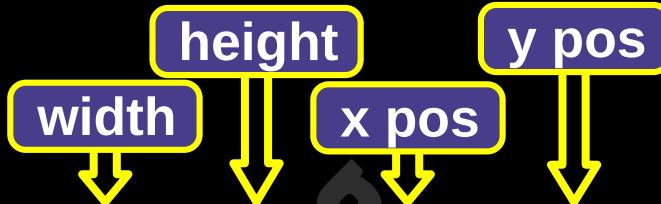
New Window Labels Placed

```
from tkinter import *
```

```
ourWindow = Tk()
```

```
ourWindow.geometry('400x200+200+100')
```

```
ourWindow.title('Greeter')
```



```
ourFirstLabel = Label(ourWindow,
```

```
text = 'Hi There',
```

```
fg = 'aqua', bg = 'black').place(x = 75, y = 75)
```

```
ourSecondLabel = Label(ourWindow,
```

```
text = 'Happy Scripting',
```

```
fg = 'aqua', bg = 'black').place(x = 75, y = 115)
```

```
ourWindow.mainloop()
```

We make a window, with a title of Greeter, and style two labels, that are placed in our window at specified locations of x and y.

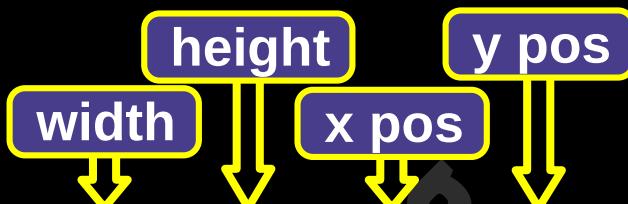
New Window Labels Grid Sticky

```
from tkinter import *
```

```
ourWindow = Tk()
```

```
ourWindow.geometry('400x200+200+100')
```

```
ourWindow.title('Greeter')
```



```
ourLabel1 = Label(ourWindow,
```

```
text = 'Hi Everyone', fg = 'aqua',
```

```
bg = 'black').grid(row = 0, column = 0, sticky = W)
```

```
ourLabel2 = Label(ourWindow,
```

```
text = 'Happy Scripting', fg = 'black',
```

```
bg = 'aqua').grid(row = 0, column = 1, sticky = W)
```

```
ourWindow.mainloop()
```

We make a window, with a title of Greeter, and style two labels, that are placed in our window using `grid` function and sticky options.

New Window Labels Button

```
from tkinter import *
def ourNewLabel():
    ourLabel3 = Label(ourWindow, text = 'Hi Everyone',
                      fg = 'blue').grid(row = 3, column = 0, sticky = W)
    return
ourWindow = Tk()
ourWindow.geometry('400x200+200+100')
ourWindow.title('Our Window')
ourLabel1 = Label(ourWindow,
                  text = 'Howdy', fg = 'aqua',
                  bg = 'black').grid(row = 0, column = 0, sticky = W)
ourLabel2 = Label(ourWindow,
                  text = 'How are you?', fg = 'black',
                  bg = 'aqua').grid(row = 0, column = 1, sticky = W)
ourButton1 = Button(ourWindow,
                    text = 'Click Here', command = ourNewLabel,
                    fg = 'white', bg = 'black').grid(row = 1, column = 0,
                                                    sticky = W)
ourWindow.mainloop()
```

New Window, while, counts 5 Rows

```
import time
from tkinter import *
```

```
ourWindow = Tk()
```

```
ourWindow.geometry('275x210+200+150')
```

```
ourWindow.title('Greeting App')
```

```
ourTitle = "Visitor Counter"
```

```
ourText = "People"
```

```
counter = 0
```

```
x = 1
```

```
while x <= 5:
```

```
    counter += 1
```

```
    ourMessage = str(counter) + ' ' + ourText
```

```
    x = x + 1;
```

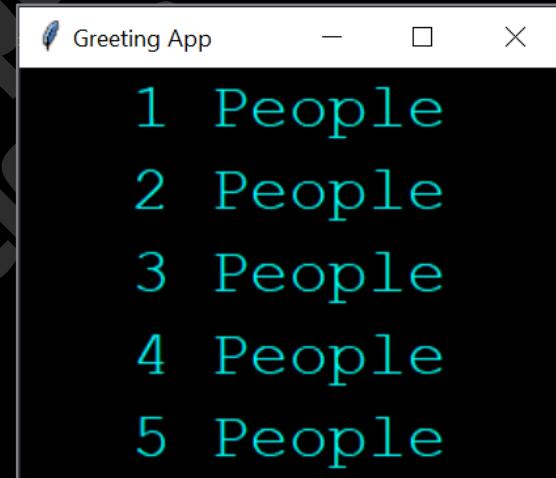
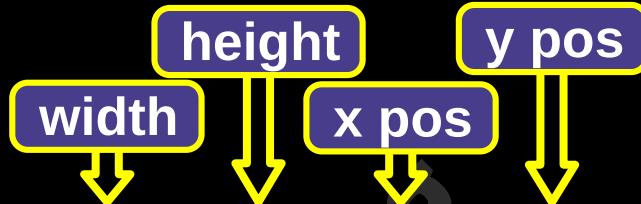
```
ourLabel = Label(ourWindow,
```

```
text = ourMessage, width = '200',
```

```
font=("Courier", 25), fg = 'aqua',
```

```
bg = 'black').pack()
```

```
ourWindow.mainloop()
```



.....
Next page
shows a
slight
variation

New Window, while, counts 5 Rows

```
import time
from tkinter import *
ourWindow = Tk()
ourWindow.geometry('275x125+200+150')
ourWindow.title('Greeting App')
```

```
ourTitle = "Visitor Counter"
ourText = "People"
```

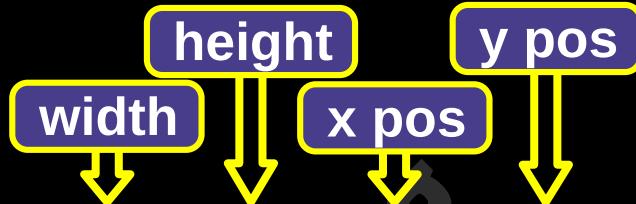
```
x = 1
```

```
while x <= 3:
```

```
    ourMessage = str(x) + ' ' + ourText
    x = x + 1;
```

```
    ourLabel = Label(ourWindow,
text = ourMessage, width = '200',
font=("Courier", 25), fg = 'aqua',
bg = 'black').pack()
```

```
ourWindow.mainloop()
```



x is used
as the
counting
variable

This script is a slight variation of the previous page.

New Window, Button Counts Up

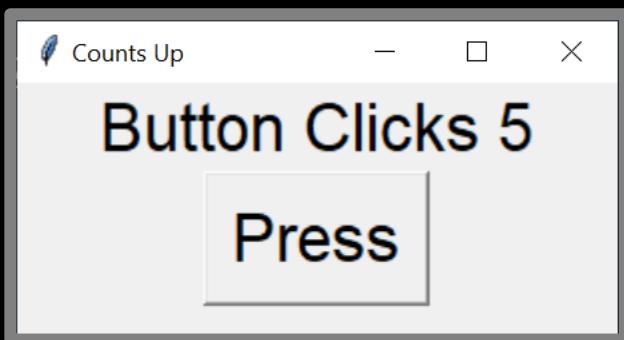
```
from tkinter import*
count = 0
def ourCounter():
    global count
    count = count + 1
    ourLabel1.configure(text=f'Button Clicks {count}')

ourWindow = Tk()
ourWindow.geometry('300x125+200+150')
ourWindow.title("Counts Up")

ourLabel1 = Label(ourWindow, font=("Arial", 25))
ourLabel1.pack()

ourButton = Button(ourWindow, text="Press",
                   command=ourCounter, font=("Arial", 25))
ourButton.pack()

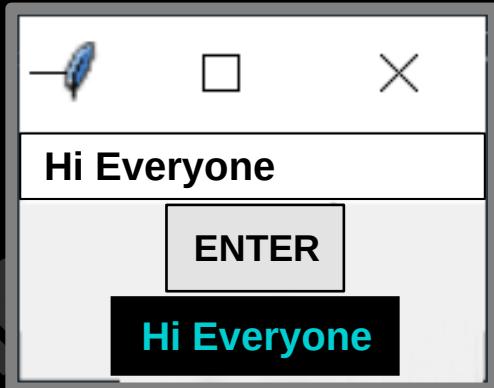
ourWindow.mainloop()
```



Text Field for User Input

Text Input Field - Entry, pack

```
from tkinter import *
ourWindow = Tk()
ourTextField1 = Entry(ourWindow)
ourTextField1.pack()
def ourFunction():
    textEntered = ourTextField1.get()
    ourLabel = Label(ourWindow,
                     text = textEntered, fg = 'aqua', bg = 'black').pack()
    ourButton = Button(ourWindow, text = "ENTER",
                       command = ourFunction).pack()
ourWindow.mainloop()
```

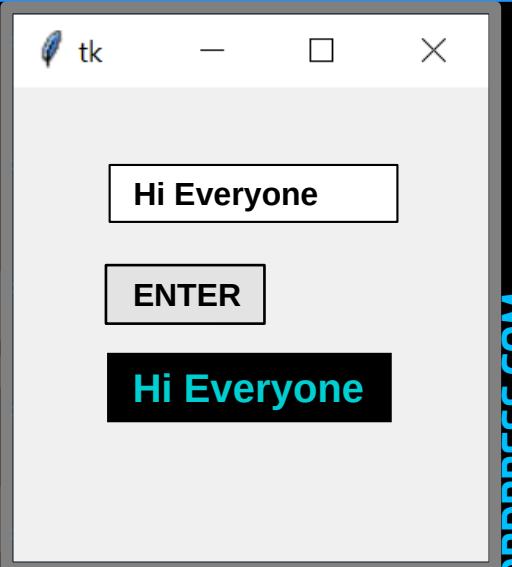


Entry is how we make Text Input Boxes for users to type in their information.

In this example, we place the Text Boxes using the pack() method, which positions elements in relation to one another.

Text Input Field - Entry, place

```
from tkinter import *
ourWindow = Tk()
ourTextField1 = Entry(ourWindow)
ourTextField1.place(x=40, y=35)
def ourFunction():
    textEntered = ourTextField1.get()
    ourLabel = Label(ourWindow,
                     text = textEntered, fg = 'aqua', bg = 'black')
    ourLabel.place(x = 50, y = 115)
    ourLabel.width=200
    ourLabel.height=10
    ourButton = Button(ourWindow, text = "ENTER",
                       command = ourFunction)
    ourButton.width=15
    ourButton.height=10
    ourButton.place(x=50,y=75)
ourWindow.mainloop()
```



Text Input Field - Entry - bg Color

Christopher Topalian

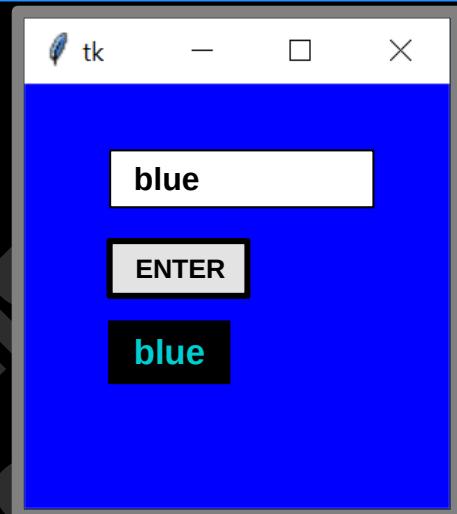
```
from tkinter import *
ourWindow = Tk()
ourTextField1 = Entry(ourWindow)
ourTextField1.place(x=40, y=35)

def ourFunction():
    textEntered = ourTextField1.get()
    if(textEntered == 'blue'):
        ourWindow.configure(bg='blue')
    elif(textEntered=='tan'):
        ourWindow.configure(bg='tan')

    ourLabel = Label(ourWindow,
    text = textEntered, fg = 'aqua', bg = 'black')
    ourLabel.place(x = 50, y = 115)
    ourLabel.width=200
    ourLabel.height=10

    ourButton = Button(ourWindow, text = "ENTER",
    command = ourFunction)
    ourButton.width=15
    ourButton.height=10
    ourButton.place(x=50,y=75)

    ourWindow.mainloop()
```



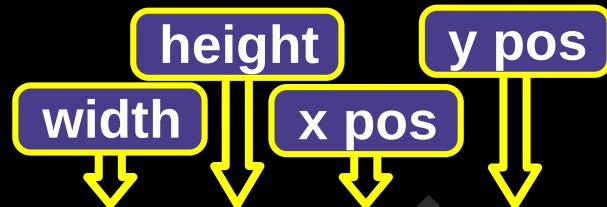
Textbox Updates Label by Typing

```
from tkinter import *
```

```
ourWindow = Tk()
```

```
ourWindow.geometry('275x75+200+150')
```

```
ourWindow.title('Textbox to Label')
```



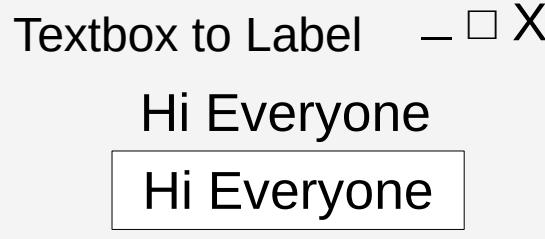
```
ourVariable = StringVar()
```

```
ourVariable.set('Hi Everyone')
```

```
ourLabel1 = Label(ourWindow,  
textvariable = ourVariable)  
ourLabel1.pack()
```

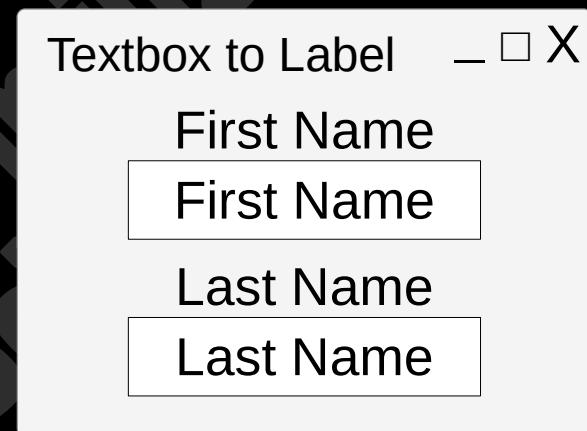
```
ourTextArea = Entry(ourWindow,  
textvariable = ourVariable)  
ourTextArea.pack()
```

```
ourWindow.mainloop()
```



Textbox - Two Updated Labels, Typing

```
from tkinter import *
ourWindow = Tk()
ourWindow.geometry('275x100+200+150')
ourWindow.title('Textbox to Label')
ourVariable1 = StringVar()
ourVariable1.set('First Name')
ourLabel1 = Label(ourWindow,
textvariable = ourVariable1)
ourLabel1.pack()
ourTextArea = Entry(ourWindow,
textvariable = ourVariable1)
ourTextArea.pack()
#-----
ourVariable2 = StringVar()
ourVariable2.set('Last Name')
ourLabel2 = Label(ourWindow,
textvariable = ourVariable2)
ourLabel2.pack()
ourTextArea2 = Entry(ourWindow,
textvariable = ourVariable2)
ourTextArea2.pack()
ourWindow.mainloop()
```

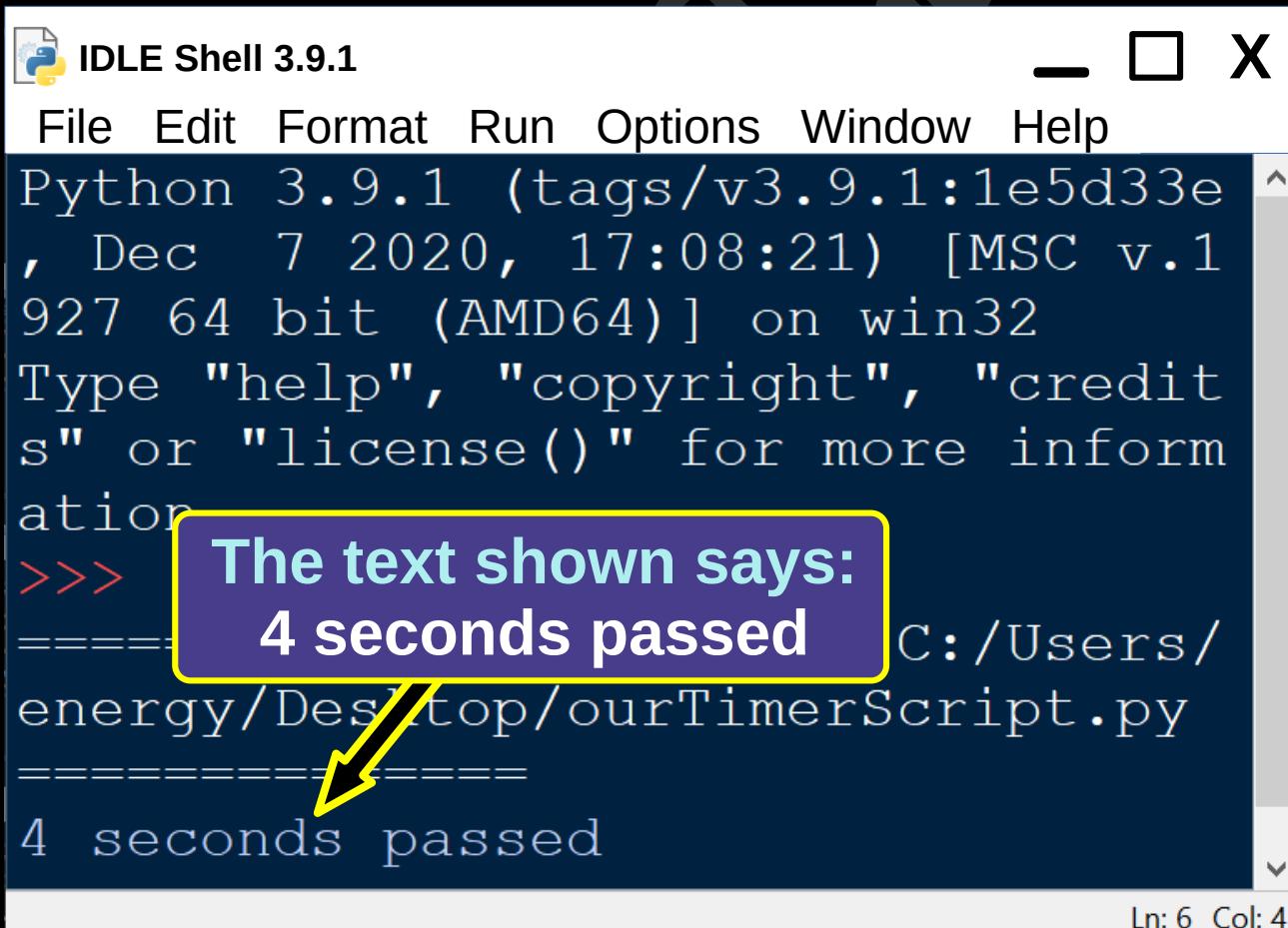


timers

College of Scripting
Music & Science

Timer - Activates One Time

```
import time  
time.sleep(4.0)  
print('4 seconds passed')  
input('Press Enter to Exit')
```



The screenshot shows the IDLE Shell interface with the title "IDLE Shell 3.9.1". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The window displays Python version information and a help message. A yellow callout box highlights the text "The text shown says: 4 seconds passed" from the command-line output. A yellow arrow points from the bottom of the callout box to the word "passed" in the output. The output text is:
>>> ===== The text shown says:
===== 4 seconds passed C:/Users/
energy/Desktop/ourTimerScript.py
===== 4 seconds passed

Timer - Activates One Time

```
import time, ctypes
```

```
ourTitle = "Greeting Timer"
```

```
ourText = "Hi Everyone"
```

```
time.sleep(4.0)
```

```
ctypes.windll.user32.MessageBoxW(0,  
ourText, ourTitle, 0)
```

Greeting Timer X

Hi Everyone

OK

sleep function is found in the **time** module, which is why we import **time**

In this example, our Message Box is created ONLY once, after a 4 second sleep.

On the next page, we make a repeating timer, to keep activating, a specified number of times.

Timer - Keep Activating, while

```
import time, ctypes
```

```
ourTitle = "Greeting Timer"
```

```
ourText = "Hi Everyone"
```

```
x = 1
```

```
while x <= 5:
```

```
    time.sleep(2.0)
```

```
    x = x + 1;
```

```
ctypes.windll.user32.MessageBoxW(0,  
ourText, ourTitle, 0)
```

Greeting Timer X

Hi Everyone

OK

Our Message Box is created, and after the user presses OK, 2 seconds passes, and another Message Box is created, with a max number of 5 Message Boxes allowed. The **while** loop is where we specify how many times we want our timer to activate.

Timer - Keep Activating, counter

```
import time, ctypes
```

```
ourTitle = "Visitor Counter"
```

```
ourText = "People"
```

```
counter = 0
```

```
x = 1
```

```
while x <= 5:
```

```
    time.sleep(2.0)
```

```
    counter += 1
```

```
    ourMessage = str(counter) + ' ' + ourText
```

```
    x = x + 1;
```

```
    ctypes.windll.user32.MessageBoxW(0,  
        ourMessage, ourTitle, 0)
```

Visitor Counter X

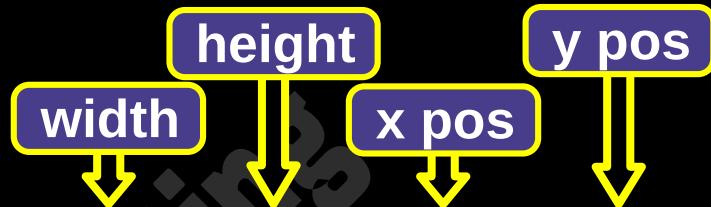
5 People

OK

Our Message Box, with a counter, counts to 5

timer, Window, while, font styling

```
import time  
from tkinter import *\n\nourWindow = Tk()  
ourWindow.geometry('300x200+200+150')  
ourWindow.title('Greeting App')\n\ntime.sleep(60.0)\n\nourLabel = Label(ourWindow,  
text = 'This is Fun', width = '200',  
font=("Courier", 25), fg = 'aqua',  
bg = 'black').pack()\n\nourWindow.mainloop()
```



This Script is a 1 Minute Timer, that will open your Application Window, 60 seconds after you choose to Run it.

Clock

College of Scripting
Music & Science

Clock Updates in Seconds

```
from tkinter import *
```

```
from time import*
```

```
ourWindow = Tk()
```

```
ourWindow.geometry("200x50+60+600")
```

```
ourWindow.title("Clock Updating")
```

```
def ourClock():
```

```
    formatted = strftime("%H:%M:%S %p")
```

```
    currentTime.config(text = formatted)
```

```
    ourWindow.after(1000, timeout)
```

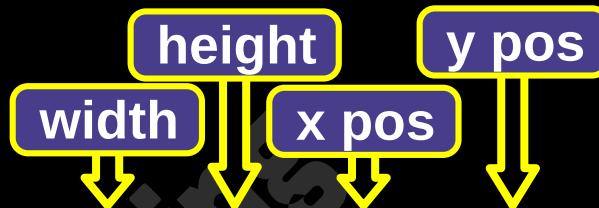
```
currentTime= Label(text=",
```

```
bg="black",fg="white", font=("Arial", 25))
```

```
currentTime.pack()
```

```
ourClock()
```

```
ourWindow.mainloop()
```



Clock Updates in Seconds

```
from tkinter import *
from time import*
ourWindow = Tk()
ourWindow.geometry('300x75+200+150')
ourWindow.title('Updating Clock')

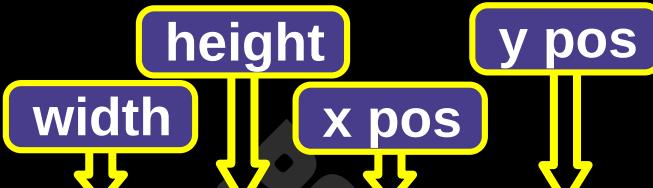
def ourClock():
    formatted = strftime('%H:%M:%S %p')
    ourWindow.config(text = formatted)
    ourWindow.after(1000, ourClock)

ourWindow = Label(ourWindow,
font = ('arial', 35, 'bold'),
bg = 'black', fg = 'white')

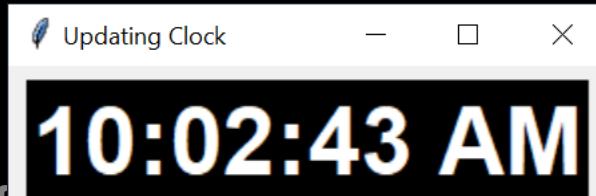
ourWindow.pack(fill="none", expand=True)

ourClock()

ourWindow.mainloop()
```



The diagram illustrates the geometry parameters for a window. It shows four boxes with arrows pointing downwards: 'width' (purple), 'height' (yellow), 'x pos' (purple), and 'y pos' (yellow). These parameters are used in the `geometry` method of the window object.



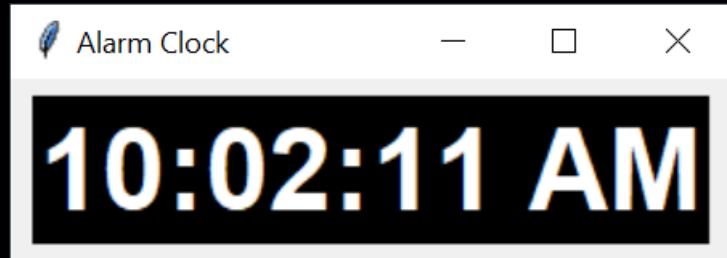
A screenshot of a window titled "Updating Clock". The window contains a single label displaying the time "10:02:43 AM" in a large, bold, black font against a white background.

Clock – Alarm Clock - Beep

Christopher Topalian

```
from tkinter import *
from time import*
import winsound
ourWindow = Tk()
ourWindow.geometry('300x75+200+150')
ourWindow.title('Alarm Clock')

def ourClock():
    formatted = strftime('%H:%M:%S %p')
    ourWindow.config(text = formatted)
    ourWindow.after(1000, ourClock)
    if(formatted > '13:44:10PM'):
        print('Alarm Activated')
        winsound.Beep(1000,500)
        ourWindow.configure(bg='blue')
    ourWindow = Label(ourWindow,
font = ('arial', 35, 'bold'),
bg = 'black', fg = 'white')
    ourWindow.pack(fill="none", expand=True)
ourClock()
ourWindow.mainloop()
```



Pitch is
0 to 2500 Hz
Length is in
milli seconds

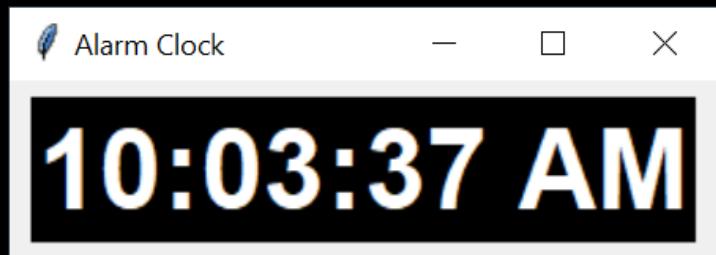
Clock – Alarm Clock - PlaySound

Christopher Topalish

```
from tkinter import *
from time import*
import winsound
ourWindow = Tk()
ourWindow.geometry('300x75+200+150')
ourWindow.title('Alarm Clock')

def ourClock():
    formatted = strftime('%H:%M:%S %p')
    ourWindow.config(text = formatted)
    ourWindow.after(1000, ourClock)
    if(formatted > '13:44:10PM'):
        print('Alarm Activated')
        winsound.PlaySound('MB_ICONASTERISK', True)
        ourWindow.configure(bg='blue')

ourWindow = Label(ourWindow,
font = ('arial', 35, 'bold'),
bg = 'black', fg = 'white')
ourWindow.pack(fill="none", expand=True)
ourClock()
ourWindow.mainloop()
```



Clock – Alarm Clock - PlaySound

Christopher Topalish

```
from tkinter import *
from time import*
import winsound
ourWindow = Tk()
ourWindow.geometry('725x75+200+150')
ourWindow.title('Alarm Clock')

def ourClock():
    theTime = localtime()
    formatted = asctime(theTime)
    ourWindow.config(text = formatted)
    ourWindow.after(1000, ourClock)
    if(formatted > '13:44:10PM'):
        print('Alarm Activated')
        winsound.PlaySound('MB_ICONASTERISK', True)
        ourWindow.configure(bg='blue')

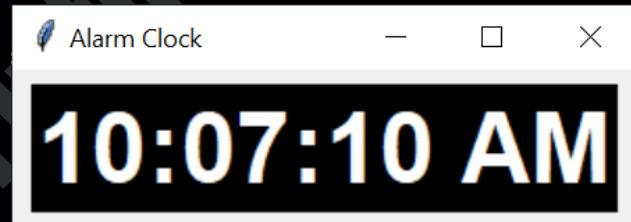
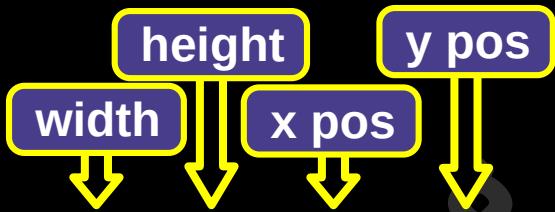
ourWindow = Label(ourWindow,
font = ('arial', 35, 'bold'),
bg = 'black', fg = 'white')
ourWindow.pack(fill="none", expand=True)
ourClock()
ourWindow.mainloop()
```



Alarm Clock – with Start and End

Christopher Lopatich

```
from tkinter import *
from time import*
import winsound
ourWindow = Tk()
ourWindow.geometry('300x75+200+150')
ourWindow.title('Alarm Clock')
start = '19:50:00PM'
end = '19:50:10PM'
def ourClock():
    formatted = strftime('%H:%M:%S %p')
    ourWindow.config(text = formatted)
    ourWindow.after(1000, ourClock)
    if(formatted > start and formatted < end):
        print('Alarm Activated')
        winsound.PlaySound('MB_ICONASTERISK', True)
        ourWindow.configure(bg='blue')
ourWindow = Label(ourWindow,
font = ('arial', 35, 'bold'),
bg = 'black', fg = 'white')
ourWindow.pack(fill="none", expand=True)
ourClock()
ourWindow.mainloop()
```



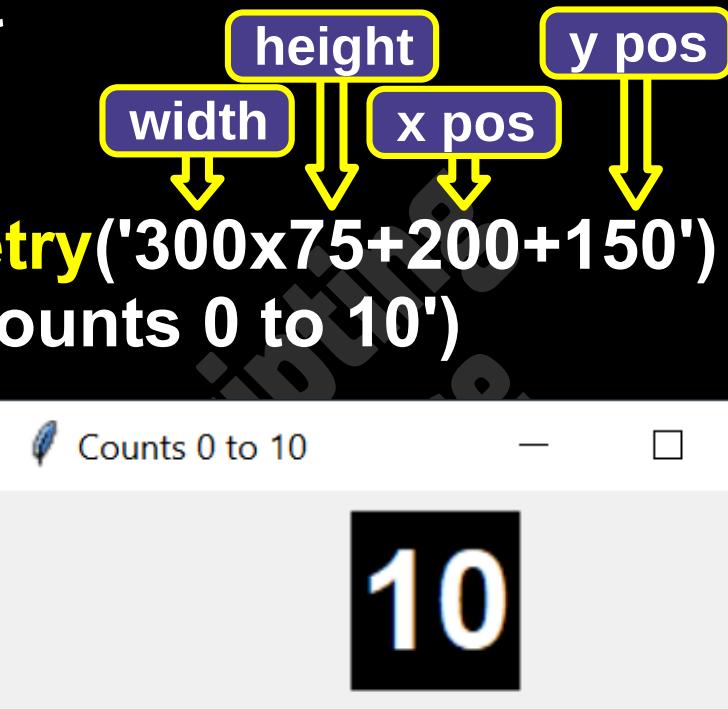
Counter

College of Scripting
Music & Science

Counts Up – 0 to 10

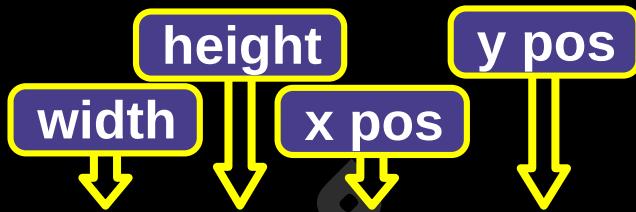
Christopher Topalian

```
from tkinter import *
from time import*
ourWindow = Tk()
ourWindow.geometry('300x75+200+150')
ourWindow.title('Counts 0 to 10')
count = 0
def ourCounter():
    global count
    if(count != 10):
        count += 1
        ourWindow.config(text = count)
        ourWindow.after(1000, ourCounter)
ourWindow = Label(ourWindow,
font = ('arial', 35, 'bold'),
bg = 'black', fg = 'white')
ourWindow.pack(fill="none", expand=True)
ourCounter()
ourWindow.mainloop()
```



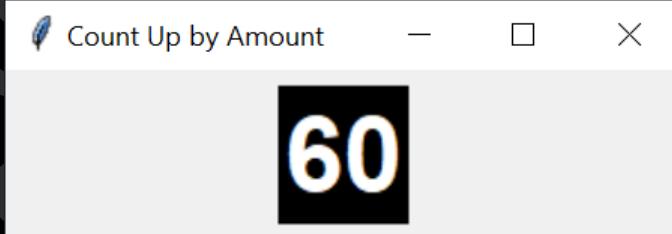
Counts Down – 10 to 0

```
from tkinter import *
from time import*
ourWindow = Tk()
ourWindow.geometry('300x75+200+150')
ourWindow.title('Counts 10 to 0')
count = 11
def ourCounter():
    global count
    if(count != 0):
        count -= 1
        ourWindow.config(text = count)
        ourWindow.after(1000, ourCounter)
ourWindow = Label(ourWindow,
font = ('arial', 35, 'bold'),
bg = 'black', fg = 'white')
ourWindow.pack(fill="none", expand=True)
ourCounter()
ourWindow.mainloop()
```



Counts Up – Specified Amount

```
from tkinter import *
from time import*
ourWindow = Tk()
ourWindow.geometry('300x75+200+150')
ourWindow.title('Count Up by Amount')
count = 0
def ourCounter(seconds):
    global count
    if(count != seconds):
        count += 1
        ourWindow.config(text = count)
        ourWindow.after(1000, ourCounter, seconds)
ourWindow = Label(ourWindow,
font = ('arial', 35, 'bold'),
bg = 'black', fg = 'white')
ourWindow.pack(fill="none", expand=True)
ourCounter(60)
ourWindow.mainloop()
```



The code above demonstrates how to create a window that counts up from 0 to a specified number (in this case, 60). It uses the Tkinter library to handle the window and the time library to manage the timer. The window is styled with a large, bold, black font on a black background.

Button

College of Scripting
Music & Science

Button Updates Label on Press

```
from tkinter import *
```

```
ourWindow = Tk()
```

```
ourWindow.geometry('275x100+200+150')
```

```
ourWindow.title('Button Updates Label')
```

```
def ourFunction():
```

```
    ourVariable1.set('howdy')
```

```
ourVariable1 = StringVar()
```

```
ourVariable1.set('Hi Everyone')
```

```
ourLabel1 = Label(ourWindow,
```

```
textvariable = ourVariable1)
```

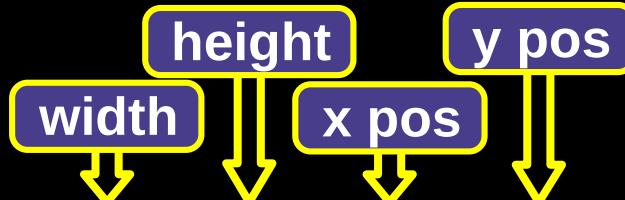
```
ourLabel1.pack()
```

```
ourButton = Button(ourWindow, text="Press",
```

```
command=ourFunction, font=("Arial", 25))
```

```
ourButton.pack()
```

```
ourWindow.mainloop()
```



Button Updates Label – □ X

Hi Everyone

Press

Button Updates Label – □ X

Howdy

Press

Button – Toggle Switch – On, Off

Christopher Topalian

```
from tkinter import *
ourWindow = Tk()
ourWindow.geometry('275x100+200+150')
ourWindow.title('Button Toggle Switch')
on = False
def ourFunction():
    global on
    if(on == False):
        ourVariable1.set('On')
        on = True
    elif(on == True):
        ourVariable1.set('Off')
        on = False
ourVariable1 = StringVar()
ourVariable1.set('Hi Everyone')
ourLabel1 = Label(ourWindow,
textvariable = ourVariable1)
ourLabel1.pack()
ourButton = Button(ourWindow, text="Press",
command=ourFunction, font=("Arial", 25))
ourButton.pack()
ourWindow.mainloop()
```

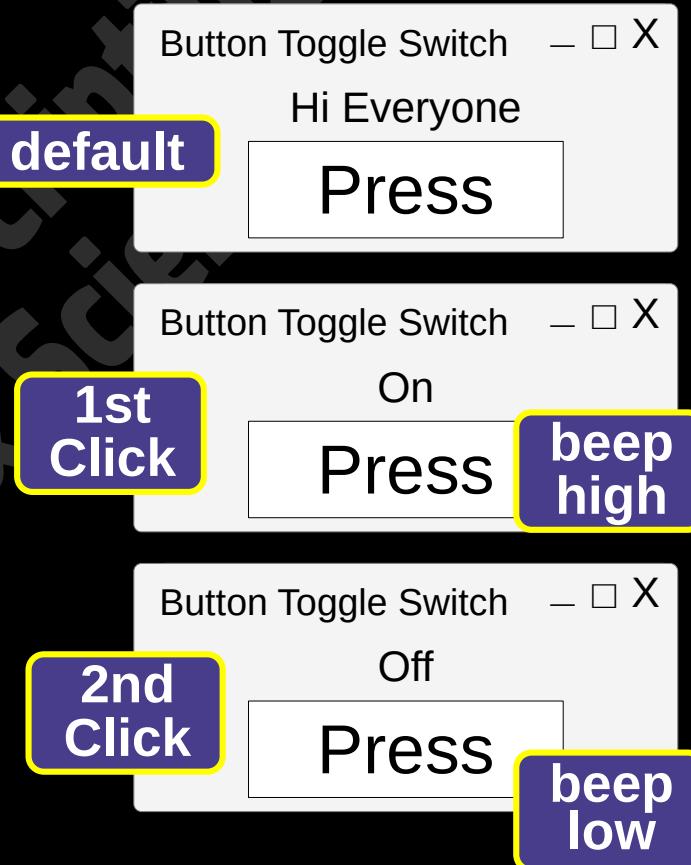


Button – Toggle Switch with Beeps

```

from tkinter import *
import winsound
ourWindow = Tk()
ourWindow.geometry('275x100+200+150')
ourWindow.title('Button Toggle Switch')
on = False
def ourFunction():
    global on
    if(on == False):
        ourVariable1.set('on')
        winsound.Beep(1000,100)
        on = True
    elif(on == True):
        ourVariable1.set('off')
        winsound.Beep(500,100)
        on = False
    ourVariable1 = StringVar()
    ourVariable1.set('Hi Everyone')
    ourLabel1 = Label(ourWindow,
textvariable = ourVariable1)
    ourLabel1.pack()
    ourButton = Button(ourWindow, text="Press",
command=ourFunction, font=("Arial", 25))
    ourButton.pack()
    ourWindow.mainloop()

```



Button Counts Up by 1

```
from tkinter import *
ourWindow = Tk()
ourWindow.geometry('275x100+200+150')
ourWindow.title('Button Counts Up by 1')

counter = 0
def countUp():
    global counter
    counter += 1
    ourVariable1.set('Number of Clicks ' + str(counter))

ourVariable1 = StringVar()
ourVariable1.set('Hi Everyone')
ourLabel1 = Label(ourWindow,
textvariable = ourVariable1)
ourLabel1.pack()

ourButton = Button(ourWindow, text="Press",
command=countUp, font=("Arial", 25))
ourButton.pack()

ourWindow.mainloop()
```

Button Counts Up by 1 – □ X

Number of Clicks 17

Press

Button Counts Down by 1

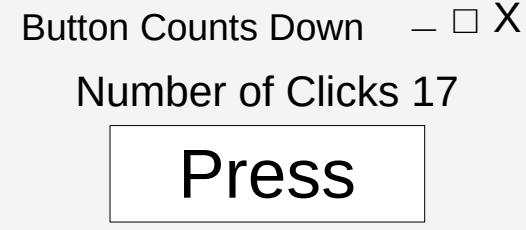
```
from tkinter import *
ourWindow = Tk()
ourWindow.geometry('275x100+200+150')
ourWindow.title('Button Counts Down')

counter = 100
def countDown():
    global counter
    counter -= 1
    ourVariable1.set('Number of Clicks ' + str(counter))

ourVariable1 = StringVar()
ourVariable1.set('Hi Everyone')
ourLabel1 = Label(ourWindow,
textvariable = ourVariable1)
ourLabel1.pack()

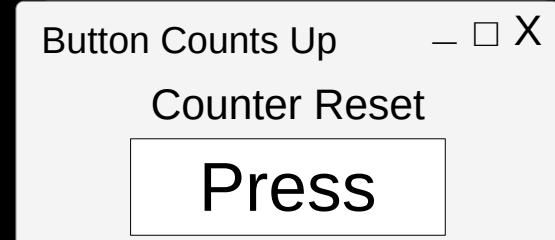
ourButton = Button(ourWindow, text="Press",
command=countDown, font=("Arial", 25))
ourButton.pack()

ourWindow.mainloop()
```



Button Counts Up to 10, resets

```
from tkinter import *
ourWindow = Tk()
ourWindow.geometry('275x100+200+150')
ourWindow.title('Button Counts Up')
counter = 0
def ourFunction():
    global counter
    counter += 1
    ourVariable1.set('Number of Clicks ' + str(counter))
    if(counter == 10):
        ourVariable1.set('Counter Reset')
        counter = 0
ourVariable1 = StringVar()
ourVariable1.set('Hi Everyone')
ourLabel1 = Label(ourWindow,
textvariable = ourVariable1)
ourLabel1.pack()
ourButton = Button(ourWindow, text="Press",
command=ourFunction, font=("Arial", 25))
ourButton.pack()
ourWindow.mainloop()
```



Christopher Topalian

Button Counts Down to 0, resets

```
from tkinter import *
ourWindow = Tk()
ourWindow.geometry('275x100+200+150')
ourWindow.title('Button Counts Down')

counter = 10
def ourFunction():
    global counter
    counter -= 1
    ourVariable1.set('Number of Clicks ' + str(counter))
    if(counter == 0):
        ourVariable1.set('Counter Reset')
        counter = 10

ourVariable1 = StringVar()
ourVariable1.set('Hi Everyone')
ourLabel1 = Label(ourWindow,
textvariable = ourVariable1)
ourLabel1.pack()

ourButton = Button(ourWindow, text="Press",
command=ourFunction, font=("Arial", 25))
ourButton.pack()

ourWindow.mainloop()
```

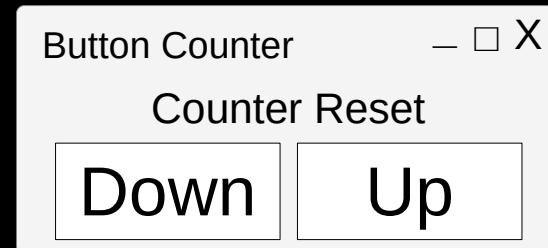


```

from tkinter import *
ourWindow = Tk()
ourWindow.geometry('275x100+200+150')
ourWindow.title('Button Counter')
counter = 0
def countUp():
    global counter
    counter += 1
    ourVariable1.set('Number of Clicks ' + str(counter))
def countDown():
    global counter
    counter -= 1
    ourVariable1.set('Number of Clicks ' + str(counter))
ourVariable1 = StringVar()
ourVariable1.set('Number of Clicks')
ourLabel1 = Label(ourWindow,
textvariable = ourVariable1)
ourLabel1.pack()
downButton = Button(ourWindow, text="Down",
command=countDown, font=("Arial", 25))
downButton.pack(side=LEFT)
upButton = Button(ourWindow, text="Up",
command=countUp, font=("Arial", 25))
upButton.pack(side=LEFT)
ourWindow.mainloop()

```

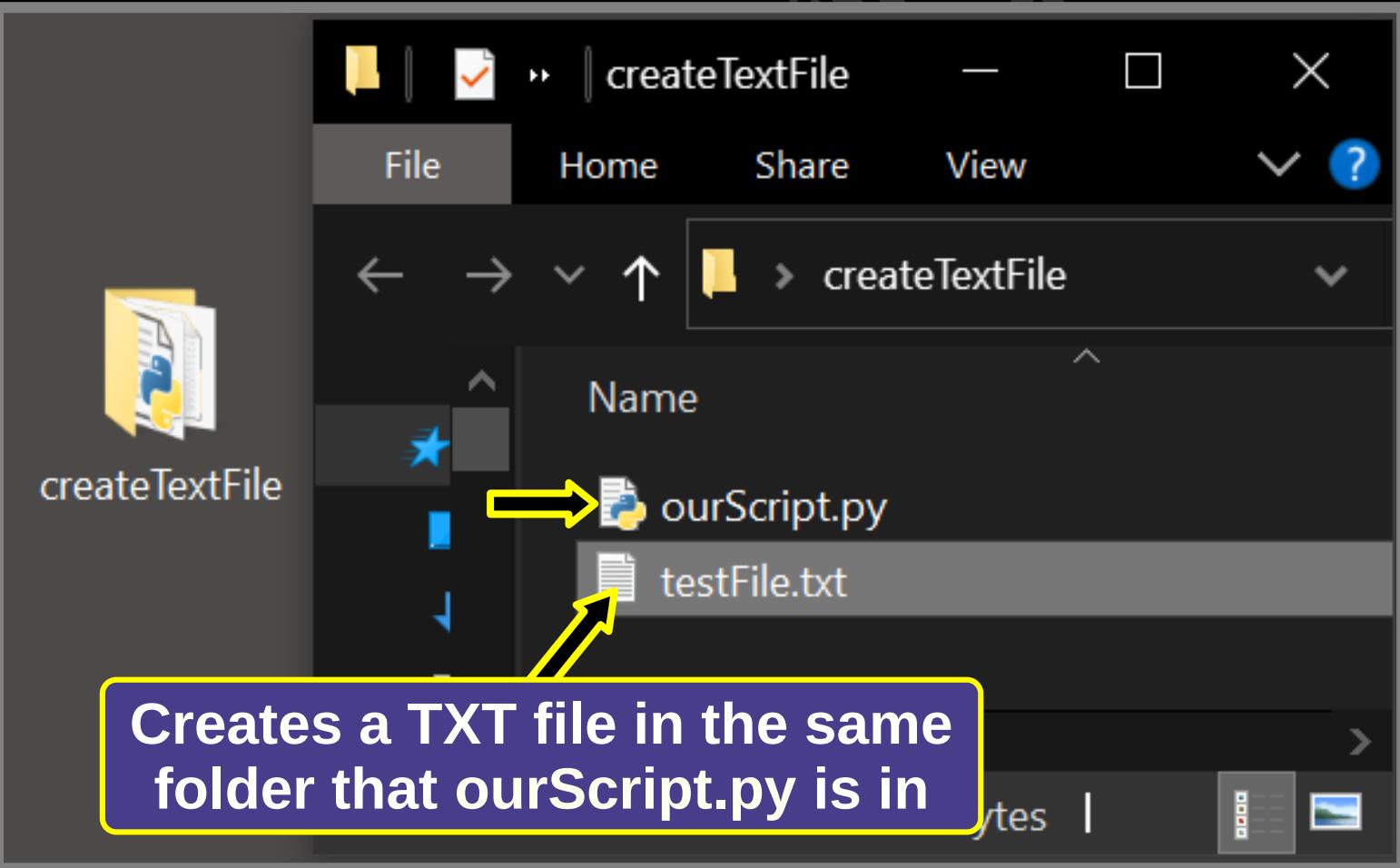
Button Count Up Count Down



College of Scripting Music & Science

Create a Text File

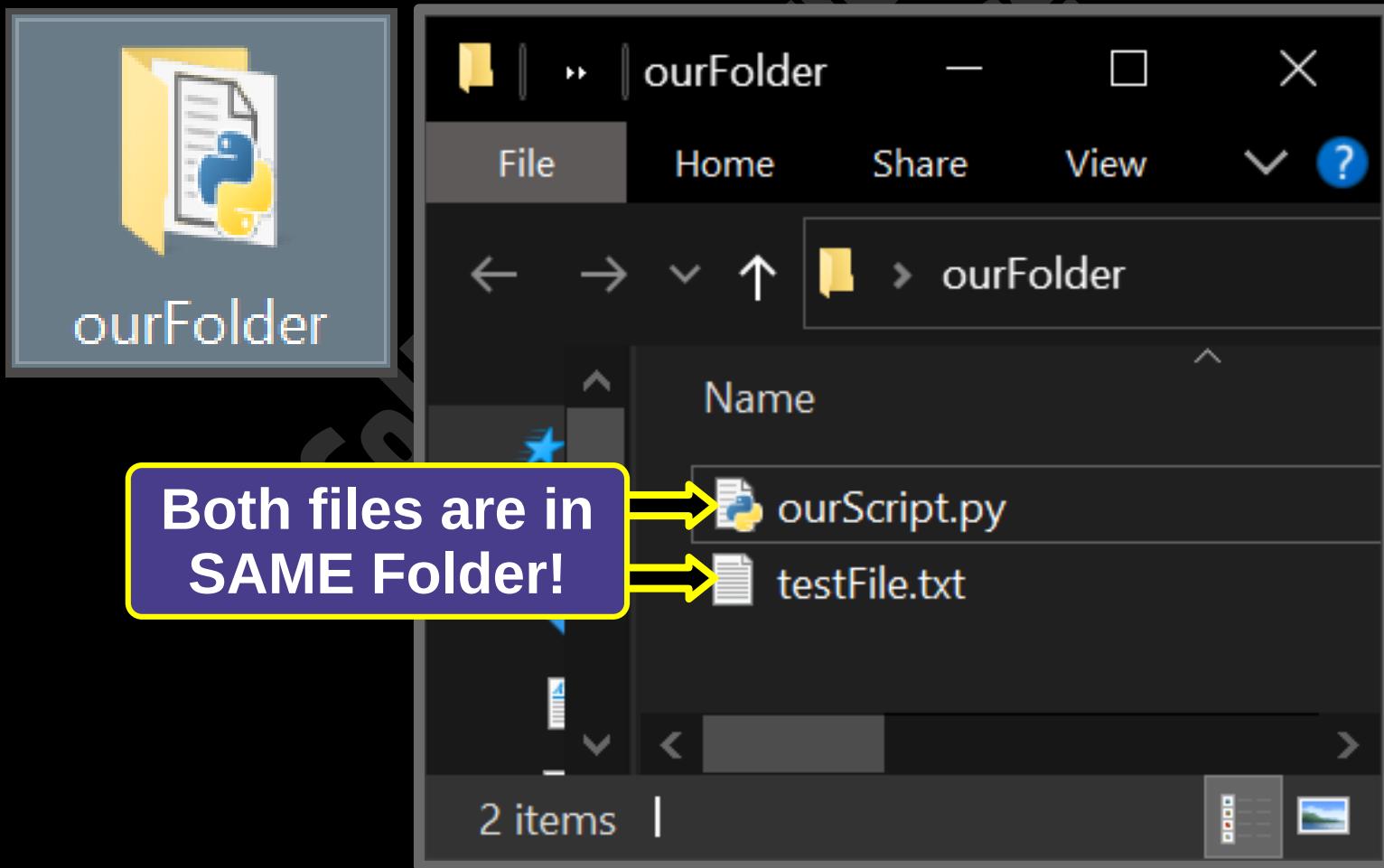
```
ourFile = open("testFile.txt", "w")
ourFile.write("Hi Everyone")
ourFile.close()
```



Open a File (in same folder)

```
import os
```

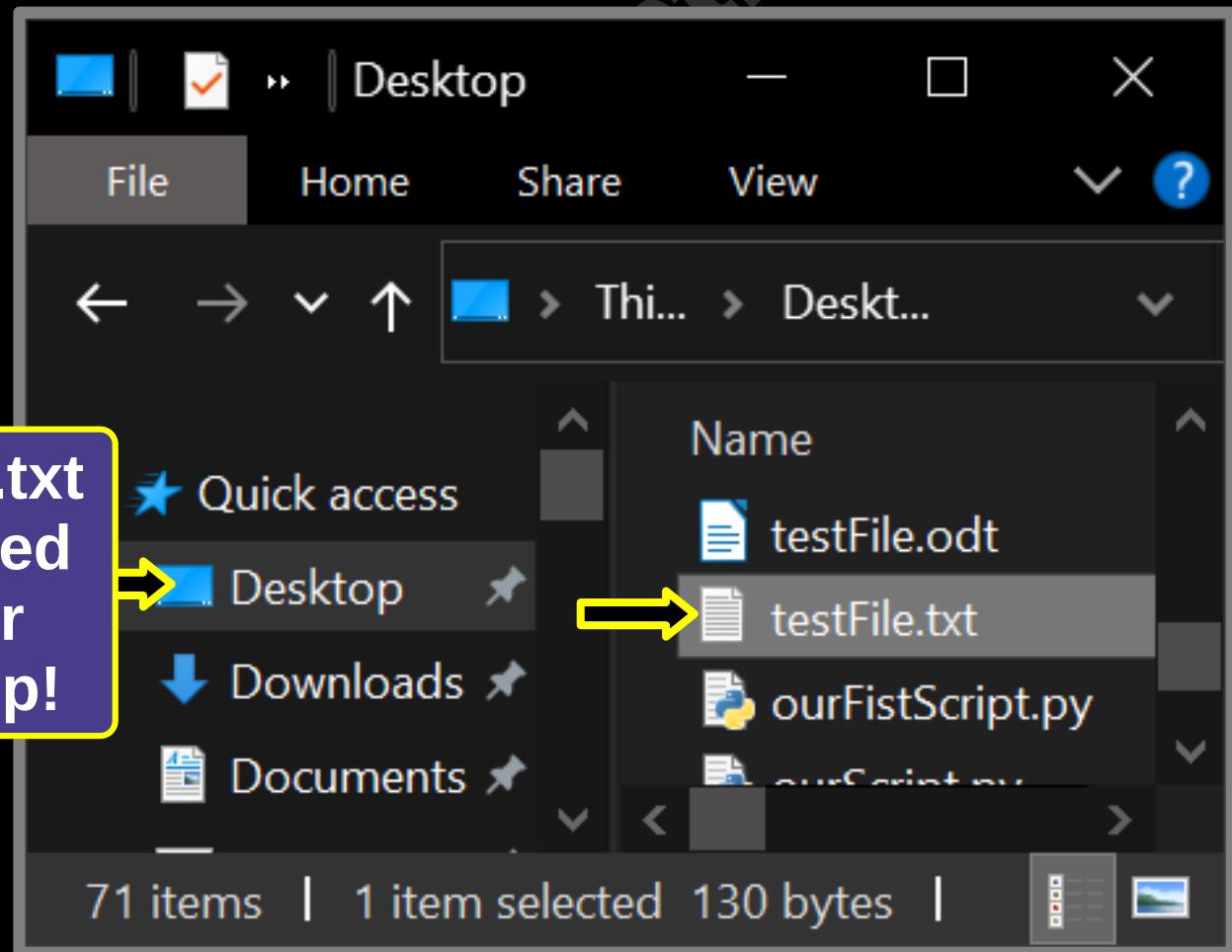
```
os.startfile('testFile.txt')
```



Open a File (from any Folder)

```
import os
```

```
os.startfile('C:/Users/you/Desktop/testFile.txt')
```



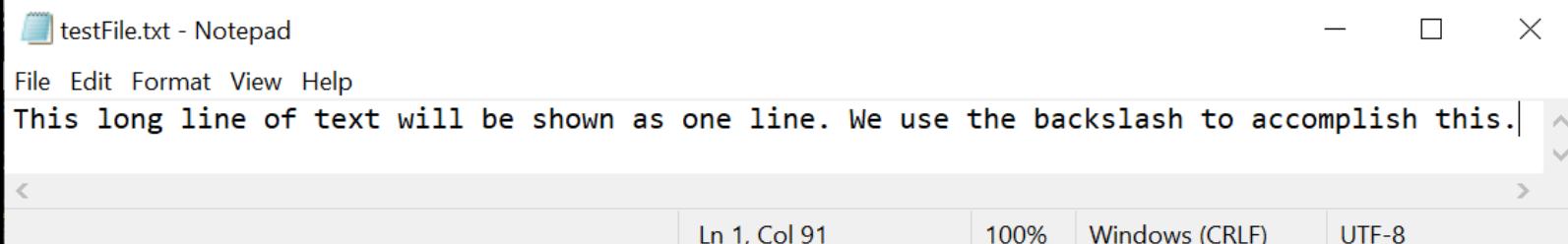
**testFile.txt
is located
on our
Desktop!**

Create a Text File - Long Line

**ourText = 'This long line of text \
will be shown as one line. \
We use the backslash \
to accomplish this.'**

```
ourFile = open('testFile.txt', 'w')  
ourFile.write(ourText)  
ourFile.close()
```

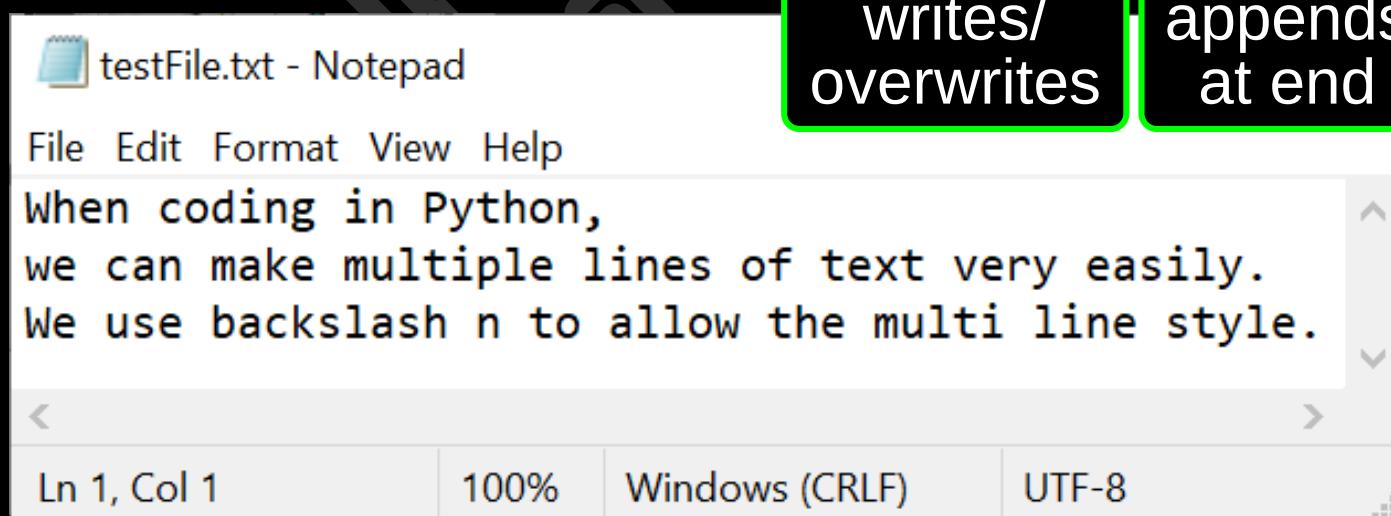
We write ONE long line of text into the text file.
If text file doesn't exist yet, it is created.
If text file already exists, w means overwrite it.



Create a Text File - Multi Line

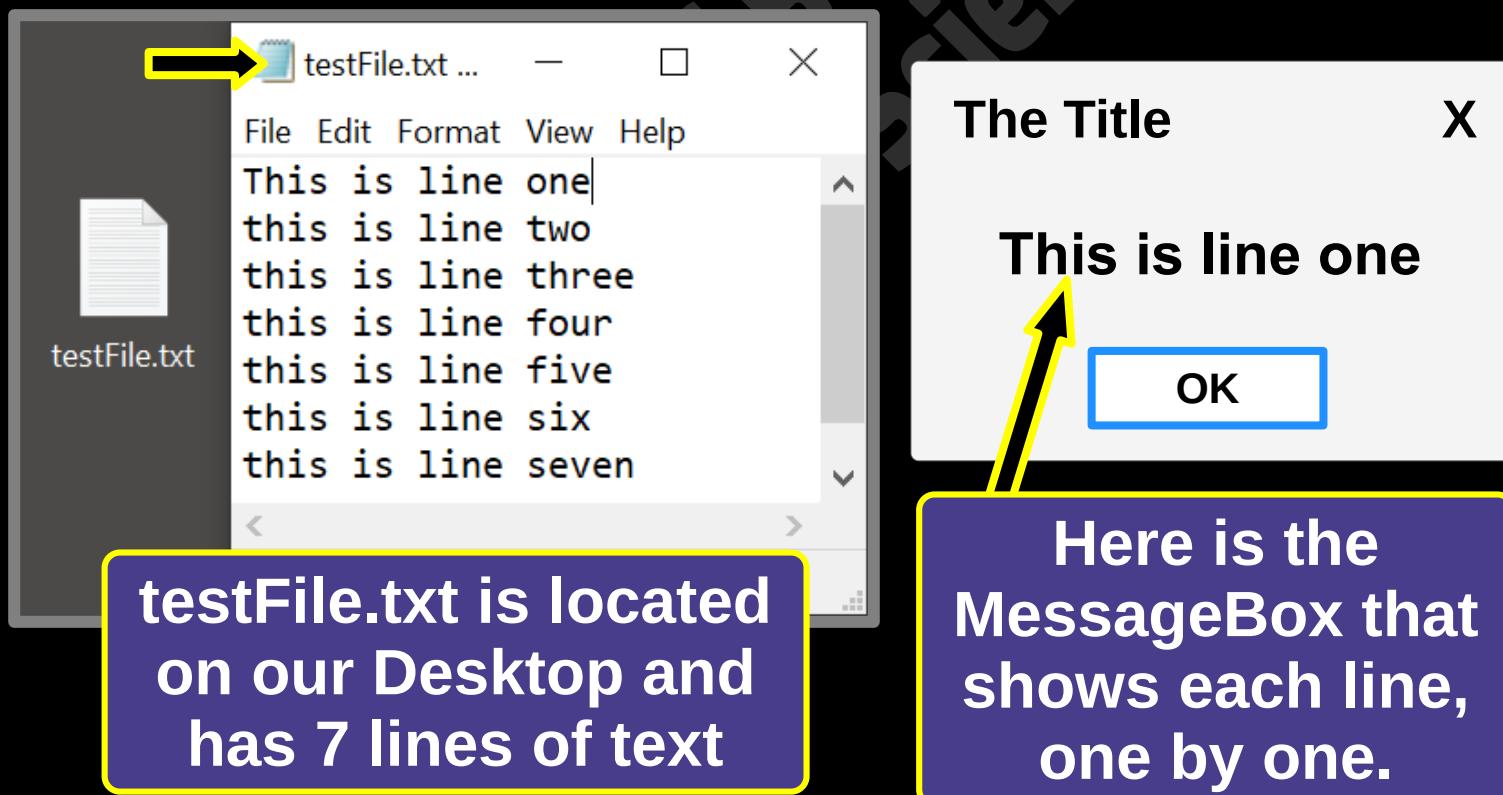
ourText = 'When coding in Python,\nwe can make multiple lines\nof text very easily. \n\nWe use backslash n to allow the\nmulti line style.'

```
ourFile = open('testFile.txt', 'w')
ourFile.write(ourText)
ourFile.close()
```



Read a Text File, for loop messages

```
import ctypes  
  
here = open('C:/Users/you/Desktop/testFile.txt')  
  
for x in here:  
    ctypes.windll.user32.MessageBoxW(0,  
        x, "The Title", 0)
```



Read Text File, Make Labels in Window

```
from tkinter import *
ourWindow = Tk()
ourWindow.geometry('400x300+200+100')
ourWindow.title('Howdy')

text = open('C:/Users/energy/Desktop/testFile.txt')

x = 75
y = 5

for z in text:
    ourFirstLabel = Label(ourWindow,
text = z, fg = 'aqua',
bg = 'black').place(x = x, y = y)
    y += 40

ourWindow.mainloop()
```

The diagram shows three yellow boxes at the top right with arrows pointing down to specific parameters in the code. The first box contains 'width' with an arrow pointing to the 'x' parameter in the 'place' method. The second box contains 'height' with an arrow pointing to the 'y' parameter in the 'place' method. The third box contains 'y pos' with an arrow pointing to the 'y' parameter in the 'place' method.

We read a text file, that has multiple lines of text. Each line of text becomes a label, that is positioned, evenly spaced, in our window.

**When placed
in Any Folder
and Activated,
this next script
shows the
Full Path Names
of All Files
in the Folder
and Subfolders**

Show File Path Names of Files in the Folder that this script is Run in

```
import os

path = os.path.dirname(os.path.realpath(__file__))

theFiles = []

for folder, subfolder, ourFiles in os.walk(path):
    for file in ourFiles:
        theFiles.append(os.path.join(folder,file))

for theName in theFiles:
    print(theName)

input('Press Enter to Exit')
```

two underscore



folder, means any folder we place this script in
subfolder, means any subfolders in the folder
ourFiles, means any files in folder & subfolders

Show File Path Names of Files in the Folder

```
import os
```

by extension type

```
path = os.path.dirname(os.path.realpath(__file__))
```

```
for folder, subfolder, ourFiles in os.walk(path):  
    for file in ourFiles:
```

```
        if(file.endswith(".jpg")):
```

```
            print(os.path.join(folder,file))
```

```
input('Press Enter to Exit')
```

When this script is placed in any Folder and Run, it shows the full path names of the files in that folder and its subfolders, that have the extension .jpg

We specify any file type that we want, such as .txt, .ods, .odt, .pdf, .gif, .png or other.

We can also specify more extension types to show, by using the or operator, as shown on the next page.

Show File Path Names of Files in the Folder

```
import os  
by extension types  
  
path = os.path.dirname(os.path.realpath(__file__))  
  
for folder, subfolder, ourFiles in os.walk(path):  
    for file in ourFiles:  
        if(file.endswith(".gif") or file.endswith(".jpg")):  
            print(os.path.join(folder,file))  
  
input('Press Enter to Exit')
```

When this script is placed in any Folder and Run, it shows the full path names of the files in that folder and its subfolders, that have the extension .gif **or** .jpg

We may specify as many file types as we want. We use the **or** operator as shown below, to specify multiple extension types.

```
if(file.endswith(".gif") or file.endswith(".jpg") or  
file.endswith(".bmp") or file.endswith(".tiff")):
```

Show File Path Names, Write to Text File

```
import os
```

with line breaks

```
path = os.path.dirname(os.path.realpath(__file__))
```

```
theFiles = [ ]
```

```
for folder, subfolder, ourFiles in os.walk(path):
```

```
    for file in ourFiles:
```

```
        theFiles.append(os.path.join(folder,file))
```

```
with open('testFile.txt', 'w') as ourNewFile:
```

```
    for item in theFiles:
```

```
        ourNewFile.write("%s\n" % item)
```

```
print("\n".join(theFiles))
```

```
input('Press Enter to Exit')
```

When this script is placed in any Folder and Run, it shows the full path names of the files in that folder and its subfolders and creates a TEXT FILE with those full path names shown on new lines.

**This next script
shows the
File Path Names
of the Files
in Any Folder that
we SPECIFY,
including its
Subfolder Files**

Show File Path Names of Files in Any Folder

```
import os
```

```
path = "C:/Users/energy/Desktop/ourApp"
```

```
theFiles = []
```

We specify the exact location
of the folder we want to use

```
for folder, subfolder, ourFiles in os.walk(path):
```

```
    for file in ourFiles:
```

```
        theFiles.append(os.path.join(folder,file))
```

```
for theName in theFiles:
```

```
    print(theName)
```

```
input('Press Enter to Exit')
```

that we specify

Shows us the full path names of all files in the folder we specified, and its subfolder files too.

On my computer, I'm using the username **energy**. But, in your case, you write the name of **your username** instead.

Comments

Single Line

Multi Line

COMMENT CODE OUT - Single Line

```
ourText = 'Hi Everyone'  
print(ourText)  
# print('Howdy')
```

Type the Pound
Symbol

#

is also known
as the Number
Symbol

is used for
Single Line
Commenting

We place # before any line of code to comment that line out.

The pound symbol is also known as the number symbol.

The # symbol is used to comment ONLY ONE LINE OF CODE AT A TIME!

The Next page teaches you Multi Line Commenting.

COMMENT CODE OUT - Multi Line

```
→ "ourText = 'Hi Everyone'  
print(ourText)" ←  
print('Howdy')
```

Type the Single
Quote Symbol
3 Times

'''

' is also known
as the
Apostrophe
Symbol

3 Single Quotes
are used for Multi
Line Commenting

We place "" before the first line
of code to we want to comment
out, and then place "" at the end

The Single Quote is also known
as the Apostrophe symbol.

3 Single Quotes "" are used to
comment MULTIPLE LINES
OF CODE AT A TIME.

Next, we learn how to use
comments to document our
code with detailed information.

COMMENT INFO - Single Line

```
ourText = 'Hi Everyone'
```

```
print(ourText) #shows Hi Everyone
```

```
print('Howdy') #shows Howdy
```

Type the Pound
Symbol

#

is also known
as the Number
Symbol

is used for
Single Line
Commenting

This time, instead of using the # Comment symbol to comment code out, we instead use it to COMMENT about the code!

Adding Comment information to our Code is very useful for small and large programs.

It is very good to add many comments to our code as we are making our applications.

COMMENT INFO – Multi Line

```
ourText = 'Hi Everyone'
```

```
print(ourText)
```

```
print('Howdy')
```

```
''' We assign a value to ourText  
we print ourText and Howdy.'''
```

Type the Single
Quote Symbol
3 Times

'''

' is also known
as the
Apostrophe
Symbol

3 Single Quotes ""
are used for Multi
Line Commenting

Instead of using the
3 Single Quotes "" to
comment code out,
we now use it to make
multi line COMMENTS
about the code!

Adding Multi Line Comment
information to our Code is
very useful in of all our scripts.
We document our code with
comments as we are making
our applications.

Long Lines to Multi Lines is Allowed

```
import ctypes
```

```
ourTitle = 'Instructions'
```

```
ourText = 'When coding in Python. \
We can make long lines \
of text. We use the \
backslash to allow the \
multi line style.'
```

```
ctypes.windll.user32.MessageBoxW(0,
ourText, ourTitle, 0)
```

Backslash

We can use the **BACK SLASH ** to tell the Python editor to allow **long lines of code** to be placed onto **multiple lines** instead the one long line. This is very useful for increased readability, for the person scripting.

Long Lines to Multi Lines is Allowed

```
import ctypes
```

```
ourTitle = 'Instructions'
```

```
ourText = ('When coding in Python.  
We can make long lines  
of text. We use the  
backslash to allow the  
multi line style.') ←
```

Parenthesis

```
ctypes.windll.user32.MessageBoxW(0,  
ourText, ourTitle, 0)
```

Another way that we can achieve this increased readability is with **PARENTHESES** and **QUOTATION MARKS**, to tell the Python editor to allow long lines to be allowed as a multiple line section, to make it easier for the person scripting.

VARIABLES & FUNCTIONS

College of Scripting Music & Science

Global, Local Variables, Function, return

```
import ctypes
```

```
ourTitle = "Greeting App"
```

Global Variable

```
def ourFunction():
```

custom function

```
    ourText = "Howdy"
```

Local Variable

```
    return ourText
```

return the Variable

```
ctypes.windll.user32.MessageBoxW(0,  
ourFunction(), ourTitle, 0)
```

ourTitle is a global variable, that is a string.
ourText is a local variable, that is a string.

ourTitle can be used in other functions,
because it is global.

ourText can only be used in **ourFunction()**,
because it is local to **ourFunction()**.

TABBING CODE for Our Functions

```
import ctypes
```

```
ourTitle = ''
```

```
ourText = ''
```

Empty
Global
String
Variables

Notice, we use
the TAB button
on our Keyboard!

TAB

TABS are
important in
Python
Language!

```
def ourTitleFunction():
```

```
    ↪ ourTitle = "Greeting App"
```

```
    ↪ return ourTitle
```

```
def ourTextFunction():
```

```
    ↪ ourText = "Hi Everyone"
```

```
    ↪ return ourText
```

```
ctypes.windll.user32.MessageBoxW(0,  
ourTextFunction(), ourTitleFunction(), 0)
```

We make our custom function by writing
def ourTitleFunction(): and on the next line,
we press the **TAB** button on our **keyboards**,
to add a TAB space.

INTERNET

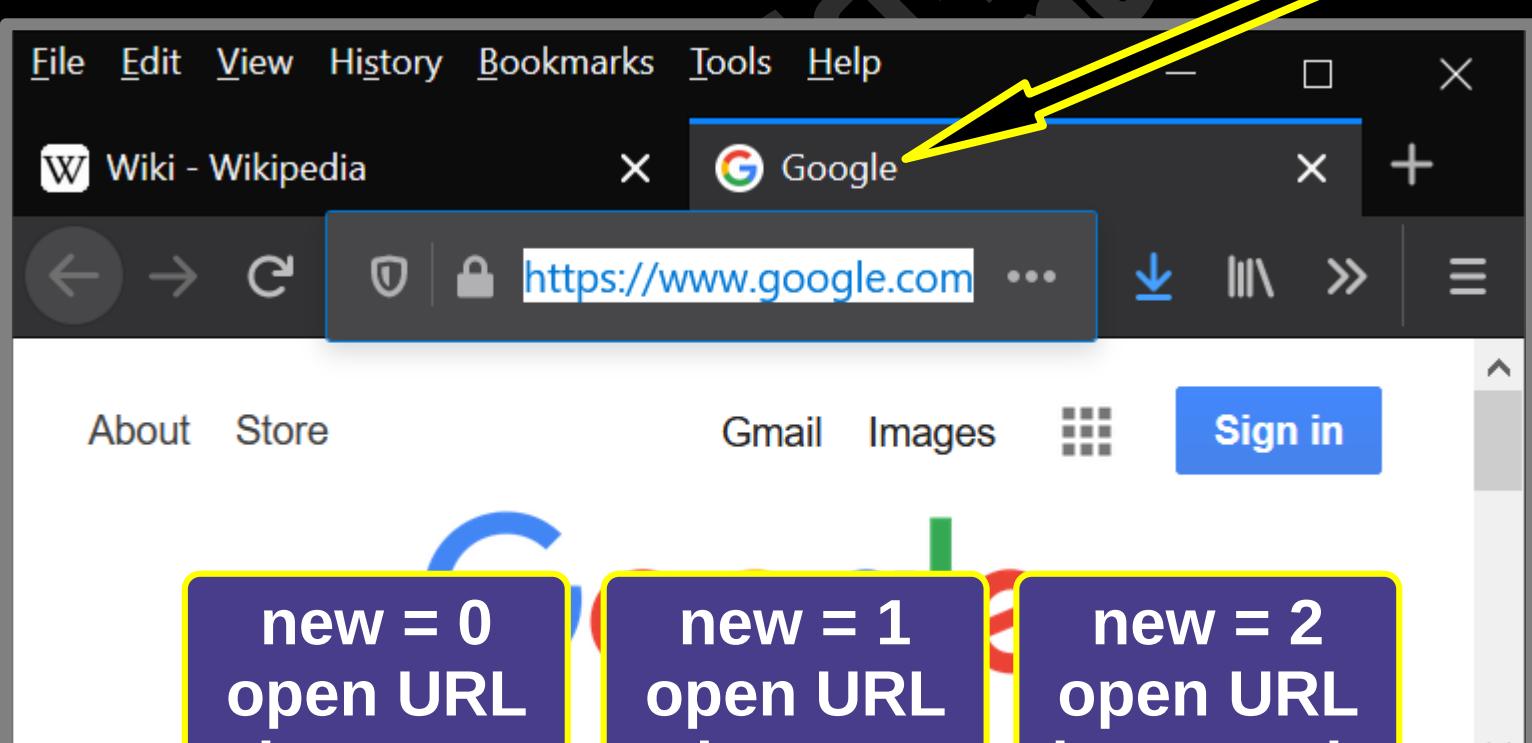
College of Scripting
Music & Science

Open Web Browser, Go to URL

```
import webbrowser
```

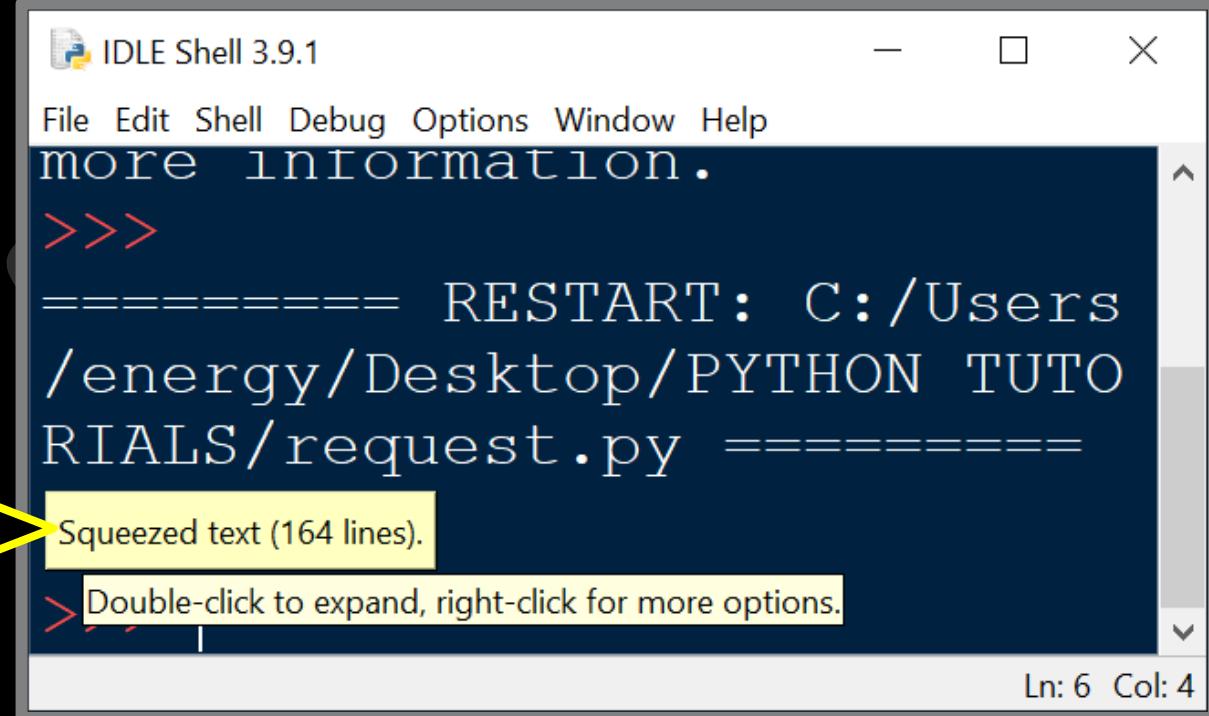
```
theUrl = 'https://google.com'
```

```
webbrowser.open(theUrl, new=2)
```



Read Web Page Data, `urllib`

```
import urllib.request  
  
temp = 'https://www.google.com/'  
  
theUrl = urllib.request.urlopen(temp)  
print(theUrl.read())  
  
input('Press Enter to Close App')
```



The screenshot shows the Python IDLE Shell interface. The code in the shell window is:

```
IDLE Shell 3.9.1  
File Edit Shell Debug Options Window Help  
more information.  
===== RESTART: C:/Users/energy/Desktop/PYTHON TUTORIALS/request.py =====  
Squeezed text (164 lines).  
> Double-click to expand, right-click for more options.
```

A yellow callout box with a yellow arrow points to the text "Squeezed text (164 lines.)". The status bar at the bottom right of the window shows "Ln: 6 Col: 4".

Double
Click to
Show
the
Data

Read Web Page Data, `urllib`, wrapped

```
import urllib.request  
theUrl = (urllib.request.urlopen(  
    'https://www.google.com/'))  
print(theUrl.read())  
input('Press Enter to Close App')
```

This script is very similar to the script on the previous page, but, in this version, we have **wrapped the code statement with parenthesis**, to allow us to put the long line of code on multiple lines, for easy coding!

List of Dictionaries

College of Scripting
Music & Science

List of Dictionaries - Show All

```
people = [  
    {  
        'name' : 'Melissa',  
        'date' : '2021/04/01'  
    },  
    {  
        'name' : 'Tabitha',  
        'date' : '2021/04/05'  
    }  
]  
print(people)  
input('Press Enter to Exit')
```

```
[{'name': 'Melissa', 'date': '2021/04/01'}, {'name':  
 'Tabitha', 'date': '2021/04/05'}]
```

List of Dictionaries - Show All

```
people = [  
    {  
        'name' : 'Melissa',  
        'age' : 47,  
    },  
    {  
        'name' : 'Tabitha',  
        'age' : 52  
    }  
]  
print(people)  
input('Press Enter to Exit')
```

```
[{'name': 'Melissa', 'age': 47}, {'name': 'Tabitha',  
'age': 52}]
```

List of Dictionaries - Show All

```
people = [
{
    'name' : 'Melissa',
    'date' : '2021/04/01',
},
{
    'name' : 'Tabitha',
    'date' : '2021/04/05'
}
]
x = 0
while x < len(people):
    print(people[x])
    x += 1
input('Press Enter to Exit')
```

```
{'name': 'Melissa', 'date': '2021/04/01'}
{'name': 'Tabitha', 'date': '2021/04/05'}
```

List of Dictionaries - Show All

```
people = [
{
    'name' : 'Melissa',
    'date' : '2021/04/01',
},
{
    'name' : 'Tabitha',
    'date' : '2021/04/05'
}
]
for z in range(len(people)):
    print(people[z])
input('Press Enter to Exit')
```

```
{'name': 'Melissa', 'date': '2021/04/01'}
{'name': 'Tabitha', 'date': '2021/04/05'}
```

List of Dictionaries SHOW KEY

List of Dictionaries - Show All Names

```
people = [
    {
        'name' : 'Melissa',
        'date' : '2021/04/01',
    },
    {
        'name' : 'Tabitha',
        'date' : '2021/04/05'
    }
]
x = 0
while x < len(people):
    print(people[x]['name'])
    x += 1
input('Press Enter to Exit')
```

Melissa
Tabitha

List of Dictionaries - Show All Dates

```
people = [
{
    'name' : 'Melissa',
    'date' : '2021/04/01',
},
{
    'name' : 'Tabitha',
    'date' : '2021/04/05'
}
]
x = 0
while x < len(people):
    print(people[x]['date'])
    x += 1
input('Press Enter to Exit')
```

List of Dictionaries - Show First Letter

```
people = [  
    {  
        'name' : 'Melissa',  
        'date' : '2021/04/01',  
    },  
    {  
        'name' : 'Tabitha',  
        'date' : '2021/04/05'  
    }  
]  
x = 0  
while x < len(people):  
    print(people[x]['name'][0])  
    x += 1  
input('Press Enter to Exit')
```

M
T

List of Dictionaries - Show First Letters

```
people = [  
    {  
        'name' : 'Melissa',  
        'date' : '2021/04/01',  
    },  
    {  
        'name' : 'Tabitha',  
        'date' : '2021/04/05'  
    }  
]  
x = 0  
while x < len(people):  
    print(people[x]['name'][0] +  
          people[x]['name'][1] +  
          people[x]['name'][2])  
    x += 1  
input('Press Enter to Exit')
```

Mel
Tab

List of Dictionaries - Show All Names

```
people = [  
    {  
        'name' : 'Melissa',  
        'date' : '2021/04/01',  
    },  
    {  
        'name' : 'Tabitha',  
        'date' : '2021/04/05'  
    }  
]  
for z in range(len(people)):  
    print(people[z]['name'])  
input('Press Enter to Exit')
```

List of Dictionaries - Show All Names

```
people = [
{
    'name' : 'Melissa',
    'date' : '2021/04/01',
},
{
    'name' : 'Tabitha',
    'date' : '2021/04/05'
}
]
for z in range(len(people)):
    print(people[z]['name'] + '\n')
input('Press Enter to Exit')
```

List of Dictionaries – Show All Dates

```
people = [
{
    'name' : 'Melissa',
    'date' : '2021/04/01',
},
{
    'name' : 'Tabitha',
    'date' : '2021/04/05'
}
]
for z in range(len(people)):
    print(people[z]['date'] + '\n')
input('Press Enter to Exit')
```

List of Dictionaries - Show Name, Date

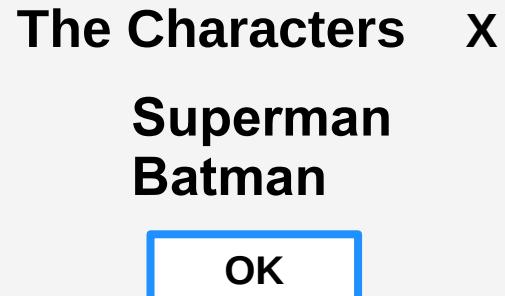
```
people = [  
    {  
        'name' : 'Melissa',  
        'date' : '2021/04/01'  
    },  
    {  
        'name' : 'Tabitha',  
        'date' : '2021/04/05'  
    }  
]  
for z in range(len(people)):  
    print(people[z]['name'] + '\n' +  
          people[z]['date'] + '\n\n')  
input('Press Enter to Exit')
```

List of Dictionaries - Show All

```
import ctypes
characters = [
{
    'name': 'Superman',
    'location': 'Metropolis'
},
{
    'name': 'Batman',
    'location': 'Gotham'
}
]
answer = ""

for z in range(len(characters)):
    answer += characters[z]['name'] + '\n'
    print(z)

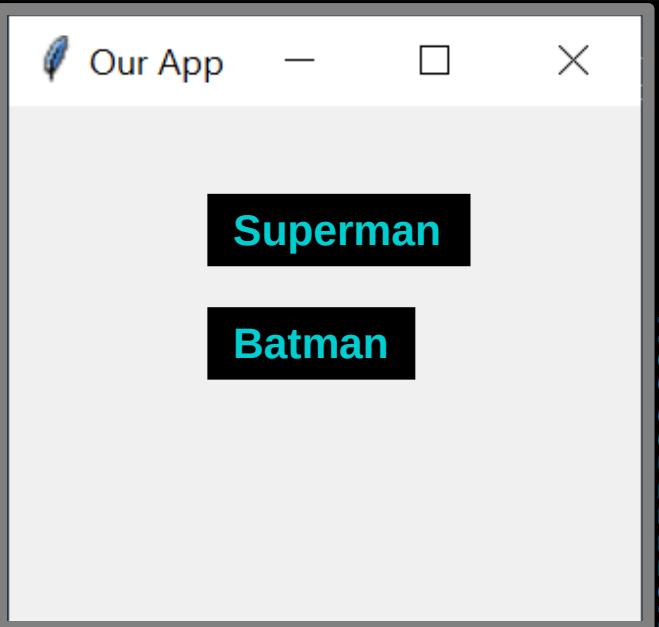
ctypes.windll.user32.MessageBoxW(0,
answer, "The Characters", 0)
```



List of Dictionaries in Window, Labels

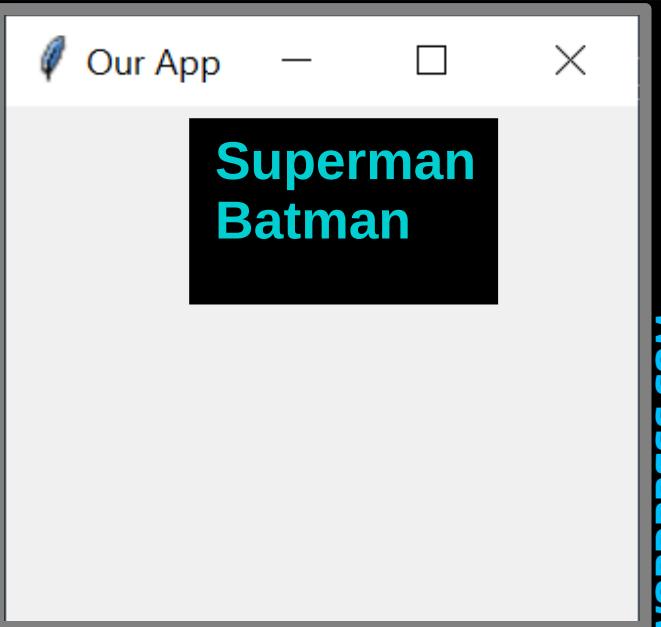
Christopher Topalian

```
from tkinter import *
characters = [
{
    'name': 'Superman',
    'location': 'Metropolis'
},
{
    'name': 'Batman',
    'location': 'Gotham'
}
]
ourWindow = Tk()
ourWindow.geometry('215x175+300+150')
ourWindow.title('Our App')
x = 75
y = 1
answer = ""
for z in range(len(characters)):
    answer = characters[z]['name']
    y += 40
    Label(ourWindow,
          text = answer, fg = 'aqua', bg = 'black').place(x = x, y = y)
ourWindow.mainloop()
```



List of Dictionaries in Window, 1 label

```
from tkinter import *
characters = [
    {
        'name': 'Superman',
        'location': 'Metropolis'
    },
    {
        'name': 'Batman',
        'location': 'Gotham'
    }
]
ourWindow = Tk()
ourWindow.geometry('215x175+300+150')
ourWindow.title('Our App')
x = 75
y = 0
answer = " "
for z in range(len(characters)):
    answer += characters[z]['name'] + '\n'
ourFirstLabel = Label(ourWindow,
text = answer, fg = 'aqua', bg = 'black').place(x = x, y = y)
ourWindow.mainloop()
```



List of Dictionaries

FILTER BY **NAME**

List of Dictionaries – Show Name

```
people = [
{
    'name' : 'Melissa',
    'date' : '2021/04/01',
},
{
    'name' : 'Tabitha',
    'date' : '2021/04/05'
}
]
x = 0
while x < len(people):
    if(people[x]['name']=='Melissa'):
        print('Hi Melissa')
    x += 1
input('Press Enter to Exit')
```

List of Dictionaries – Show Name

```
people = [
```

```
{
```

```
    'name' : 'Melissa',  
    'date' : '2021/04/01',
```

```
},
```

```
{
```

```
    'name' : 'Tabitha',  
    'date' : '2021/04/05'
```

```
}
```

```
]
```

```
for z in range(len(people)):
```

```
    if(people[z]['name']=='Melissa'):
```

```
        print('Hi Melissa')
```

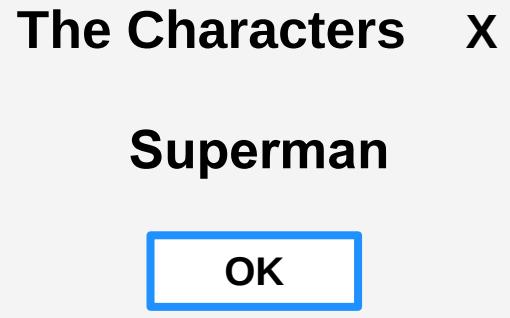
```
input('Press Enter to Exit')
```

Hi Melissa

List of Dictionaries - Show Name

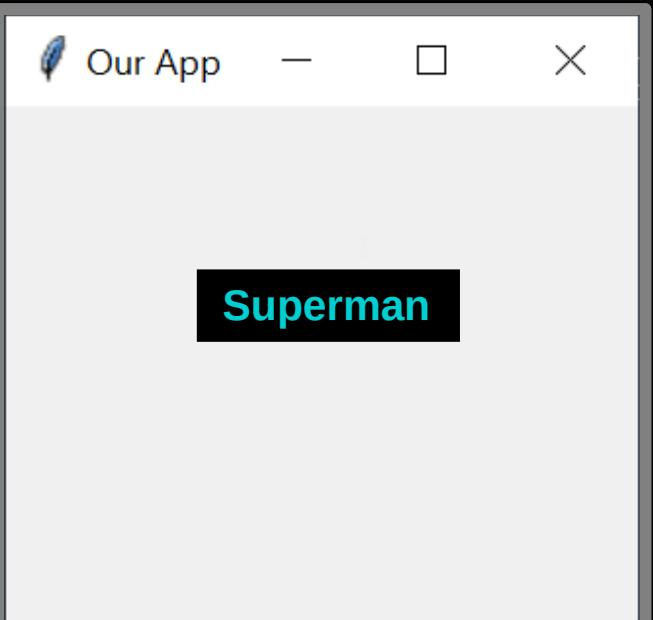
```
import ctypes
characters = [
{
    'name': 'Superman',
    'location': 'Metropolis'
},
{
    'name': 'Batman',
    'location': 'Gotham'
}
]
answer = " "
for z in range(len(characters)):
    if(characters[z]['name']=='Superman'):
        answer += characters[z]['name'] + '\n'
print(z)

ctypes.windll.user32.MessageBoxW(0,
answer, "The Characters", 0)
```



List of Dictionaries in Window – Show Name

```
from tkinter import *
characters = [
    {
        'name': 'Superman',
        'location': 'Metropolis'
    },
    {
        'name': 'Batman',
        'location': 'Gotham'
    }
]
ourWindow = Tk()
ourWindow.geometry('215x175+300+150')
ourWindow.title('Our App')
x = 75
y = 1
answer = ""
for z in range(len(characters)):
    if(characters[z]['name']=='Superman'):
        answer = characters[z]['name']
        y += 40
Label(ourWindow,
      text = answer, fg = 'aqua', bg = 'black').place(x = x, y = y)
ourWindow.mainloop()
```



List of Dictionaries

FILTER BY **DATE**

List of Dictionaries – if Date Greater Than

```
people = [  
    {  
        'name' : 'Melissa',  
        'date' : '2021/04/01',  
    },  
    {  
        'name' : 'Tabitha',  
        'date' : '2021/04/05'  
    }  
]  
x = 0  
while x < len(people):  
    if(people[x]['date'] > '2021/04/02'):  
        print(people[x])  
    x += 1  
input('Press Enter to Exit')
```

{'name': 'Tabitha', 'date': '2021/04/05'}

List of Dictionaries – if Date Greater Than

```
people = [
    {
        'name' : 'Melissa',
        'date' : '2021/04/01',
    },
    {
        'name' : 'Tabitha',
        'date' : '2021/04/05'
    }
]

for z in range(len(people)):
    if(people[z]['date'] > '2021/04/02'):
        print(people[z])
input('Press Enter to Exit')
```

{'name': 'Tabitha', 'date': '2021/04/05'}

List of Dictionaries – if Date Greater Than

```
people = [
    {
        'name' : 'Melissa',
        'date' : '2021/04/01',
    },
    {
        'name' : 'Tabitha',
        'date' : '2021/04/05'
    }
]
x = 0
while x < len(people):
    if(people[x]['date'] > '2021/04/02'):
        print(people[x]['name'])
    x += 1
input('Press Enter to Exit')
```

List of Dictionaries – if Date Greater Than

```
people = [  
    {  
        'name' : 'Melissa',  
        'date' : '2021/04/01'  
    },  
    {  
        'name' : 'Tabitha',  
        'date' : '2021/04/05'  
    }  
]  
for z in range(len(people)):  
    if(people[z]['date'] > '2021/04/02'):  
        print(people[z]['name'])  
input('Press Enter to Exit')
```

List of Dictionaries - if Date Less Than

```
people = [
    {
        'name' : 'Melissa',
        'date' : '2021/04/01',
    },
    {
        'name' : 'Tabitha',
        'date' : '2021/04/05'
    }
]
x = 0
while x < len(people):
    if(people[x]['date'] < '2021/04/02'):
        print(people[x]['name'] + '\n' +
              people[x]['date'])
    x += 1
input('Press Enter to Exit')
```

Filter by
Year,
Month, Day

Melissa
2021/04/01

List of Dictionaries - if Date Less Than

```
people = [  
    {  
        'name' : 'Melissa',  
        'date' : '2021/04/01',  
    },  
    {  
        'name' : 'Tabitha',  
        'date' : '2021/04/05'  
    }  
]  
for z in range(len(people)):  
    if(people[z]['date'] < '2021/04/02'):  
        print(people[z]['name'] + '\n' +  
              people[z]['date'])  
input('Press Enter to Exit')
```

Filter by
Year,
Month, Day

Melissa
2021/04/01

List of Dictionaries - if Date Greater Than

FILTER BY
YEAR

```
people = [  
    {  
        'name' : 'Melissa',  
        'date' : '2021/04/01',  
    },  
    {  
        'name' : 'Tabitha',  
        'date' : '2021/04/05'  
    }  
]  
x = 0  
while x < len(people):  
    if(people[x]['date'] > '2021'):  
        print(people[x])  
    x += 1  
input('Press Enter to Exit')
```

```
{'name': 'Melissa', 'date': '2021/04/01'}  
'{'name': 'Tabitha', 'date': '2021/04/05'}
```

List of Dictionaries - if Date Greater Than

FILTER BY
YEAR AND
MONTH

```
people = [  
    {  
        'name' : 'Melissa',  
        'date' : '2021/04/01',  
    },  
    {  
        'name' : 'Tabitha',  
        'date' : '2021/04/05'  
    }  
]  
x = 0  
while x < len(people):  
    if(people[x]['date'] > '2021/04'):  
        print(people[x])  
    x += 1  
input('Press Enter to Exit')
```

```
{'name': 'Melissa', 'date': '2021/04/01'}  
'{'name': 'Tabitha', 'date': '2021/04/05'}
```

List of Dictionaries – if Date Equal To

FILTER BY
EXACT
DATE

```
people = [  
    {  
        'name' : 'Melissa',  
        'date' : '2021/04/01',  
    },  
    {  
        'name' : 'Tabitha',  
        'date' : '2021/04/05'  
    }  
]  
x = 0  
while x < len(people):  
    if(people[x]['date'] == '2021/04/05'):  
        print(people[x])  
    x += 1  
input('Press Enter to Exit')
```

{'name': 'Tabitha', 'date': '2021/04/05'}

List of Dictionaries – if Date Greater Than

```
people = [
    {
        'name' : 'Melissa',
        'date' : '2021/04/01 10:00AM'
    },
    {
        'name' : 'Tabitha',
        'date' : '2021/04/01 12:00AM'
    }
]
x = 0
while x < len(people):
    if(people[x]['date'] > '2021/04/01 08:00AM'):
        print(people[x])
    x += 1
input('Press Enter to Exit')
```

**FILTER BY
EXACT
DATE and
TIME**

```
{'name': 'Melissa', 'date': '2021/04/01 10:00AM'}
{'name': 'Tabitha', 'date': '2021/04/01 12:00AM'}
```

List of Dictionaries FILTER BY **DATE** using and

if Date is Greater Than and Less Than

```
people = [  
    {  
        'name' : 'Melissa',  
        'date' : '2021/03/31',  
    },  
    {  
        'name' : 'Tabitha',  
        'date' : '2021/04/05'  
    }  
]  
x = 0  
while x < len(people):  
    if(people[x]['date'] > '2021/03/30' and  
       people[x]['date'] < '2021/04/04'):  
        print(people[x]['name'])  
    x += 1  
input('Press Enter to Exit')
```

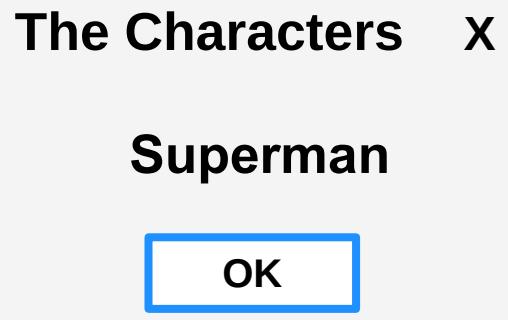
if Date is Greater Than and Less Than

```
people = [  
    {  
        'name' : 'Melissa',  
        'date' : '2021/03/31'  
    },  
    {  
        'name' : 'Tabitha',  
        'date' : '2021/04/05'  
    }  
]  
for z in range(len(people)):  
    if(people[z]['date'] > '2021/03/30' and  
       people[z]['date'] < '2021/04/04'):  
        print(people[z]['name'])  
input('Press Enter to Exit')
```

List of Dictionaries – if name and loc

Christopher Topalian

```
import ctypes
characters = [
{
    'name': 'Superman',
    'location': 'Metropolis'
},
{
    'name': 'Batman',
    'location': 'Gotham'
}
]
answer = " "
for z in range(len(characters)):
    if(characters[z]['name']=='Superman'
    and characters[z]['location']=='Metropolis'):
        answer += characters[z]['name'] + '\n'
ctypes.windll.user32.MessageBoxW(0,
answer, "The Characters", 0)
```

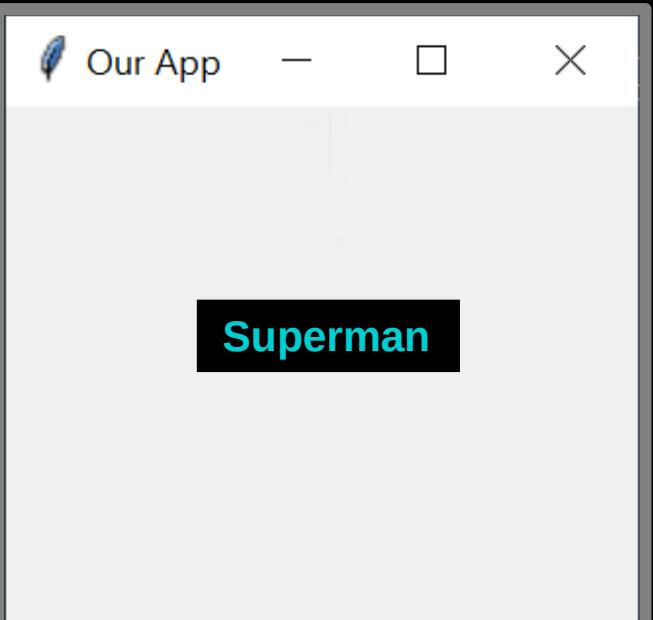


List of Dictionaries in Window - if name and loc

```

from tkinter import *
characters = [
    {
        'name': 'Superman',
        'location': 'Metropolis'
    },
    {
        'name': 'Batman',
        'location': 'Gotham'
    }
]
ourWindow = Tk()
ourWindow.geometry('215x175+300+150')
ourWindow.title('Our App')
x = 75
y = 1
answer = ""
for z in range(len(characters)):
    if(characters[z]['name']=='Superman'
       and characters[z]['location']=='Metropolis'):
        answer = characters[z]['name']
        y += 40
Label(ourWindow,
      text = answer, fg = 'aqua', bg = 'black').place(x = x, y = y)
ourWindow.mainloop()

```



List of Dictionaries

FILTER BY

DATE

using

or

if Date is Less Than or Greater Than

```
people = [
```

```
{
```

```
    'name' : 'Melissa',  
    'date' : '2021/03/31',
```

```
},
```

```
{
```

```
    'name' : 'Tabitha',  
    'date' : '2021/04/05'
```

```
}
```

```
]
```

```
x = 0
```

```
while x < len(people):
```

```
    if(people[x]['date'] < '2021/04/04' or
```

```
        people[x]['date'] > '2021/04/01'):
```

```
        print(people[x]['name'])
```

```
    x += 1
```

```
input('Press Enter to Exit')
```

Melissa
Tabitha

if Date is Less Than or Greater Than

```
people = [
{
    'name' : 'Melissa',
    'date' : '2021/03/31',
},
{
    'name' : 'Tabitha',
    'date' : '2021/04/05'
}
]
for z in range(len(people)):
    if(people[z]['date'] < '2021/04/04' or
       people[z]['date'] > '2021/04/01'):
        print(people[z]['name'])
input('Press Enter to Exit')
```

Melissa
Tabitha

if Date is Less Than or Greater Than

```
import ctypes
people = [
{
    'name' : 'Melissa',
    'date' : '2021/03/31',
},
{
    'name' : 'Tabitha',
    'date' : '2021/04/05'
}
]
answer = " "
for z in range(len(people)):
    if(people[z]['date'] < '2021/04/04'
    or people[z]['date'] > '2021/04/01'):
        answer += people[z]['name'] + '\n'
ctypes.windll.user32.MessageBoxW(0,
answer, "The People", 0)
```

The People

X

Melissa
Tabitha

OK

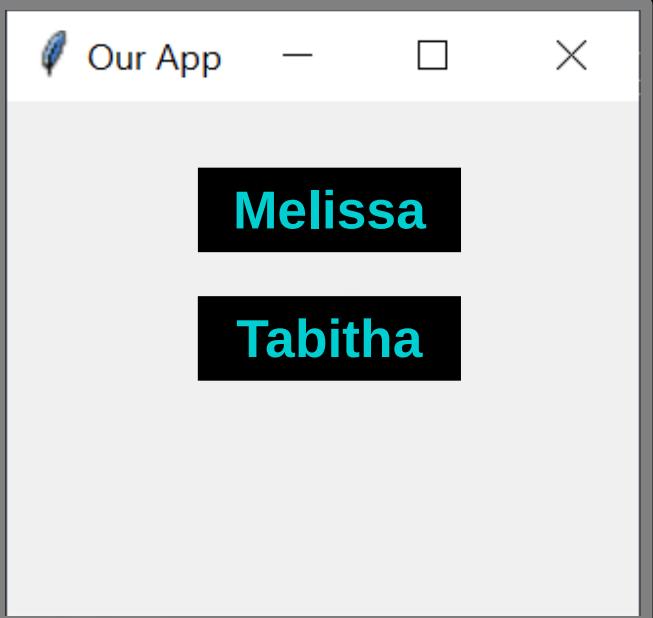
List of Dictionaries in Window – Show Name

```

from tkinter import *
people = [
    {
        'name' : 'Melissa',
        'date' : '2021/03/31',
    },
    {
        'name' : 'Tabitha',
        'date' : '2021/04/05'
    }
]
ourWindow = Tk()
ourWindow.geometry('215x175+300+150')
ourWindow.title('Our App')

x = 75
y = 1
answer = " "
for z in range(len(people)):
    if(people[z]['date'] < '2021/04/04'
    or people[z]['date'] > '2021/04/01'):
        answer = people[z]['name']
        y += 40
Label(ourWindow,
      text = answer, fg = 'aqua', bg = 'black').place(x = x, y = y)
ourWindow.mainloop()

```



List of Dictionaries **APPEND**

College of Scripting Music & Science

List of Dictionaries - APPEND

```
people = [
{
    'name' : 'Melissa',
    'date' : '2021/03/31',
},
{
    'name' : 'Tabitha',
    'date' : '2021/04/05'
}
]
people.append({'name':'John', 'date':'2022/03/23'})

x = 0
while x < len(people):
    if(people[x]['date'] > '2021/01/04' and
       people[x]['date'] < '2023/02/01'):
        print(people[x]['name'])
    x += 1
input('Press Enter to Exit')
```

Melissa
Tabitha
John

List of Dictionaries - APPEND

```
people = [
{
    'name' : 'Melissa',
    'date' : '2021/03/31',
},
{
    'name' : 'Tabitha',
    'date' : '2021/04/05'
}
]
people.append({'name':'John', 'date':'2022/03/23'})  
  
for z in range(len(people)):
    if(people[z]['date'] > '2021/01/04' and
       people[z]['date'] < '2023/02/01'):
        print(people[z]['name'])
input('Press Enter to Exit')
```

Melissa
Tabitha
John

List of Dictionaries **DELETE**

College of Scripting Music & Science

List of Dictionaries - Delete

```
people = [
    {
        'name' : 'Melissa',
        'date' : '2021/03/31',
    },
    {
        'name' : 'Tabitha',
        'date' : '2021/04/05'
    }
]
del people['name']=='Melissa']
x = 0
while x < len(people):
    if(people[x]['date'] > '2021/01/04' and
       people[x]['date'] < '2023/02/01'):
        print(people[x]['name'])
    x += 1
input('Press Enter to Exit')
```

List of Dictionaries - Delete

```
people = [
{
    'name' : 'Melissa',
    'date' : '2021/03/31',
},
{
    'name' : 'Tabitha',
    'date' : '2021/04/05'
}
]
del people['name']=='Melissa'

for z in range(len(people)):
    if(people[z]['date'] > '2021/01/04' and
       people[z]['date'] < '2023/02/01'):
        print(people[z]['name'])
input('Press Enter to Exit')
```

List of Dictionaries **DELETE** Last Item

List of Dictionaries - pop

```
people = [
    {
        'name' : 'Melissa',
        'date' : '2021/03/31',
    },
    {
        'name' : 'Tabitha',
        'date' : '2021/04/05'
    }
]
people.pop()
x = 0
while x < len(people):
    if(people[x]['date'] > '2021/01/04' and
       people[x]['date'] < '2023/02/01'):
        print(people[x]['name'])
    x += 1
input('Press Enter to Exit')
```

List of Dictionaries - pop

```
people = [  
    {  
        'name' : 'Melissa',  
        'date' : '2021/03/31',  
    },  
    {  
        'name' : 'Tabitha',  
        'date' : '2021/04/05'  
    }  
]  
people.pop()  
  
for z in range(len(people)):  
    if(people[z]['date'] > '2021/01/04' and  
        people[z]['date'] < '2023/02/01'):  
        print(people[z]['name'])  
input('Press Enter to Exit')
```

List of Dictionaries EDIT by Index

List of Dictionaries – Edit by Index

```
people = [
{
    'name' : 'Melissa',
    'date' : '2021/03/31',
},
{
    'name' : 'Tabitha',
    'date' : '2021/04/05'
}
]
people[0]=({'name':'John', 'date':'2022/03/23'})

x = 0
while x < len(people):
    if(people[x]['date'] > '2021/01/04' and
       people[x]['date'] < '2023/02/01'):
        print(people[x]['name'])
    x += 1
input('Press Enter to Exit')
```

List of Dictionaries – Edit by Index

```
people = [
{
    'name' : 'Melissa',
    'date' : '2021/03/31',
},
{
    'name' : 'Tabitha',
    'date' : '2021/04/05'
}
]
people[0]=({'name':'John', 'date':'2022/03/23'})

for z in range(len(people)):
    if(people[z]['date'] > '2021/01/04' and
       people[z]['date'] < '2023/02/01'):
        print(people[z]['name'])
input('Press Enter to Exit')
```

List of Dictionaries **EDIT by Key**

List of Dictionaries – Edit by Key

```
people = [
    {
        'name' : 'Melissa',
        'date' : '2021/03/31',
    },
    {
        'name' : 'Tabitha',
        'date' : '2021/04/05'
    }
]
people['name']=='Melissa']=({'name':'John',
                            'date':'2022/03/23'})

x = 0
while x < len(people):
    if(people[x]['date'] > '2021/01/04' and
       people[x]['date'] < '2023/02/01'):
        print(people[x]['name'])
    x += 1
input('Press Enter to Exit')
```

John
Tabitha

List of Dictionaries – Edit by Key

```
people = [
    {
        'name' : 'Melissa',
        'date' : '2021/03/31',
    },
    {
        'name' : 'Tabitha',
        'date' : '2021/04/05'
    }
]
people['name']=='Melissa']=({'name':'John',
                            'date':'2022/03/23'})

for z in range(len(people)):
    if(people[z]['date'] > '2021/01/04' and
       people[z]['date'] < '2023/02/01'):
        print(people[z]['name'])
input('Press Enter to Exit')
```

List of Dictionaries **EXTEND**

College of Scripting
Music & Science

List of Dictionaries – Extend

Christopher Topalian

```
people = [
    {
        'name' : 'Melissa',
        'date' : '2021/03/31',
    },
    {
        'name' : 'Tabitha',
        'date' : '2021/04/05'
    }
]
characters = [
    {
        'name' : 'James',
        'date' : '2010/08/31',
    },
    {
        'name' : 'Debra',
        'date' : '2012/06/02'
    }
]
people.extend(characters)
for z in range(len(people)):
    if(people[z]['date'] > '2009/01/04' and
       people[z]['date'] < '2023/02/01'):
        print(people[z]['name'])
input('Press Enter to Exit')
```

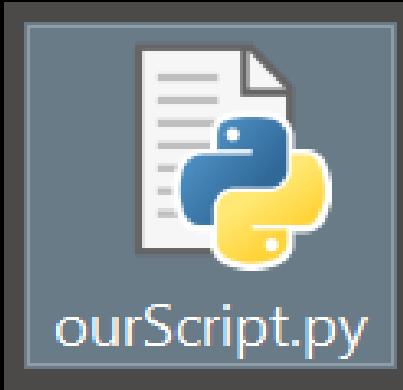
EXTERNAL FILE for EASIER READABILITY

aka

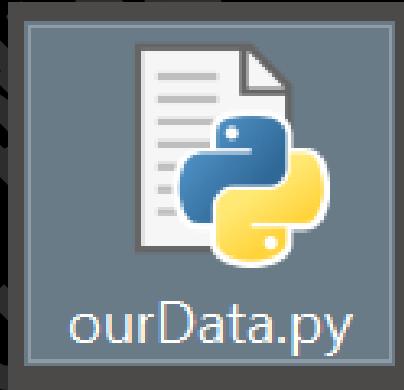
Making a Module

How to Make a MODULE

**Make a New
Script named
ourScript.py**

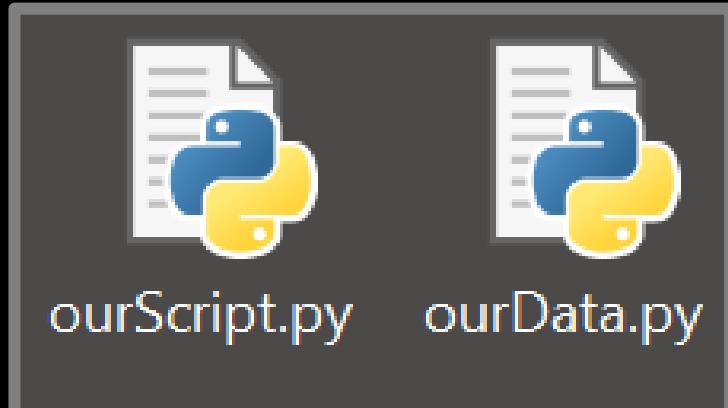


**Make a New
Script named
ourData.py**



ourScript.py will use the data that is located in the file named ourData.py

We place BOTH
ourScript.py
and
ourData.py
in the
SAME FOLDER



Import our Module using `import`

Christopher Topalian

Our External file is called a **MODULE**.
We saved our **module** as **ourData.py**

```
import ourData
```

```
print(ourData.ourText)
```

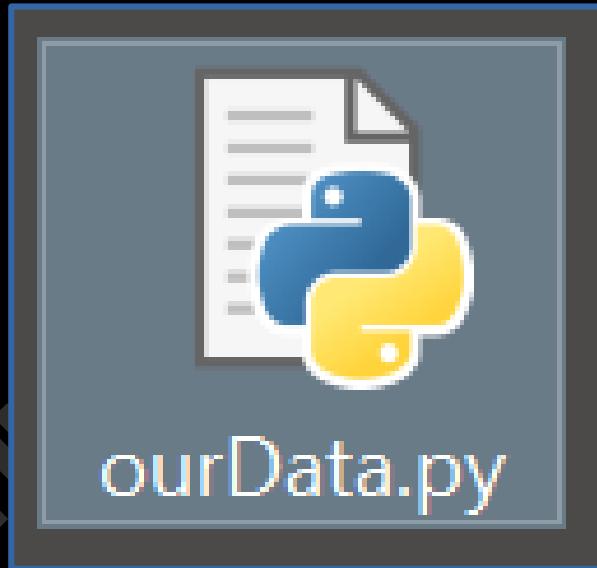
```
input('Press Enter to Exit')
```

We import ourData.py file by stating
`import ourData`

This method keeps the namespace of our code clean.
We include the **module name** to use the **data**.

↓ ↓
`print(ourData.ourText)`

First Way of Importing ourData.py



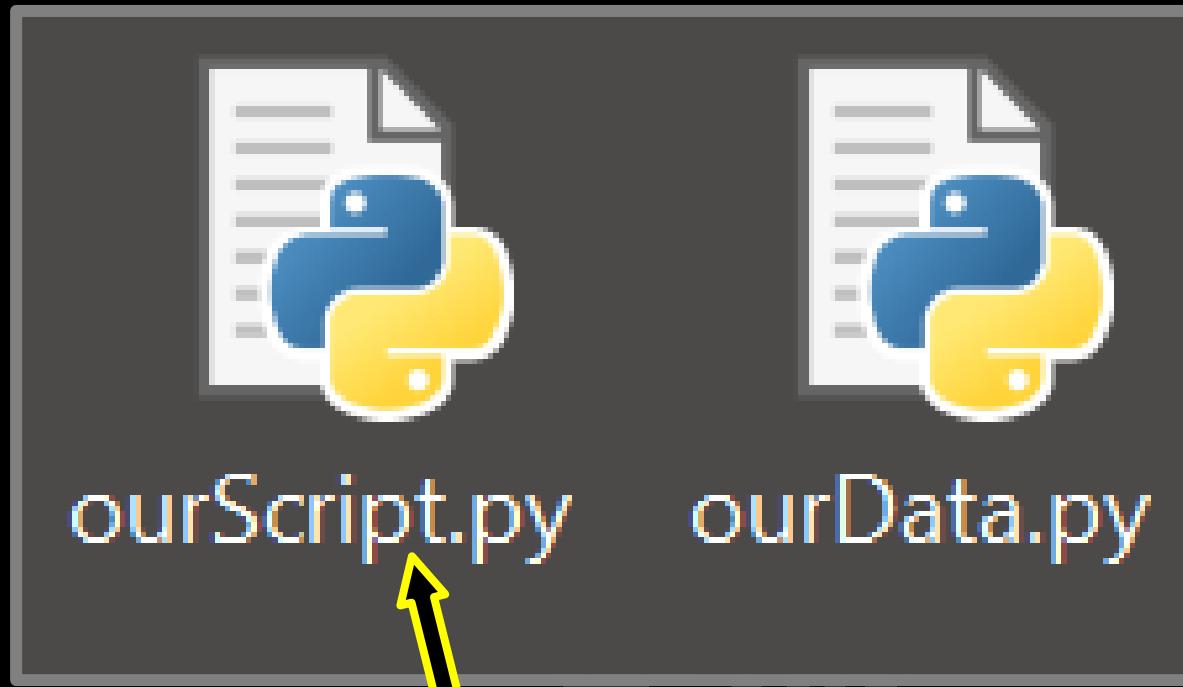
```
import ourData  
  
print(ourData.ourText)  
  
input('Press Enter to Exit')
```

ourScript.py imports the module named **ourData.py** and prints the **ourText** variable to the console.

ourText = 'Howdy'

ourData.py has only one line of code, which is the **ourText** variable assigned with the value of '**Howdy**'

This is called:
Creating a Module!



Double Click **ourScript.py**
and notice **how it USES** the Data
from **ourData.py**

Remember
ourScript.py
and
ourData.py
are in the
SAME FOLDER!

Second Way of Importing ourData.py

Our External file, is called a MODULE.
We saved our module as **ourData.py**

from ourData import*

makes coding easy, since we can use the data
without needing to include its reference name.

Allows us to use the **module data**
without having to include the **name of the module**,
but, it is also **less descriptive**, of **functions source!**

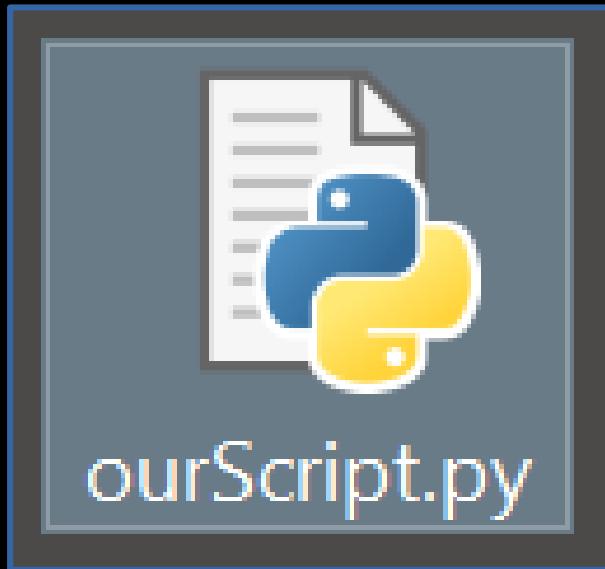
print(ourText)

from ourData import*

print(ourText)

input('Press Enter to Exit')

Second Way of Importing ourData.py



```
from ourData import*  
  
print(ourText)  
  
input('Press Enter to Exit')
```

ourScript.py imports the module named **ourData.py** and prints the **ourText** variable to the console.

ourData.py has only one line of code, which is the **ourText** variable assigned with the value of '**Howdy**'

This is called:
Creating a Module!

EXTERNAL FILE

Example 2

College of Scripting Music & Science

Our External Data File saved as **ourData.py**

```
speakersUSA = [  
    {  
        'name': 'Jane Doe',  
        'title': 'Genetics Lecture',  
        'date': '2022/04/20 07:00 AM'  
    },  
    {  
        'name': 'John Doe',  
        'title': 'Mechanics Lecture',  
        'date': '2022/04/20 09:00 AM'  
    }  
]  
  
speakersEU = [  
    {  
        'name': 'Rose Doe',  
        'title': 'Botany Lecture',  
        'date': '2022/04/20 11:00 AM'  
    },  
    {  
        'name': 'Ronald Doe',  
        'title': 'Electronics Lecture',  
        'date': '2022/04/20 01:00 PM'  
    }  
]
```

We save this script as **ourData.py**

On the next tutorial page, the script named **ourScript.py** uses the data from this page.

We place BOTH **ourData.py** and **ourScript.py** in the **SAME FOLDER!**

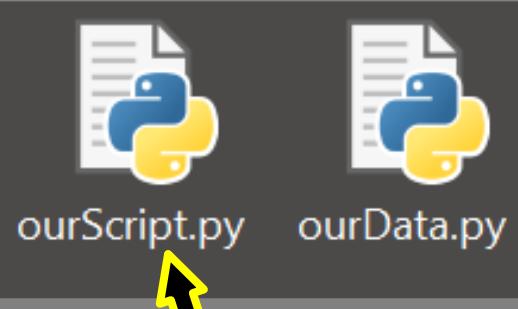
from ourData import*

```
import ctypes
from ourData import*
speakersUSA.append(speakersEU)
answer = " "
for z in speakersUSA:
    answer += str(z) + '\n\n'
ctypes.windll.user32.MessageBoxW(0,
answer, "The Speakers", 0)
```

Allows us to use the data without including module name, but doesn't state where the data came from!

We save this script as **ourScript.py**

This script USES the DATA from the previous page!



We have BOTH **ourScript.py** and **ourData.py** in the **SAME FOLDER!**

Double Click **ourScript.py** and notice how it USES the Data from **ourData.py**

import ourData

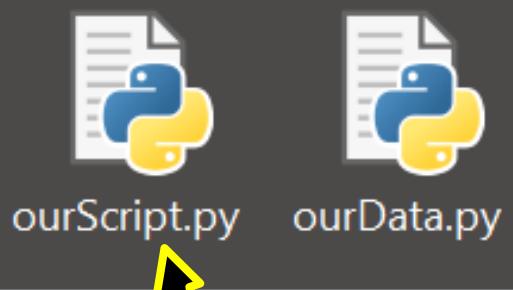
```
import ctypes, ourData  
ourData.speakersUSA.append(ourData.speakersEU)  
answer = ""  
for z in ourData.speakersUSA:  
    answer += str(z) + '\n\n'  
ctypes.windll.user32.MessageBoxW(0,  
answer, "The Speakers", 0)
```

Dot is used to access the data

Keeps Name Space Clean!

We save this script as **ourScript.py**

This script USES the DATA from a previous page!



We have BOTH **ourScript.py** and **ourData.py** in the **SAME FOLDER!**

Double Click **ourScript.py** and notice how it USES the Data from **ourData.py**

from ourData import*

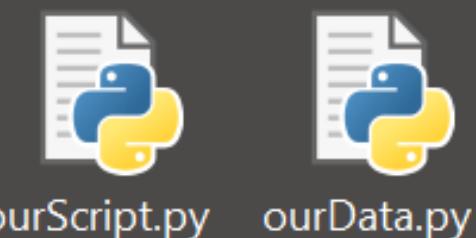
Christopher Topalian
append

```
import ctypes
from ourData import*
speakersUSA.append(speakersEU)
answer = " "
for z in speakersUSA:
    answer += str(name) + '\n\n'
ctypes.windll.user32.MessageBoxW(0,
answer, "The Speakers", 0)
```

Append the
list named
speakersEU
to the end of
the list
named
speakersUSA

We save this
script as
ourScript.py

This script
USES the DATA
from a
previous page!



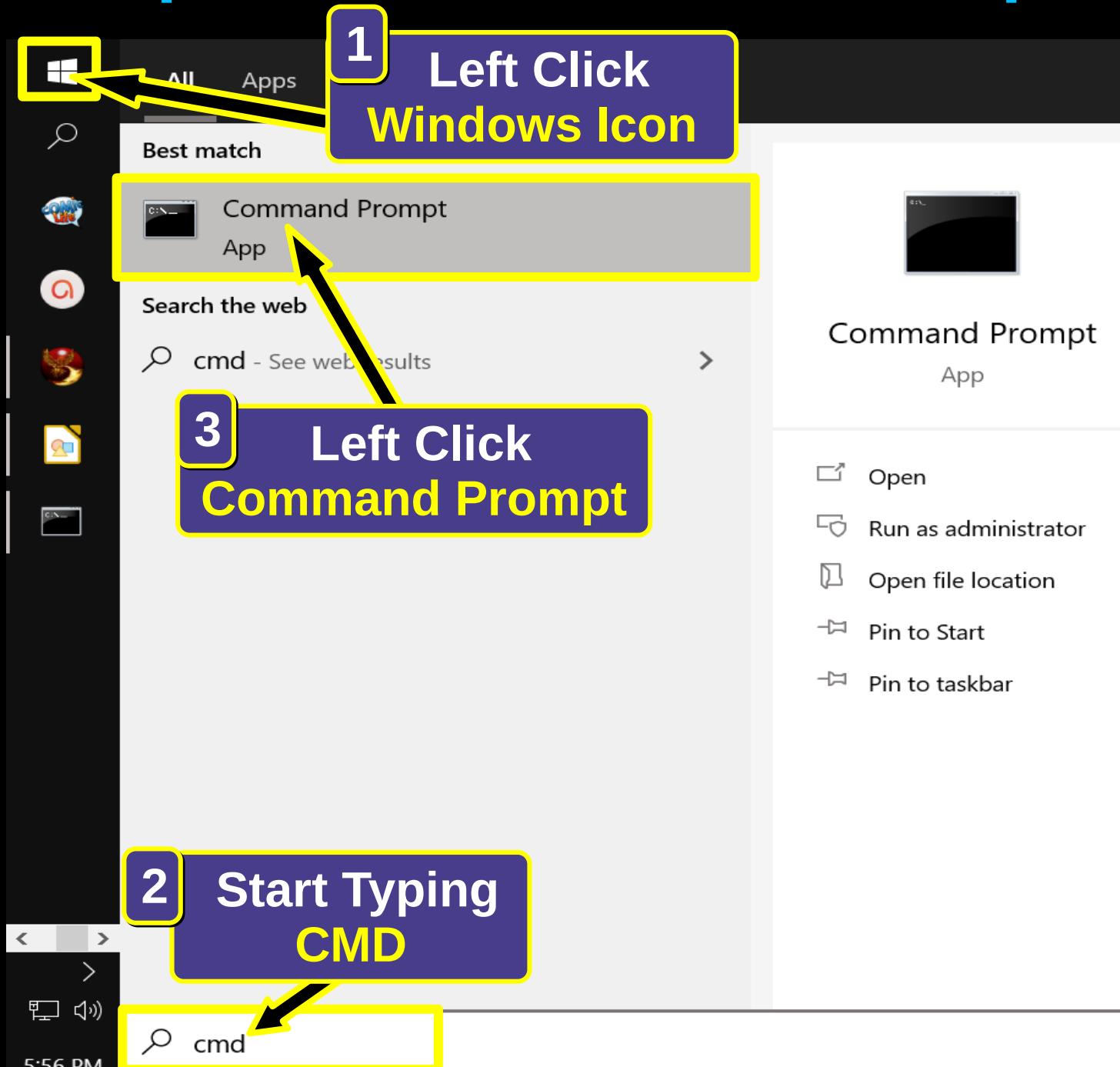
We have BOTH
ourScript.py
and
ourData.py
in the
SAME FOLDER!

Double Click
ourScript.py
and notice
how it **USES**
the Data from
ourData.py

Install Modules Using Pip

College of Scripting Music & Science

Open the Command Prompt



Command Prompt



A screenshot of a Microsoft Windows Command Prompt window. The title bar says "Command Prompt". The window content shows the text:
Microsoft Windows [Version 10.0.19041.746]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\energy>pip

A yellow arrow points from the text "Type the Word pip" to the word "pip" in the command line. A second yellow arrow points from the text "On your Keyboard press ENTER" to the word "ENTER".

Type the Word
pip

On your Keyboard press
ENTER

The Command Prompt opens.
We type the word **pip**
and then press **ENTER** on our keyboard.

If everything goes well, **pip** (**Preferred Installer Program**), will open in your command prompt.

pip

```
C:\ Command Prompt
Microsoft Windows [Version 10.0.19041.746]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\energy>pip

Usage:
  pip <command> [options]

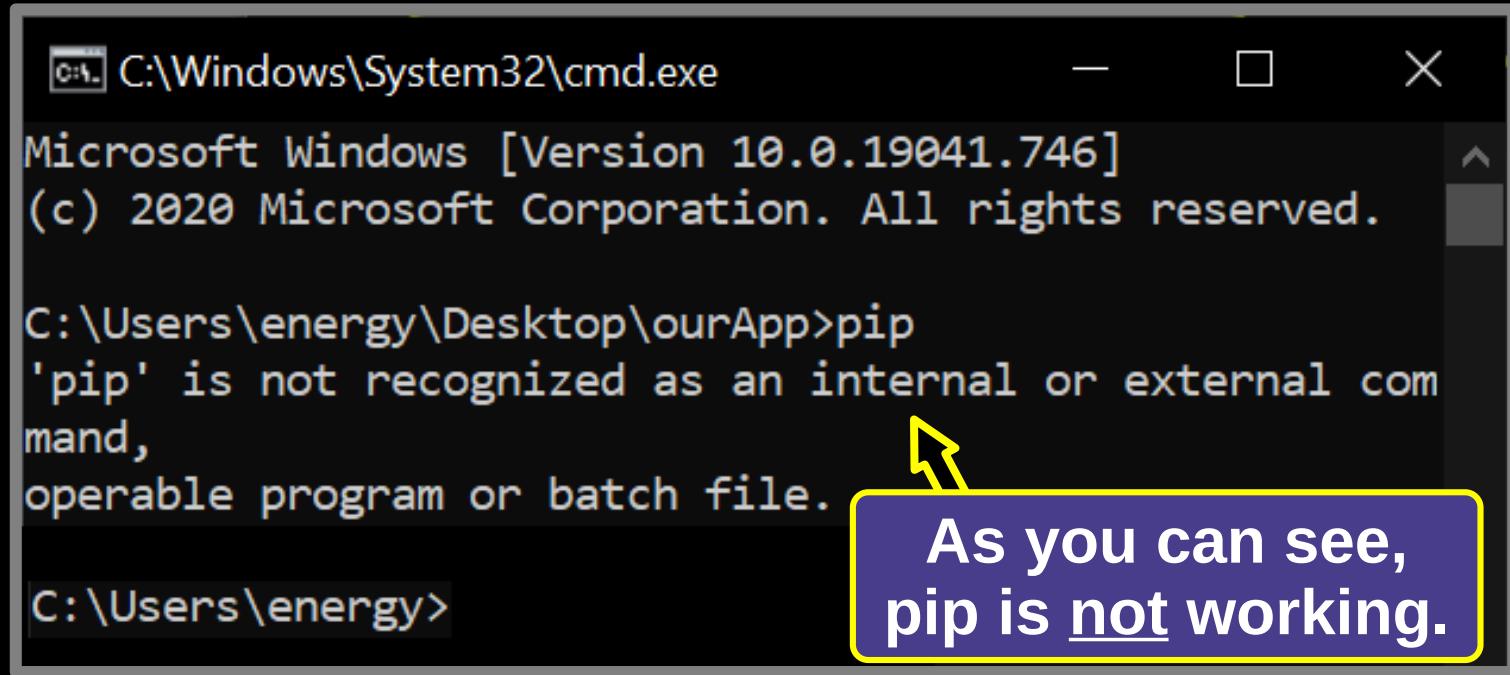
Commands:
  install                  Install packages.
  download                Download packages.
  uninstall               Uninstall packages.
  freeze                  Output installed packages in requirements format.
  list                     List installed packages.
  show                     Show information about installed packages.
  check                   Verify installed packages have compatible dependencies.
  config                  Manage local and global configuration.
  search                  Search PyPI for packages.
  cache                   Inspect and manage pip's wheel cache.
  wheel                   Build wheels from your requirements.
  hash                    Compute hashes of package archives.
  completion              A helper command used for command completion.
```

pip is the python module installer.

We will use pip to install a module named **pyinstaller**

pyinstaller is a module we use to make our **.exe** file!

What if pip is NOT working?



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.746]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\energy\Desktop\ourApp>pip
'pip' is not recognized as an internal or external com-
mand,
operable program or batch file.

C:\Users\energy>
```

As you can see,
pip is not working.

When we first installed Python,
the CHECKBOX should have been **checked**
for Add Python 3.9 to PATH!



Add Python 3.9 to PATH

There are multiple ways to fix this issue.

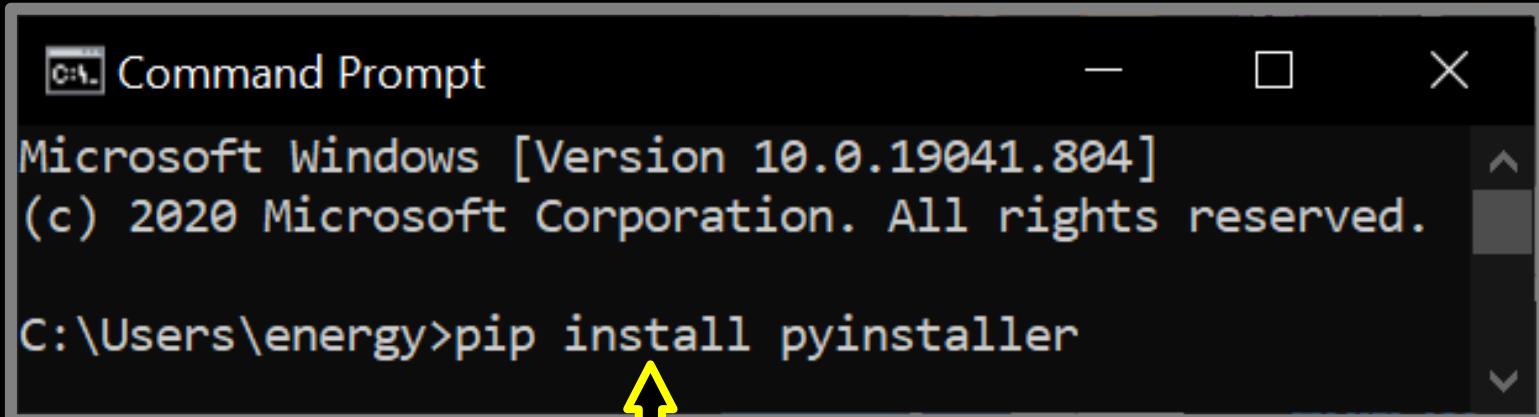
An easy way is to **uninstall** Python and **reinstall**,
this time with the **Path** checkbox **marked**.

Or we could change the environment variable.

Once pip is working, move to the next page =>

MAKE A Single File .EXE FILE with PyInstaller

Install PyInstaller



```
Command Prompt
Microsoft Windows [Version 10.0.19041.804]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\energy>pip install pyinstaller
```

Type these words
pip install pyInstaller

Keyboard press
ENTER

In our Command Prompt we type the words
pip install pyinstaller
and then press the **enter button** on our keyboard.

PyInstaller is an excellent module to make EXE files from our Python scripts!

PyInstaller Installed Successfully

Command Prompt

```
y\appdata\local\programs\python\python39\lib\site-packages  
(from pyinstaller) (49.2.1)  
Requirement already satisfied: pefile>=2017.8.1 in c:\users  
\energy\appdata\local\programs\python\python39\lib\site-pac  
kages (from pyinstaller) (2019.4.18)  
Requirement already satisfied: future in c:\users\energy\ap  
pdata\local\programs\python\python39\lib\site-packages (fro  
m pefile>=2017.8.1->pyinstaller) (0.18.2)  
Installing collected packages: pyinstaller  
Successfully installed pyinstaller-4.2
```

C:\Users\energy>

Successfully
installed
pyInstaller

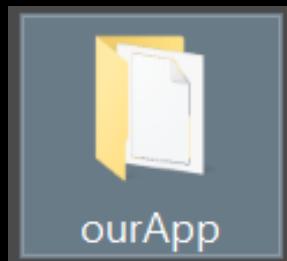
Good Job!!!

PyInstaller is now successfully installed!

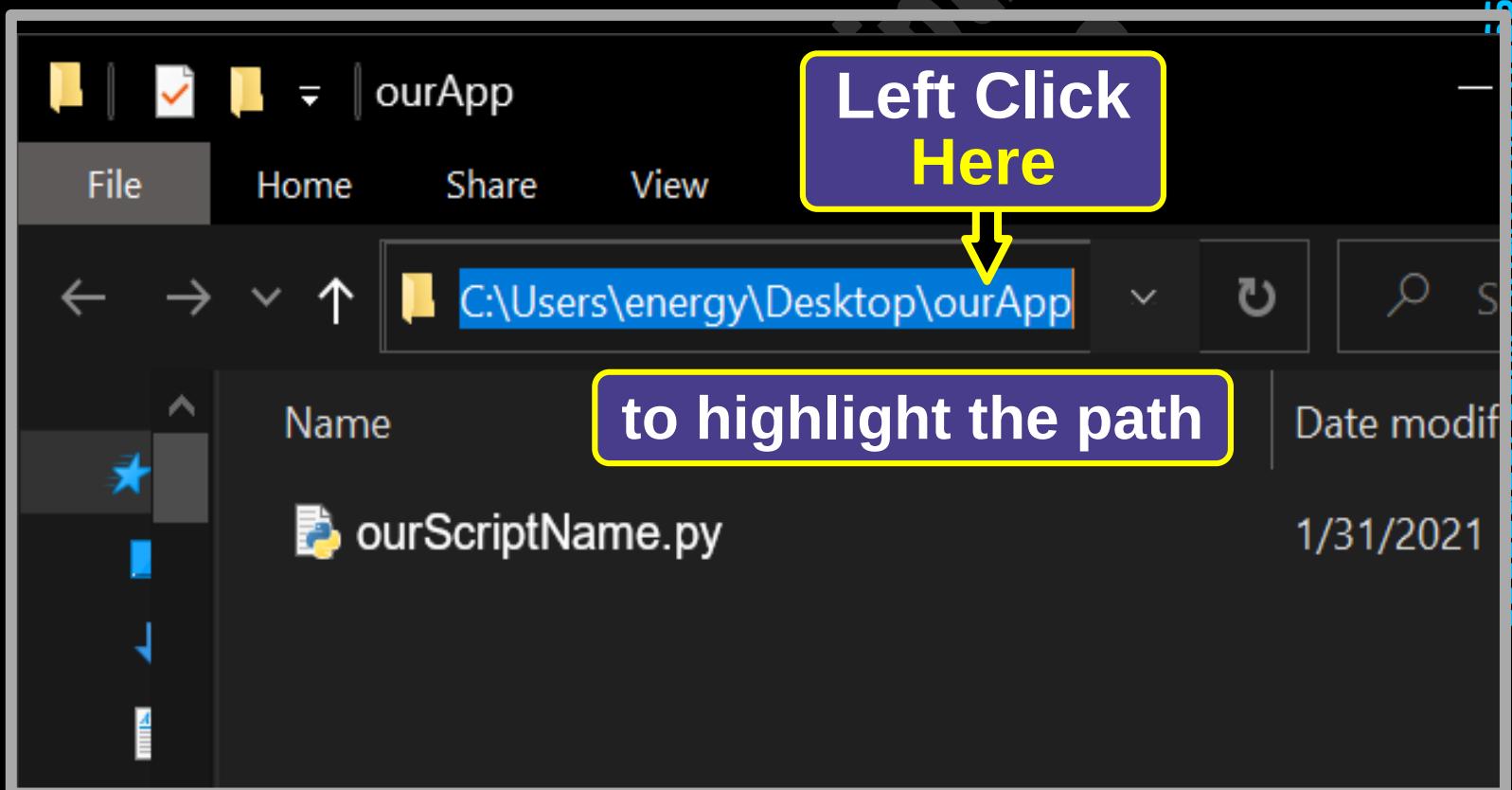
Now we can make our **.exe file** using **PyInstaller**,
as shown on the next pages.

How to Make a --onefile .exe file

We make a **Folder** on our **Desktop** named **ourApp**

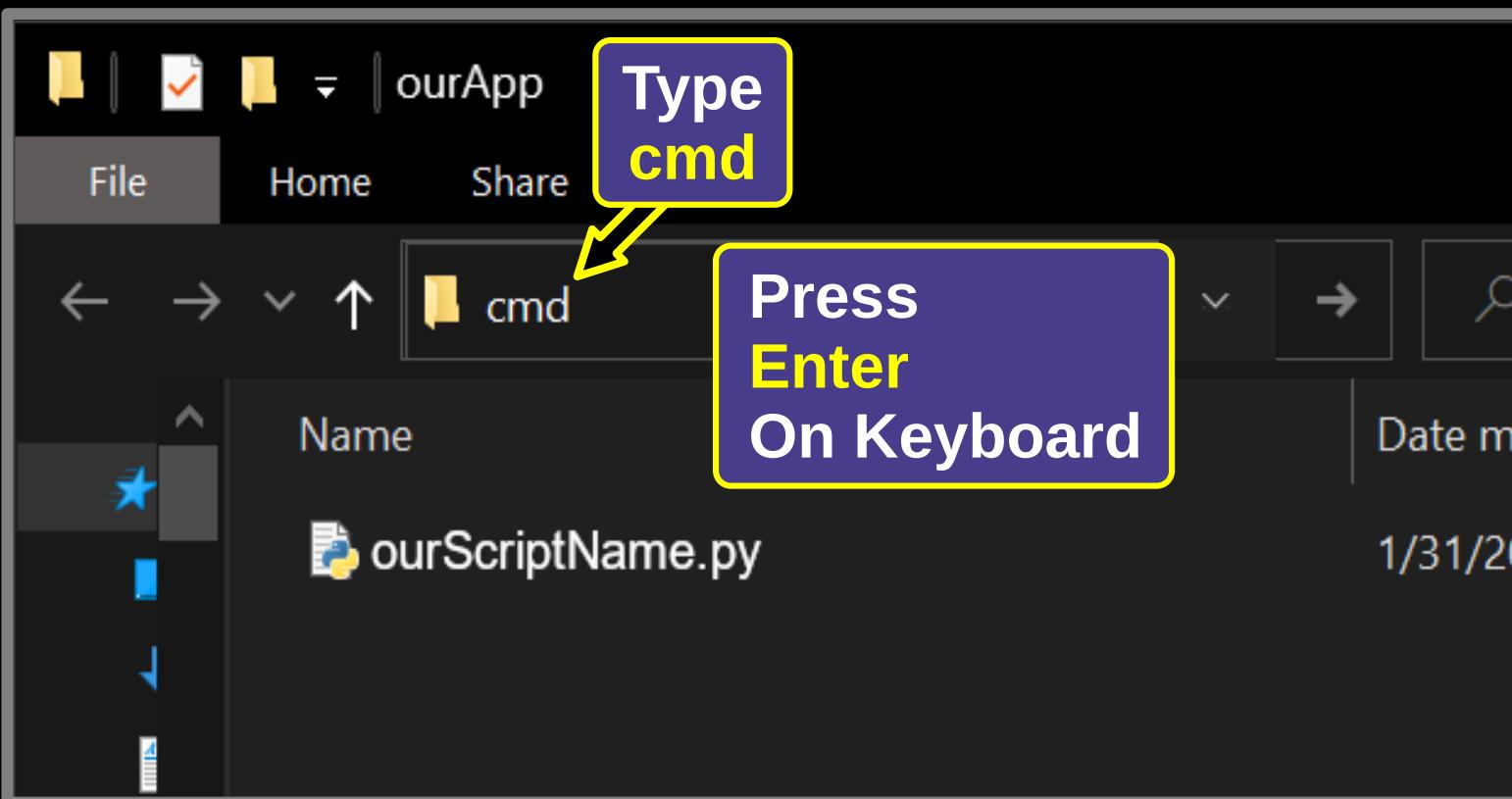


We place **any Python script** in **ourApp Folder**.



With the Path highlighted, we Type **cmd**, as shown on the next tutorial page.

continued from previous page



We type **cmd** in the path area, and then we press the **ENTER button** on our **keyboard**.

This will open **ourApp Folder** in the **Command Prompt**, with the **correct path**, as shown on the next tutorial page.

continued from previous page

As you can see, the Command Prompt opened.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.746]
(c) 2020 Microsoft Corporation. All rights reserved.
```

C:\Users\energy\Desktop\ourApp>

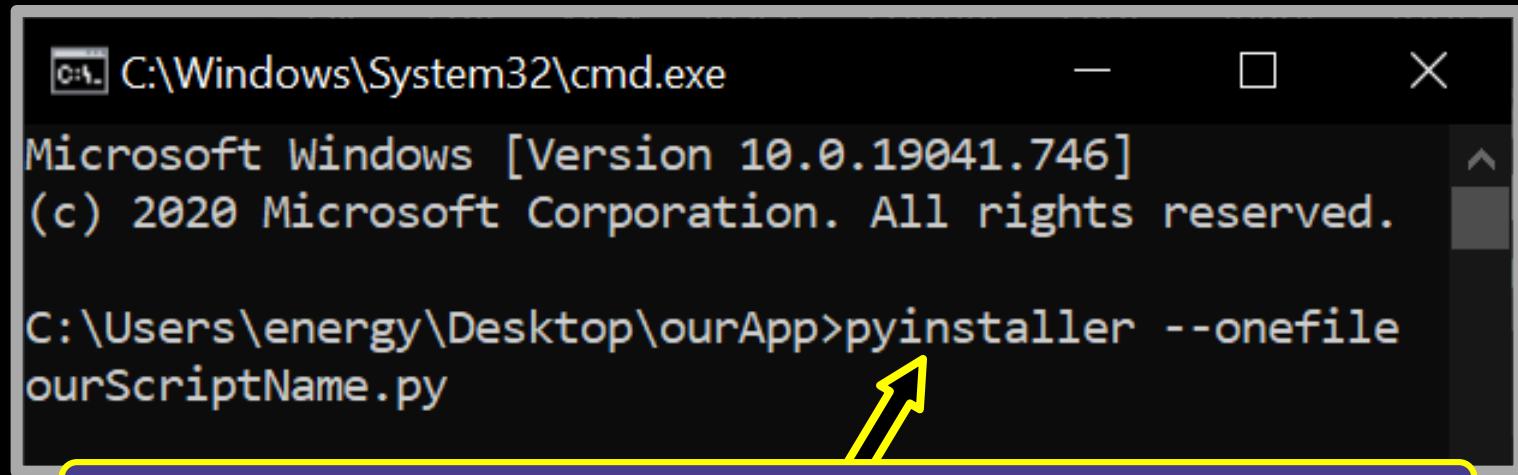


Notice that ourApp folder is chosen correctly for us.

We have ourApp Folder selected correctly as the chosen directory. We are READY!

NOTE: We could have alternatively chosen the directory manually, by typing,
`cd Desktop\ourApp`, and pressing ENTER.

continued from previous page



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.746]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\energy\Desktop\ourApp>pyinstaller --onefile
ourScriptName.py
```

Type these words
pyinstaller --onefile ourScriptName.py

Keyboard press
ENTER

This turns our python script
into an **EXECUTABLE FILE!**

Remember, **ourScriptName**
is the name of **YOUR** script!

The next pages teach you
how to open your First Application.
Good Job!

continued from previous page

```
C:\Windows\System32\cmd.exe
16149 INFO: Updating manifest in C:\Users\energy\Desktop\ourApp\build\ourScriptName\run.exe.r1iig_ag
16262 INFO: Updating resource type 24 name 1 language 0
16265 INFO: Appending archive to EXE C:\Users\energy\Desktop\ourApp\dist\ourScriptName.exe
16379 INFO: Building EXE from EXE-00.toc completed successfully.

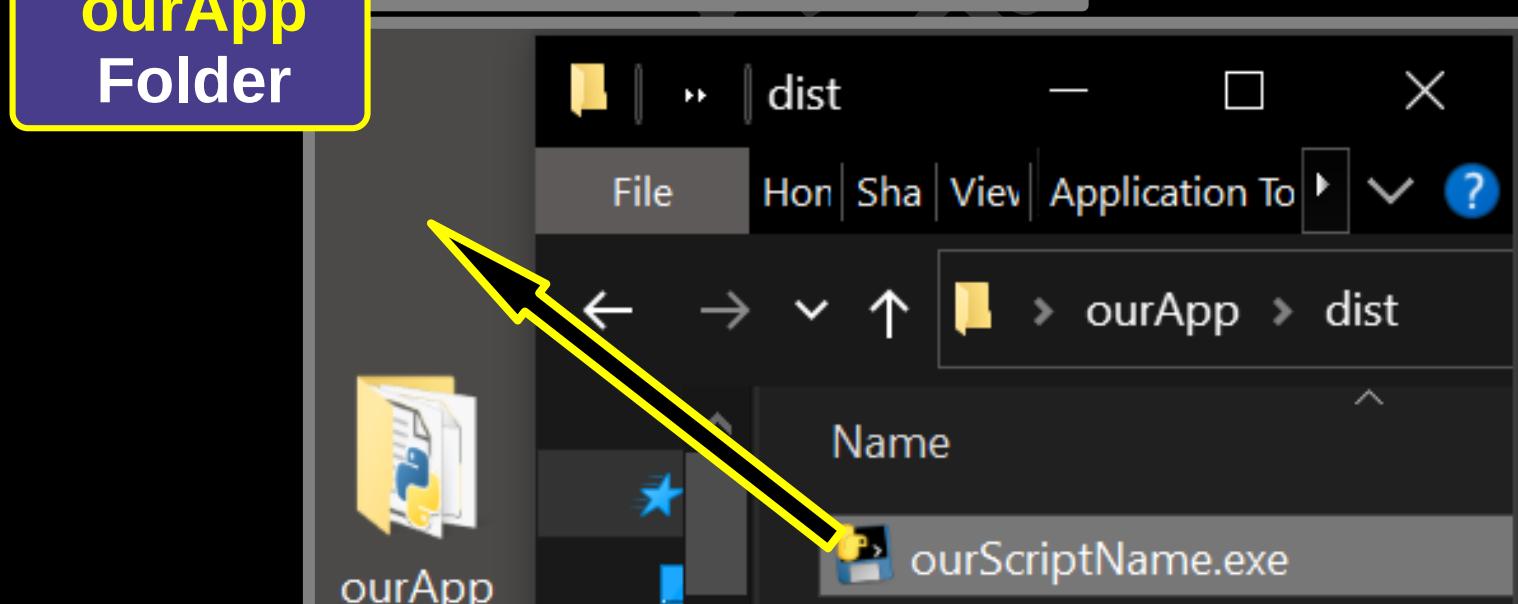
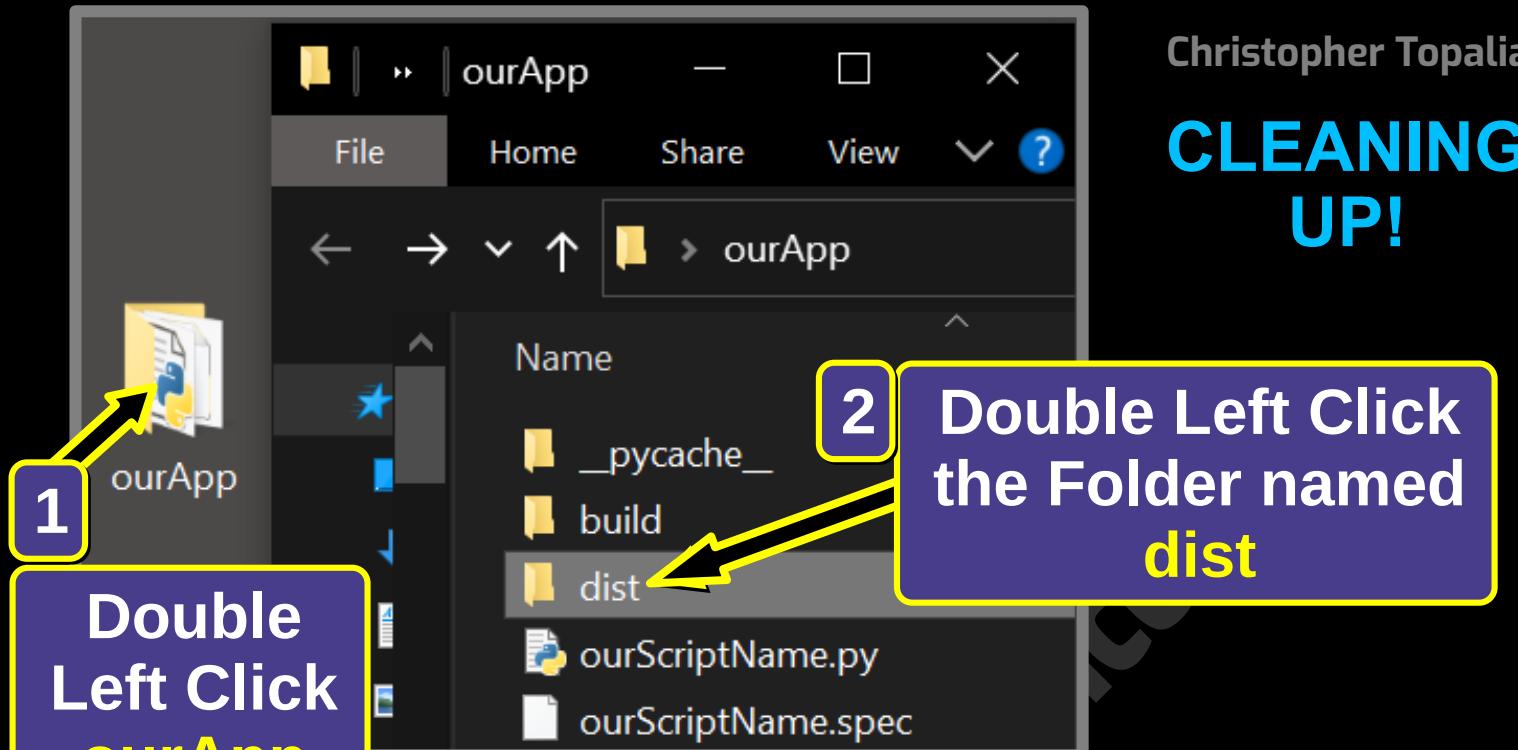
C:\Users\energy\Desktop\ourApp>
```

SUCCESS!!!

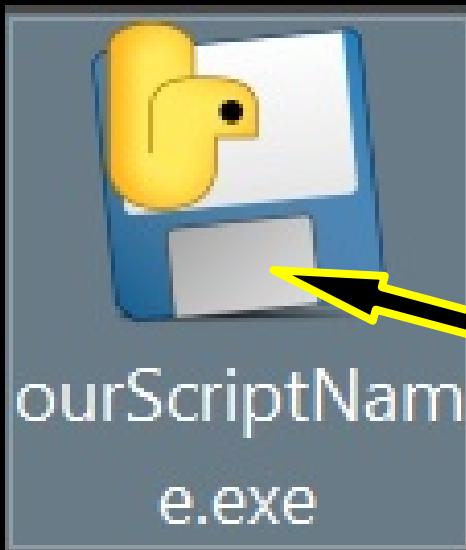
Our .exe file has been created!!!

We have successfully
created our .exe file!
We are Now App makers!

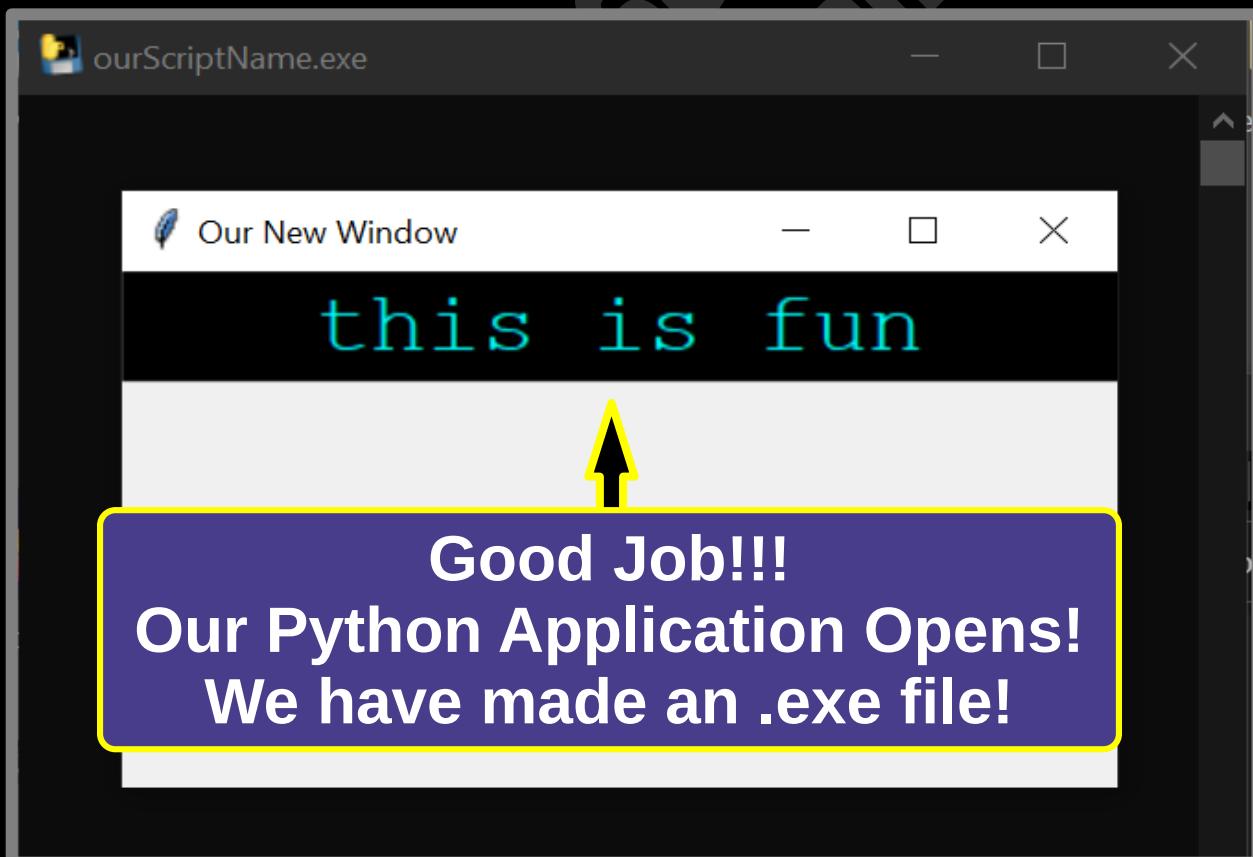
CLEANING UP!



OPENING OUR APP :-)



Double Left Click
the File named
ourScriptName.exe



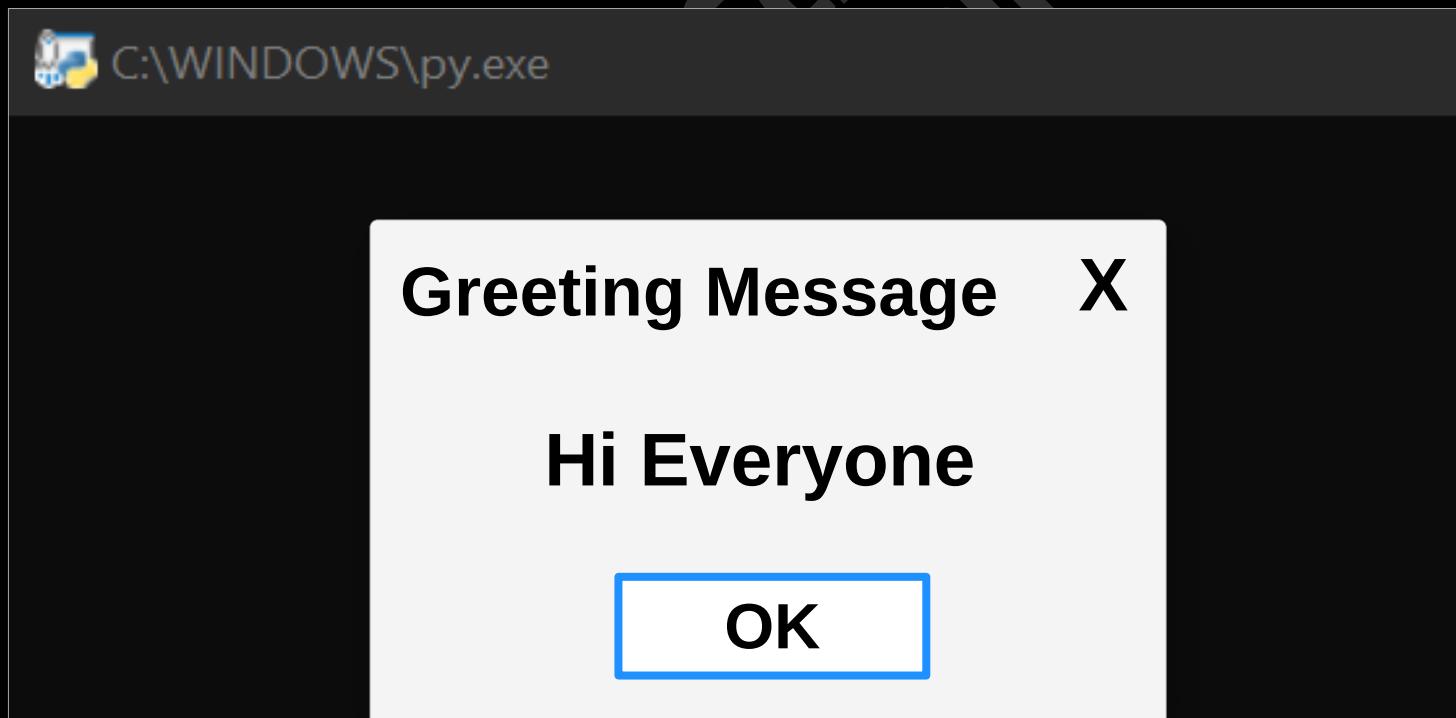
Good Job!!!
Our Python Application Opens!
We have made an .exe file!

Launch Our Apps Without Showing the Console Window

Run Scripts Without Launcher Appearing

If we save our scripts as **.py**
then the Python Launcher window
appears, when we run our scripts.

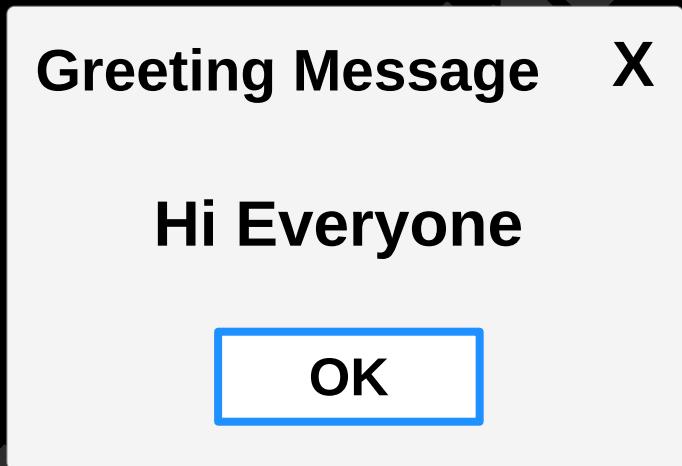
But, if we save our scripts as **.pyw**
then the Python Launcher Window
will NOT appear.



continued on next page =>

continued from previous page

Since we saved our script as
ourFirstScript.pyw
the Python Launcher Window will NOT
appear, when we double click our script!



This also works for when we make our script into an **.exe** file.

When we make the **.exe** file, we include the **.pyw** at the end of the command.

pyinstaller --onefile ourFirstScript.pyw

DATE & TIME

College of Scripting
Music & Science

Date using Message Box

Christopher Topalian

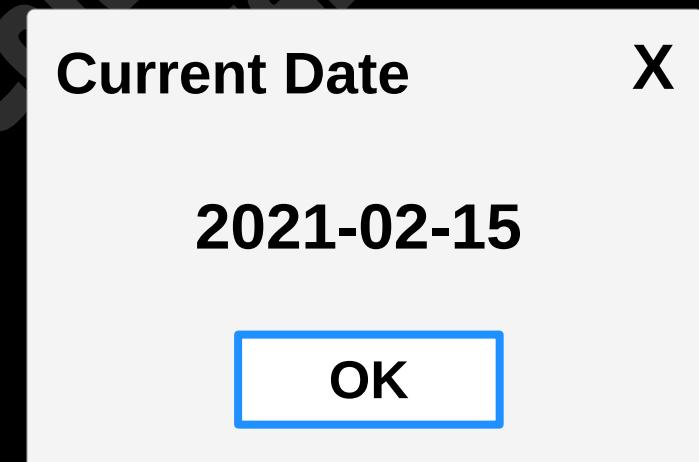
```
import ctypes  
from datetime import date
```

```
ourTitle = 'Current Date'
```

```
today = date.today()
```

```
print("Today's date is ", today)
```

```
ctypes.windll.user32.MessageBoxW(0,  
str(today), ourTitle, 0)
```



Date in a Window

Christopher Topalian

```
from datetime import date
```

```
from tkinter import*
```

```
ourWindow = Tk()
```

```
ourWindow.geometry('275x50+200+150')
```

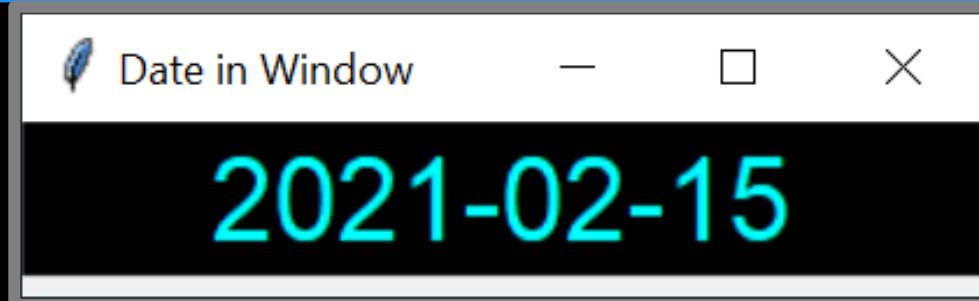
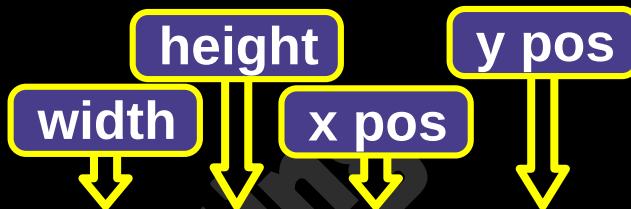
```
ourWindow.title('Date in Window')
```

```
today = date.today()
```

```
ourLabel1 = Label(ourWindow,
```

```
text = today, width = '200', font=("Arial", 25),  
fg = 'aqua', bg = 'black').pack()
```

```
ourWindow.mainloop()
```



Date using Message Box

Christopher Topalian

```
import ctypes  
from datetime import date
```

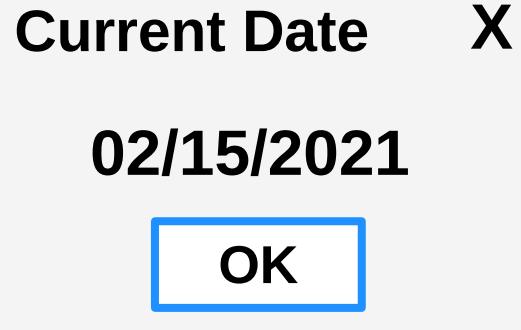
```
ourTitle = 'Current Date'
```

```
today = date.today()
```

```
formatted = today.strftime("%m/%d/%Y")
```

```
print("Today's date is ", today)
```

```
ctypes.windll.user32.MessageBoxW(0,  
str(formatted), ourTitle, 0)
```



COLLEGE OF SCRIPTING MUSIC & SCIENCE

We could also format it with a DASH divider

```
formatted = today.strftime("%m-%d-%Y")
```

02-15-2021

Date in Window, Month, Day, Year

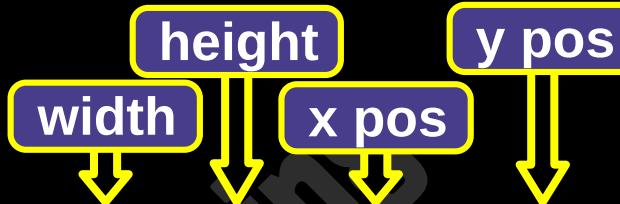
```
from datetime import date
```

```
from tkinter import*
```

```
ourWindow = Tk()
```

```
ourWindow.geometry('275x50+200+150')
```

```
ourWindow.title('Date in Window')
```



```
today = date.today()
```

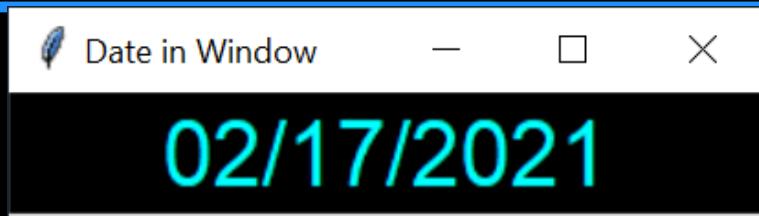
```
theTime = today.strftime("%m/%d/%Y")
```

```
ourLabel1 = Label(ourWindow,
```

```
text = theTime, width = '200', font=("Arial", 25),
```

```
fg = 'aqua', bg = 'black').pack()
```

```
ourWindow.mainloop()
```



Date in Window, YEAR

Christopher Topalian

```
from datetime import date
```

```
from tkinter import*
```

```
ourWindow = Tk()
```

```
ourWindow.geometry('275x50+200+150')
```

```
ourWindow.title('Date in Window')
```

```
today = date.today()
```

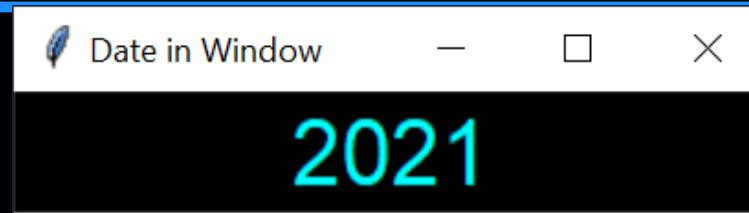
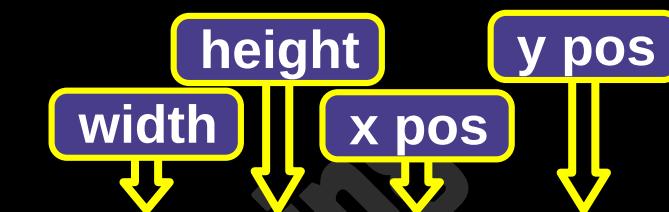
```
theTime = today.strftime("%Y")
```

```
ourLabel1 = Label(ourWindow,
```

```
text = theTime, width = '200', font=("Arial", 25),
```

```
fg = 'aqua', bg = 'black').pack()
```

```
ourWindow.mainloop()
```



Date in Window, Month, Day

Christopher Topalian

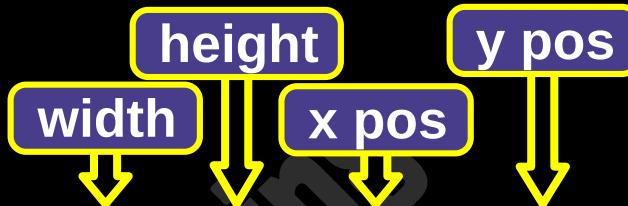
```
from datetime import date
```

```
from tkinter import*
```

```
ourWindow = Tk()
```

```
ourWindow.geometry('275x50+200+150')
```

```
ourWindow.title('Date in Window')
```

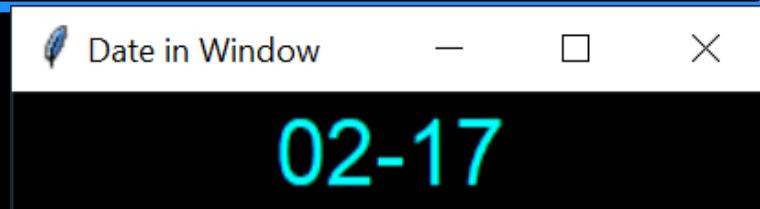


```
today = date.today()
```

```
theTime = today.strftime("%m-%d")
```

```
ourLabel1 = Label(ourWindow,  
text = theTime, width = '200', font=("Arial", 25),  
fg = 'aqua', bg = 'black').pack()
```

```
ourWindow.mainloop()
```



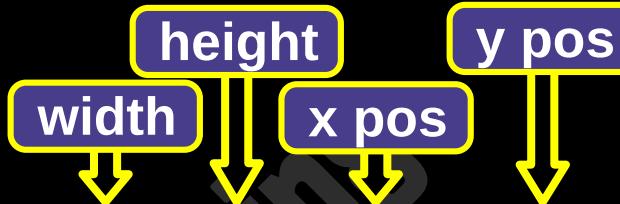
```
from datetime import date
```

```
from tkinter import*
```

```
ourWindow = Tk()
```

```
ourWindow.geometry('275x50+200+150')
```

```
ourWindow.title('Date in Window')
```



```
today = date.today()
```

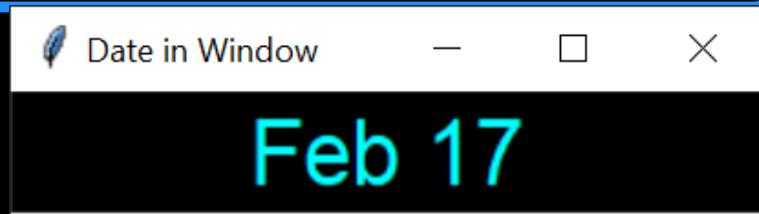
```
theTime = today.strftime("%b %d")
```

```
ourLabel1 = Label(ourWindow,
```

```
text = theTime, width = '200', font=("Arial", 25),
```

```
fg = 'aqua', bg = 'black').pack()
```

```
ourWindow.mainloop()
```



Date in a Window, MONTH

Christopher Topalian

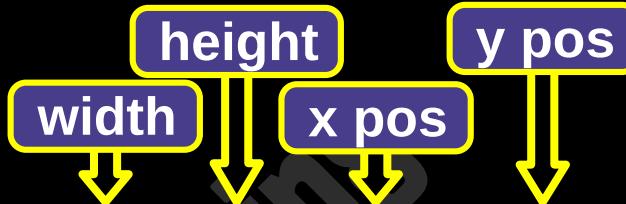
```
from datetime import date
```

```
from tkinter import*
```

```
ourWindow = Tk()
```

```
ourWindow.geometry('275x50+200+150')
```

```
ourWindow.title('Date in Window')
```

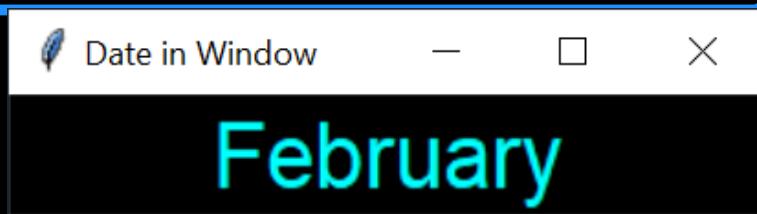


```
today = date.today()
```

```
theTime = today.strftime("%B")
```

```
ourLabel1 = Label(ourWindow,  
text = theTime, width = '200', font=("Arial", 25),  
fg = 'aqua', bg = 'black').pack()
```

```
ourWindow.mainloop()
```



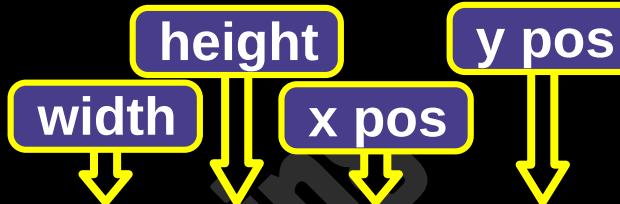
Time in Window as H, M, S, AM/PM

Christopher Topalian

```
from datetime import*
```

```
from tkinter import*
```

```
ourWindow = Tk()
```



```
ourWindow.geometry('275x50+200+150')
```

```
ourWindow.title('Time in Window')
```

```
today = datetime.today()
```

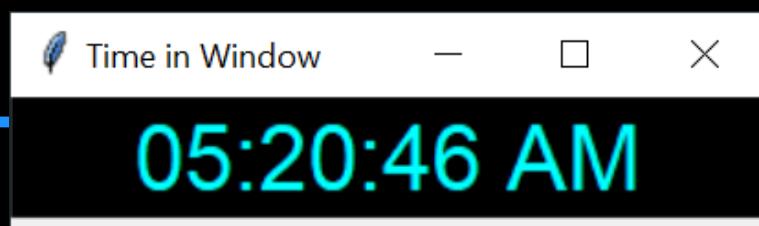
```
theTime = today.strftime("%H:%M:%S %p")
```

```
ourLabel1 = Label(ourWindow,
```

```
text = theTime, width = '200', font=("Arial", 25),
```

```
fg = 'aqua', bg = 'black').pack()
```

```
ourWindow.mainloop()
```

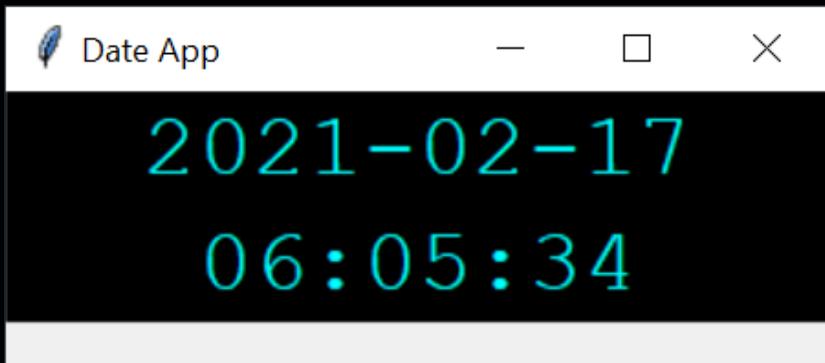


Hours, Minutes, Seconds AM or PM

Date and Time in a Window

Christopher Topalian

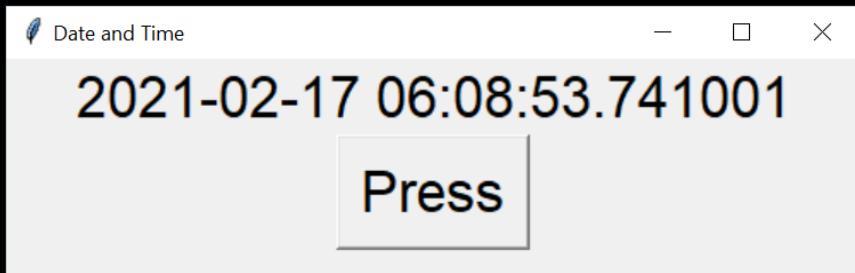
```
from tkinter import *
from datetime import*
def theDate():
    today = date.today()
    return today
def currentTime():
    now = datetime.now()
    timeNow = now.strftime("%H:%M:%S")
    return timeNow
ourWindow = Tk()
ourWindow.geometry('300x100+300+200')
ourWindow.title('Date App')
theDate = theDate()
theTime = currentTime()
ourLabel1 = Label(ourWindow,
text = theDate, width = '200', font=("Courier", 25),
fg = 'aqua', bg = 'black').pack()
ourLabel2 = Label(ourWindow,
text = theTime, width = '200', font=("Courier", 25),
fg = 'aqua', bg = 'black').pack()
ourWindow.mainloop()
```



Date and Time in Window with Button

Christopher Topalian

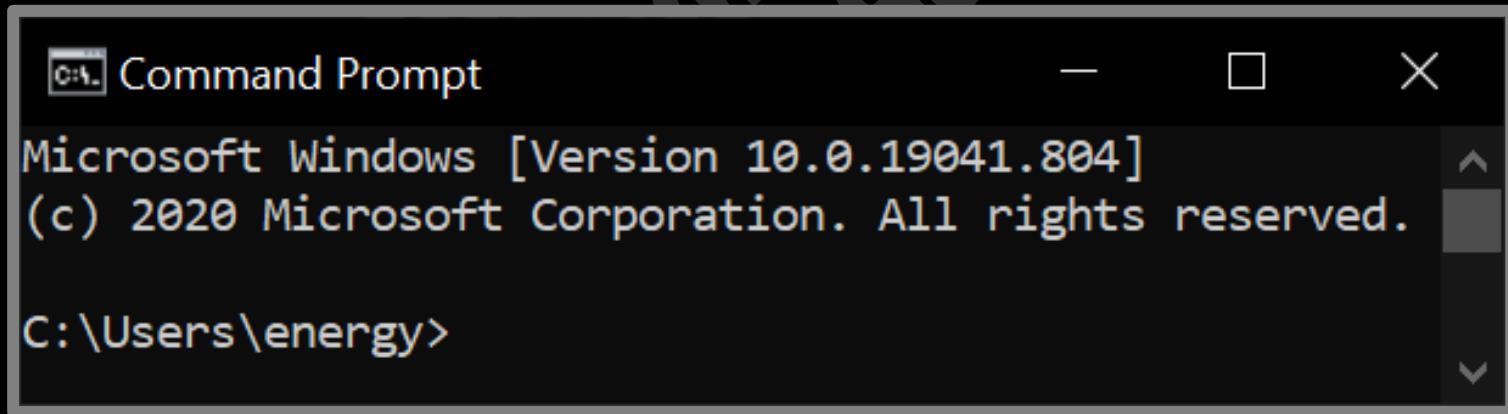
```
from tkinter import*
from datetime import*
count = 0
def ourCounter():
    global count
    count = datetime.now()
    ourLabel1.configure(text = f'{count}')
ourWindow = Tk()
ourWindow.geometry('500x125+200+150')
ourWindow.title("Date and Time")
ourLabel1 = Label(ourWindow, font=("Arial", 25))
ourLabel1.pack()
ourButton = Button(ourWindow, text="Press",
                   command=ourCounter, font=("Arial", 25))
ourButton.pack()
ourWindow.mainloop()
```



HOW TO OPEN COMMAND PROMPT

Shortcut to Open Command Prompt

Press Windows Key
Type CMD
Press Enter



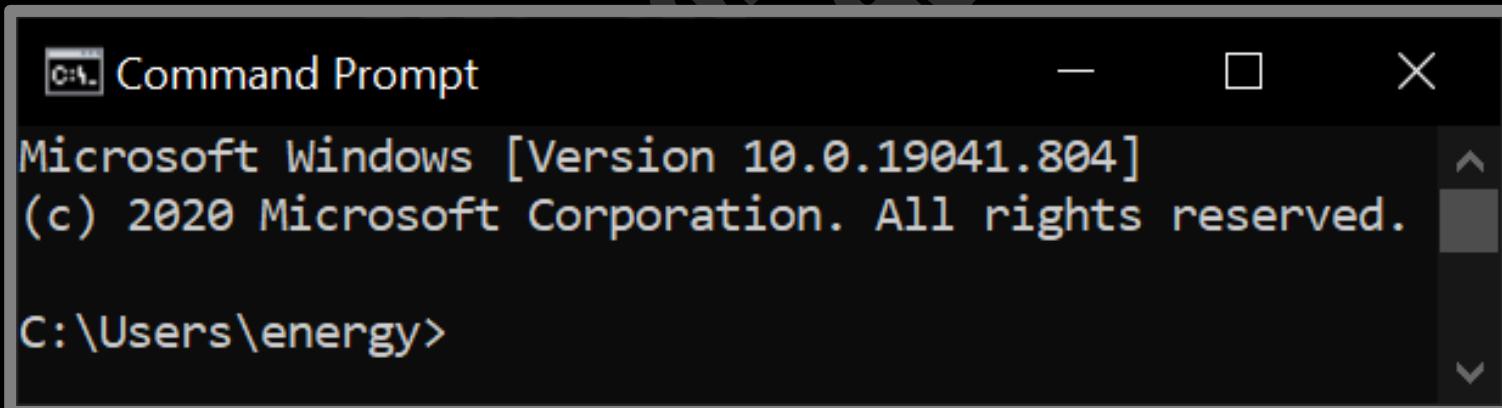
We install modules using this Command Prompt.

Shortcut to Open Command Prompt

Press Windows Key + R

Type CMD

Press Enter



We install modules using this Command Prompt.

Python Package Manager

Command Prompt

Microsoft Windows [Version 10.0.19041.804]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\energy>pip

We Type
the word
pip to see
what
commands
it offers

pip

Command Prompt
C:\Users\energy>pip
Usage:
 pip <command> [options]

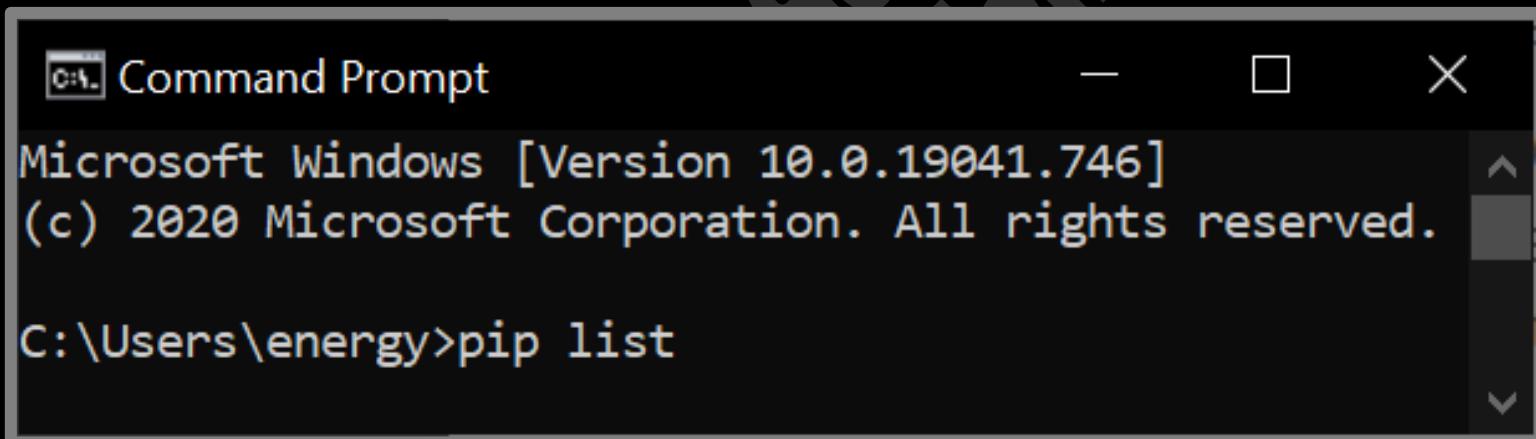
Commands:
 install
 download
 uninstall
 freeze
 list
 show
 check
 config
 search
 cache
 wheel
 hash
 completion
 debug
 help

Shows
Often
Used
Commands

Install packages.
Download packages.
Uninstall packages.
Output installed packages in requirements format.
List installed packages.
Show information about installed packages.
Verify installed packages have compatible dependencies.
Manage local and global configuration.
Search PyPI for packages.
Inspect and manage pip's wheel cache.
Build wheels from your requirements.
Compute hashes of package archives.
A helper command used for command completion.
Show information useful for debugging.
Show help for commands.

General Options:

Find Out What Modules We Have Installed



A screenshot of a Microsoft Windows Command Prompt window. The window title is "Command Prompt". The text inside the window shows the system information: "Microsoft Windows [Version 10.0.19041.746]" and "(c) 2020 Microsoft Corporation. All rights reserved.". Below that, the command "C:\Users\energy>pip list" is entered.

pip list



Command Prompt

- X

Microsoft Windows [Version 10.0.19041.746]

(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\energy>pip list

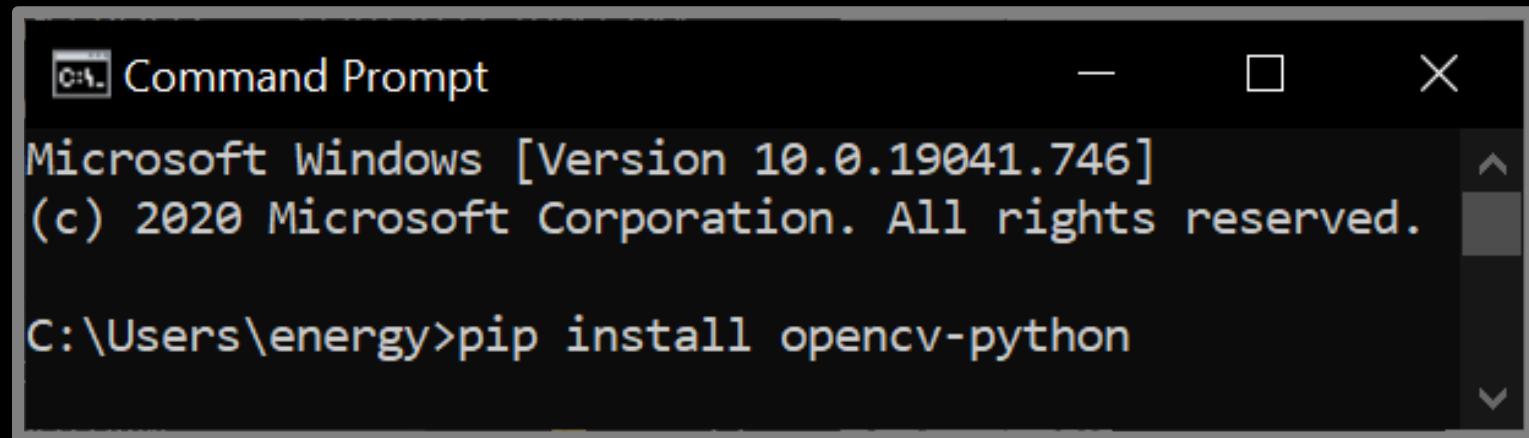
Package	Version
<hr/>	
altgraph	0.17
astroid	2.4.2
cachetools	4.2.1
certifi	2020.12.5
colorama	0.4.4
future	0.18.2
h11	0.12.0
h2	3.2.0
hpack	3.0.0
httpcore	0.12.3
httpx	0.16.1
hyperframe	5.2.0
idna	3.1
isort	5.7.0
lazy-object-proxy	1.4.3
mccabe	0.6.1
numpy	1.20.1
pandas	1.2.2
pefile	2019.4.18
Pillow	8.1.0
pip	21.0.1
py3exe	0.19.2.0

These
are
modules
that
I have
Installed

CAMERA

College of Scripting
Music & Science

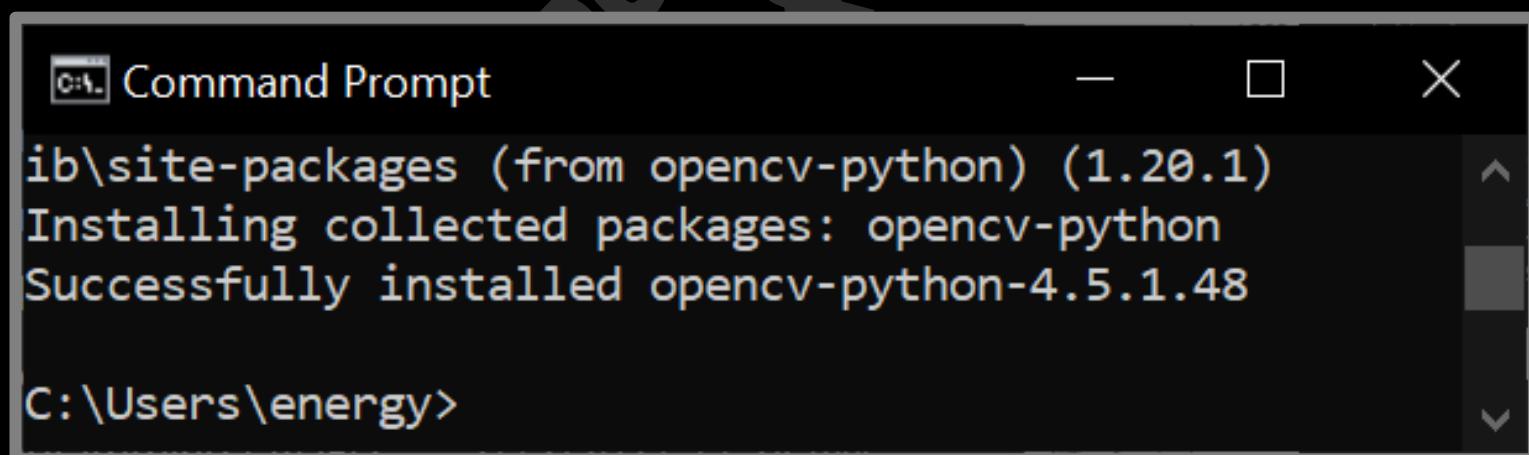
Computer Vision using OpenCv



```
Command Prompt
Microsoft Windows [Version 10.0.19041.746]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\energy>pip install opencv-python
```

pip install opencv-python



```
Command Prompt
ib\site-packages (from opencv-python) (1.20.1)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.5.1.48

C:\Users\energy>
```

Camera OPEN our Webcam

```
import cv2
```

```
cv2.namedWindow("Howdy")
theVideo = cv2.VideoCapture(0)
```

```
rval, theFrame = theVideo.read()
```

```
while rval:
```

```
    cv2.imshow("Howdy", theFrame)
    rval, theFrame = theVideo.read()
    key = cv2.waitKey(22)
```

This script opens a new window with our Web Camera as the video source!

```
import cv2
cv2.namedWindow("Howdy")
theVideo = cv2.VideoCapture(0)

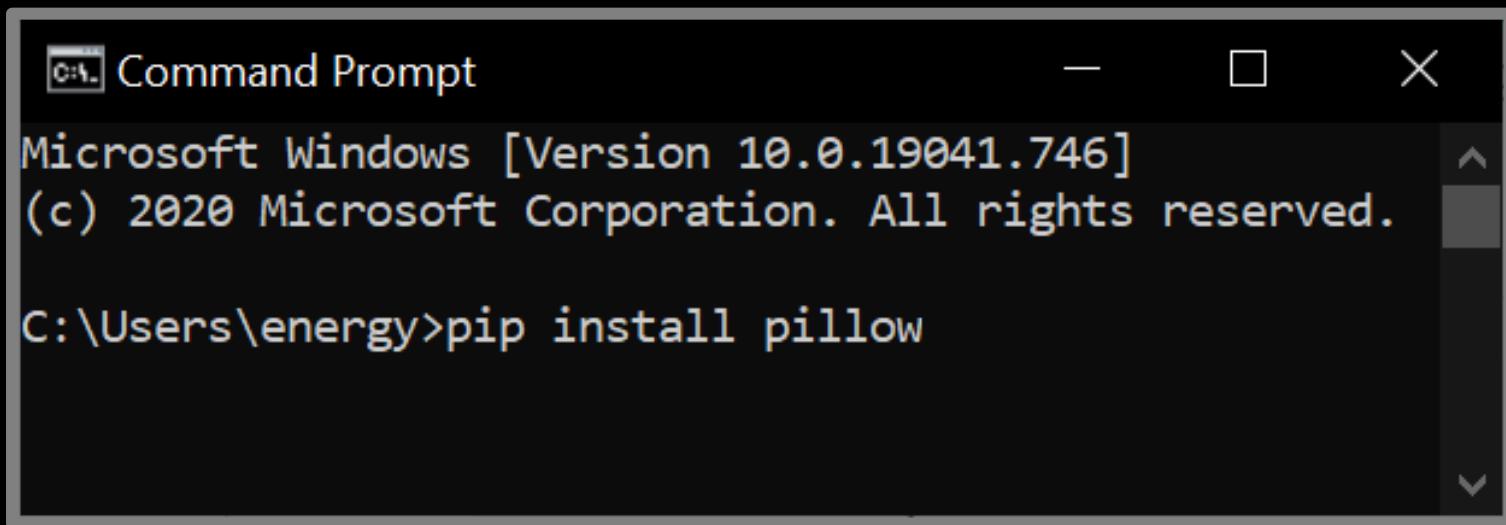
if theVideo.isOpened():
    rval, theFrame = theVideo.read()
else:
    rval = False

while rval:
    cv2.imshow("Howdy", theFrame)
    rval, theFrame = theVideo.read()
    key = cv2.waitKey(22)
    if key == 27:
        break
cv2.destroyAllWindows()
```

IMAGES

College of Scripting
Music & Science

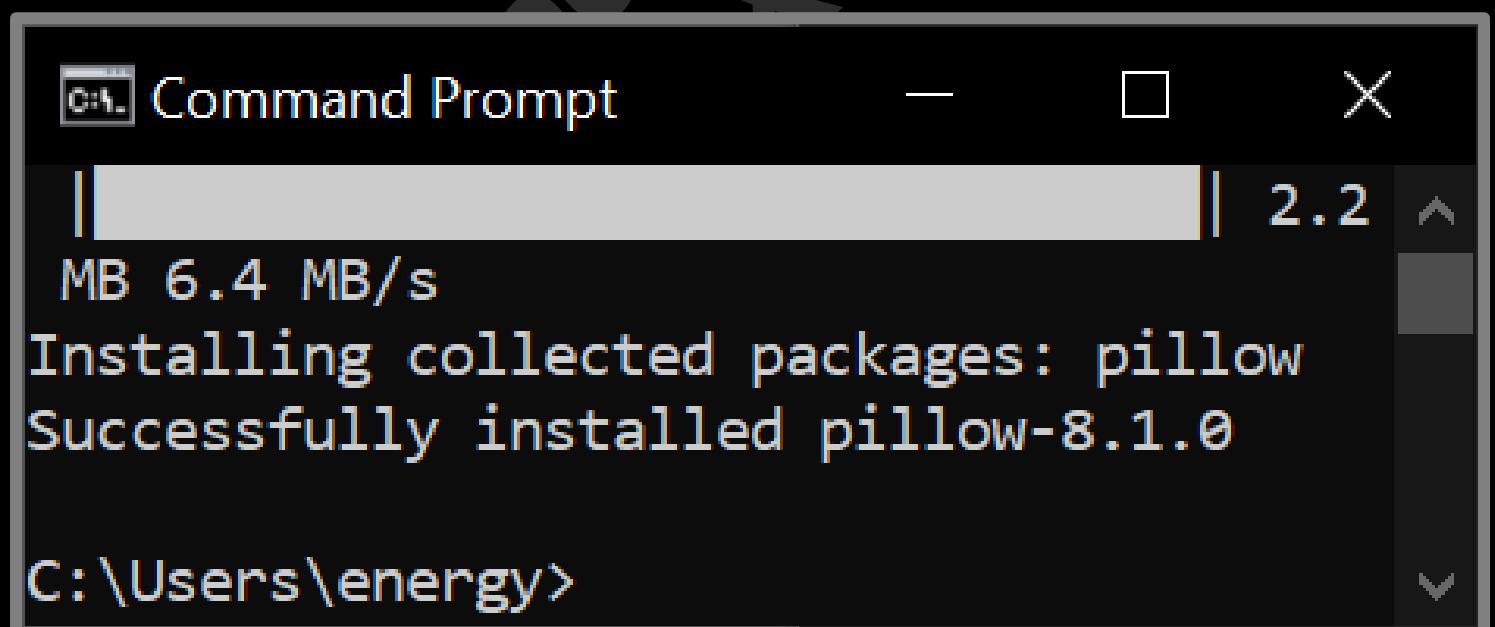
Python Image Library



```
Command Prompt
Microsoft Windows [Version 10.0.19041.746]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\energy>pip install pillow
```

pip install pillow



```
Command Prompt
| 2.2 MB 6.4 MB/s
Installing collected packages: pillow
Successfully installed pillow-8.1.0

C:\Users\energy>
```

tkinter open image, using pillow

```
import tkinter as tk  
from PIL import ImageTk, Image
```

```
ourWindow = tk.Tk()  
ourWindow.geometry("670x525+100+100")  
ourWindow.title("Our Image App")  
ourWindow.configure(background='black')
```

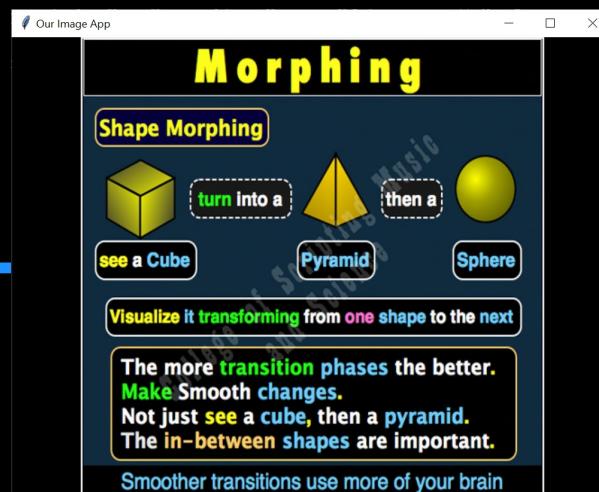
```
ourTexture = Image.open("theTexture.png")  
ourTexture = ImageTk.PhotoImage(ourTexture)
```

```
ourLabel = tk.Label(ourWindow, image = ourTexture)
```

```
ourLabel.pack(side = "top")
```

```
ourWindow.mainloop()
```

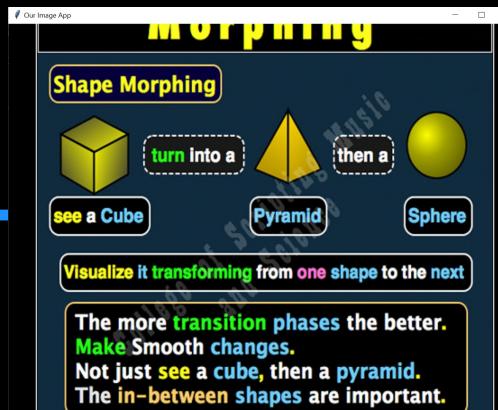
This script opens our texture in our Window



tkinter resize image, using pillow

```
import tkinter as tk  
from PIL import ImageTk, Image  
  
ourWindow = tk.Tk()  
ourWindow.geometry("900x900+100+100")  
ourWindow.title("Our Image App")  
ourWindow.configure(background='black')  
  
ourTexture = Image.open("theTexture.png")  
ourTexture = ourTexture.resize((800,800))  
ourTexture = ImageTk.PhotoImage(ourTexture)  
  
ourLabel = tk.Label(ourWindow, image = ourTexture)  
  
ourLabel.pack(side = "top")  
  
ourWindow.mainloop()
```

This script opens
our resized texture
in our Window



tkinter position image, using pillow

```
import tkinter as tk
```

```
from PIL import ImageTk, Image
```

```
ourWindow = tk.Tk()
```

```
ourWindow.geometry("900x900+100+100")
```

```
ourWindow.title("Our Image App")
```

```
ourWindow.configure(background='black')
```

```
ourTexture = Image.open("theTexture.png")
```

```
ourTexture = ourTexture.resize((700,700))
```

```
ourTexture = ImageTk.PhotoImage(ourTexture)
```

```
ourLabel = tk.Label(ourWindow, image = ourTexture)
```

```
ourLabel.place(x = 0, y = 0)
```

```
ourWindow.mainloop()
```

This script opens our positioned texture in our Window



tkinter rotate image, using pillow

```
import tkinter as tk  
from PIL import ImageTk, Image
```

```
ourWindow = tk.Tk()  
ourWindow.geometry("650x550+100+100")  
ourWindow.title("Our Image App")  
ourWindow.configure(background='black')
```

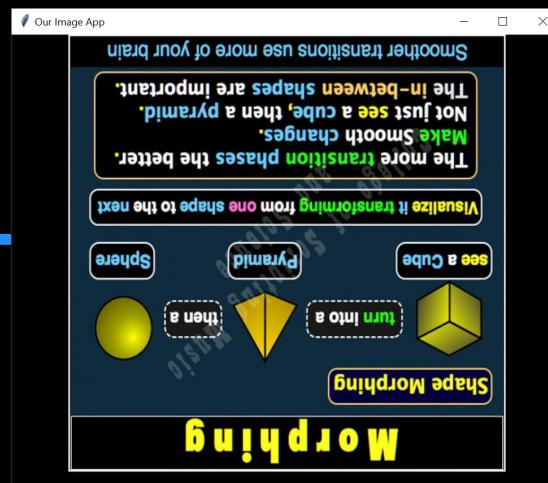
```
ourTexture = Image.open("theTexture.png")  
ourTexture = ourTexture.rotate((180))  
ourTexture = ImageTk.PhotoImage(ourTexture)
```

```
ourLabel = tk.Label(ourWindow, image = ourTexture)
```

```
ourLabel.pack(side = "top")
```

```
ourWindow.mainloop()
```

This script opens our rotated texture in our Window

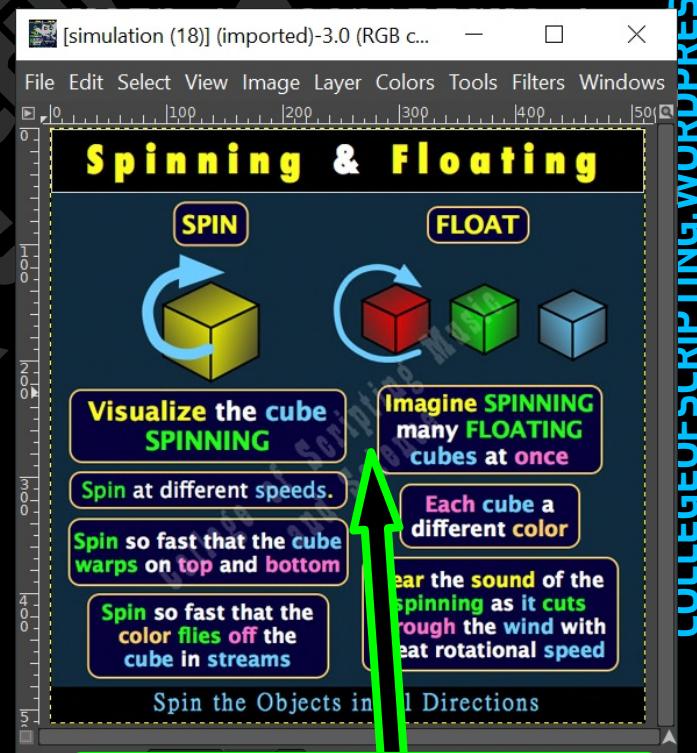
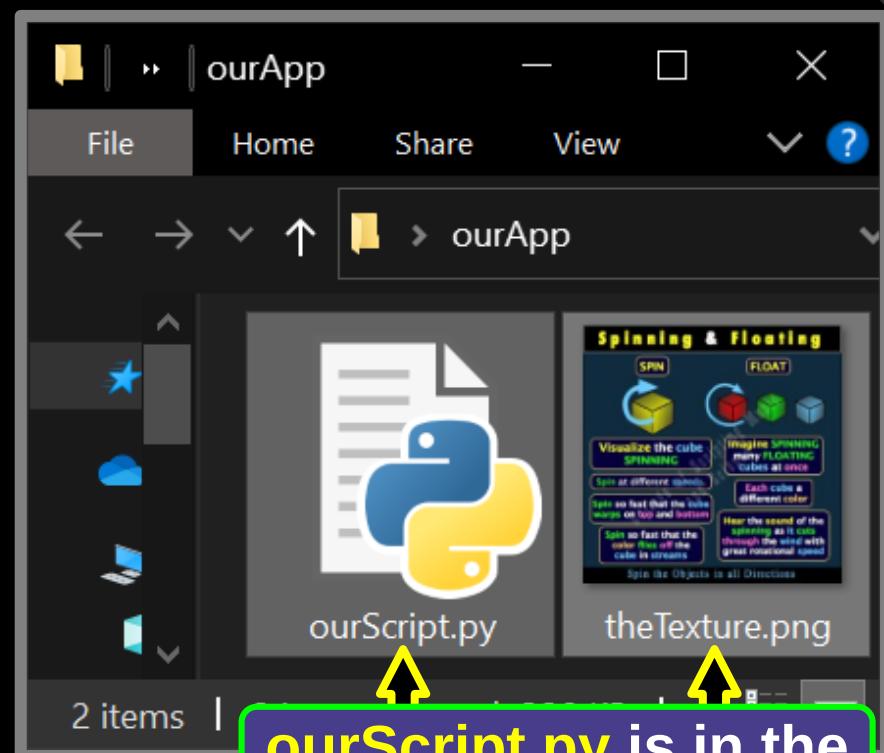


open image using pillow

from PIL import Image

```
ourTexture = Image.open("theTexture.png")
```

```
ourTexture.show()
```

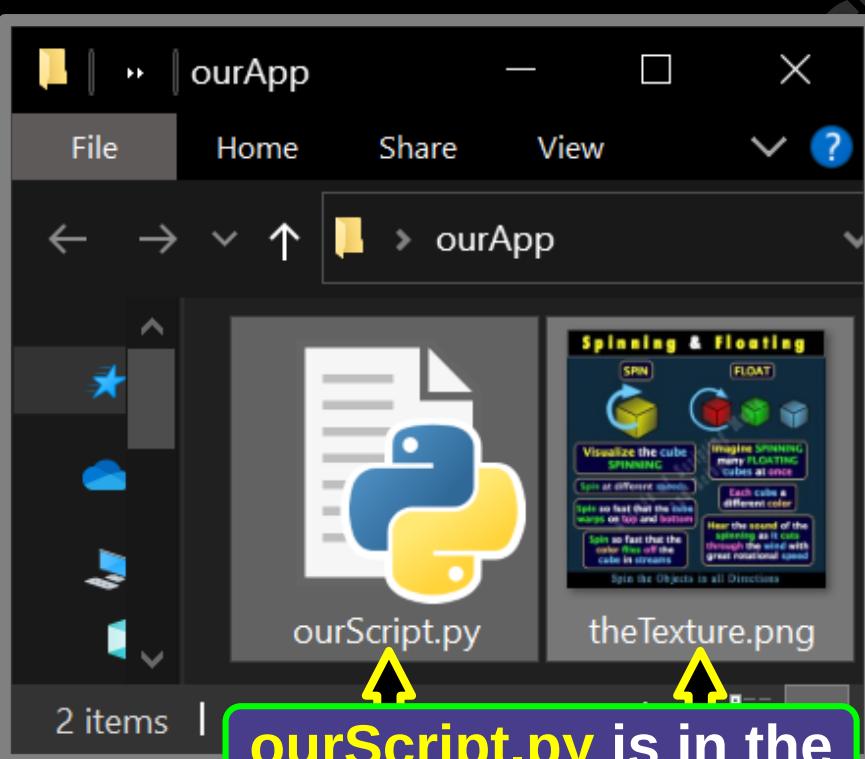


rotate image, using pillow

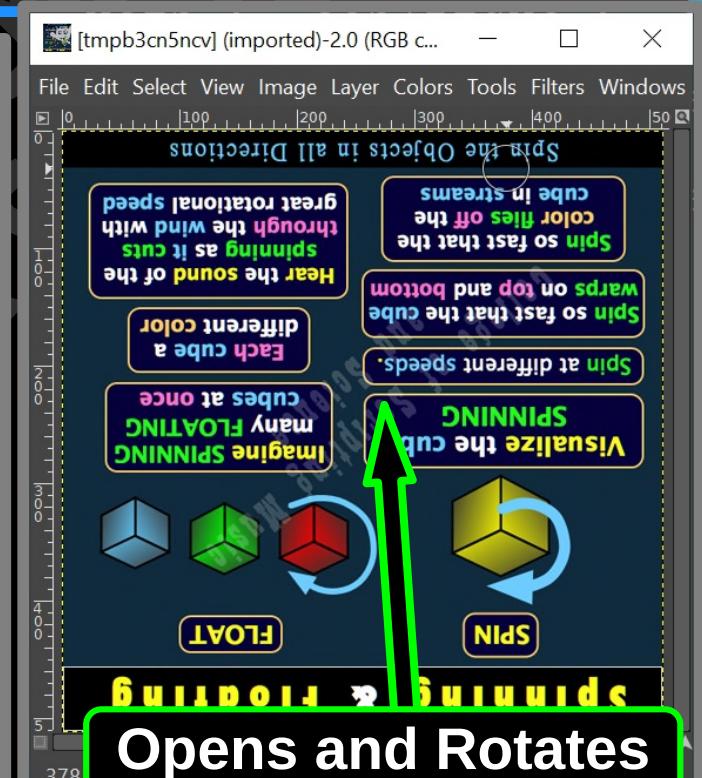
from PIL import Image

```
ourTexture = Image.open("theTexture.png")
```

```
ourTexture.rotate(180).show()
```



ourScript.py is in the
SAME FOLDER as
theTexture.png



Opens and Rotates
our Texture
using our
Default Image App

SCREEN SHOT

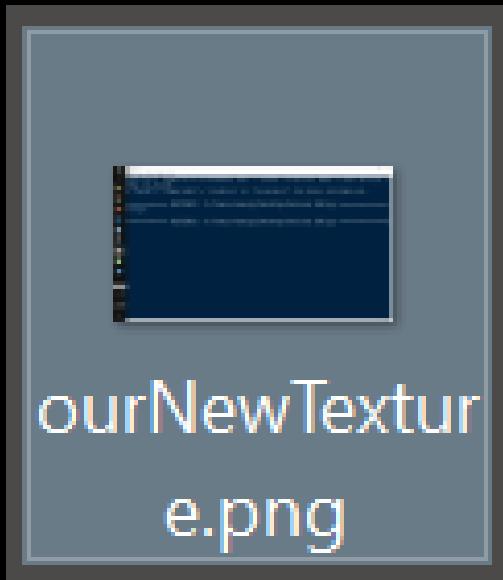
College of Scripting Music & Science

Screenshot Full Screen using pillow

```
from PIL import ImageGrab
```

```
ourImage = ImageGrab.grab()
```

```
ourImage.save('ourNewTexture.png')
```



We use the grab function to take this screen shot and save it as a .png file named ourTexture

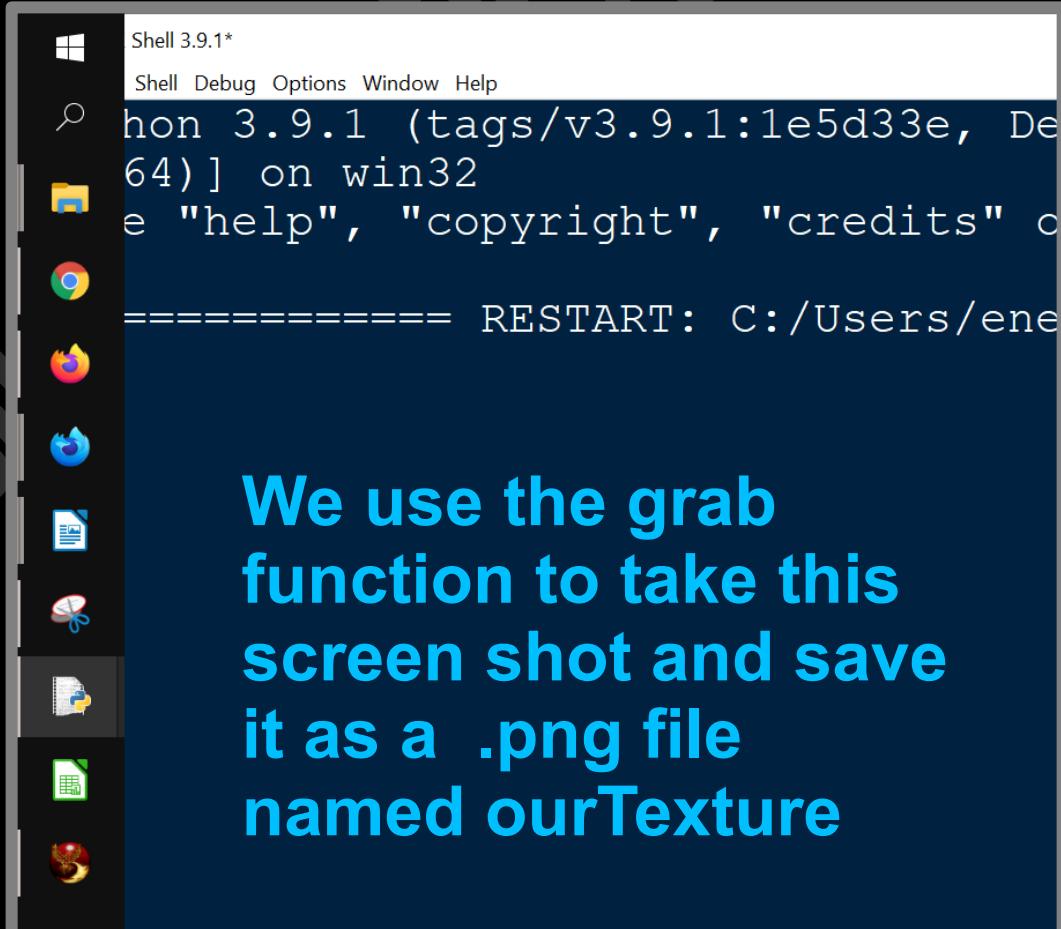
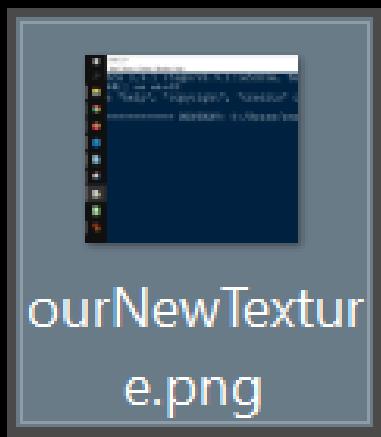
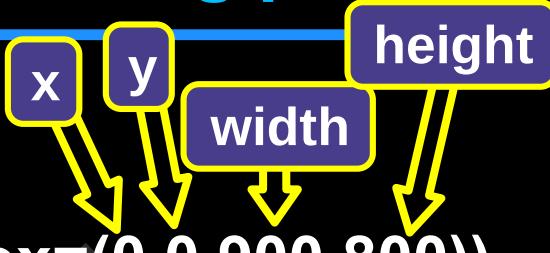
The texture is created in the location folder where this script is activated.

Screenshot Part of Screen using pillow

```
from PIL import ImageGrab
```

```
ourImage = ImageGrab.grab(bbox=(0,0,900,800))
```

```
ourImage.save('ourNewTexture.png')
```

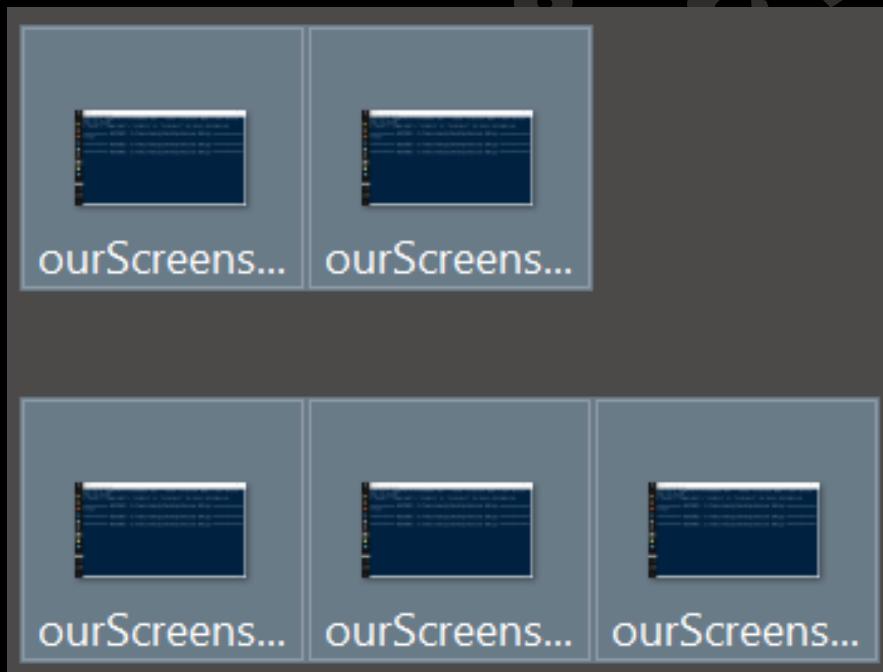


We use the `grab` function to take this screen shot and save it as a `.png` file named `ourTexture`

Screenshot Full Screen, Repeatedly

```
from PIL import ImageGrab  
import time
```

```
x = 1  
while x <= 5:  
    time.sleep(4.0)  
    x = x + 1;  
    ourImage = ImageGrab.grab()  
    ourImage.save("ourScreenshot" + str(x)+".png")
```



We grab a screenshot every 4 seconds and repeat the process 5 times.

Keyboard Controls

```
Command Prompt
Microsoft Windows [Version 10.0.19041.804]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\energy>pip install keyboard
```

pip install keyboard

```
Command Prompt
Microsoft Windows [Version 10.0.19041.804]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\energy>pip install keyboard
Collecting keyboard
  Downloading keyboard-0.13.5-py3-none-any.whl (58 kB)
    |██████████| 58 kB 466 kB/s
Installing collected packages: keyboard
Successfully installed keyboard-0.13.5

C:\Users\energy>
```

Screenshot by Keyboard Press

```
from PIL import ImageGrab  
import keyboard
```

```
while True:
```

```
    try:
```

```
        if keyboard.is_pressed('enter'):  
            ourImage = ImageGrab.grab()  
            ourImage.save("ourTexture.png")
```

```
            break
```

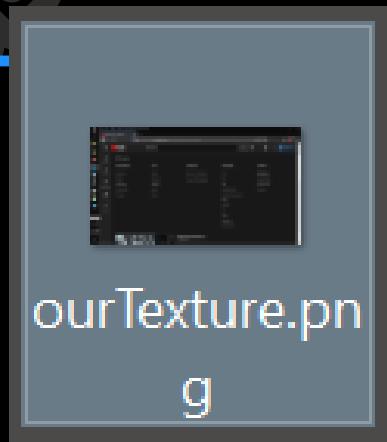
```
    else:
```

```
        pass
```

```
except:
```

```
    break
```

The screenshot is created in the folder where this script is activated from.



We grab a screenshot by pressing the Enter button on our Keyboard

SIMULATION

College of Scripting
Music & Science

Pygame to Make Games!

Command Prompt

Microsoft Windows [Version 10.0.19041.804]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\energy>pip install pygame

pip install pygame

Command Prompt

Installing collected packages: pygame
Successfully installed pygame-2.0.1

C:\Users\energy>

pygame – Window, Fill Color, Draw Circle

```
import pygame
pygame.init()

ourWindow =
pygame.display.set_mode( [500, 500] )

running = True

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    ourWindow.fill((0, 0, 0))

    pygame.draw.circle(ourWindow, (0, 150,
255), (100, 100), 50)

    pygame.display.flip()

pygame.quit()
```

We Type `pip show` and then **name of module**

```
Command Prompt  
Microsoft Windows [Version 10.0.19041.94]  
(c) 2020 Microsoft Corporation. All rights reserved.  
C:\Users\energy>pip show pygame
```

pip show pygame

```
Command Prompt  
C:\Users\energy>pip show pygame  
Name: pygame  
Version: 2.0.1  
Summary: Python Game Development  
Home-page: https://www.pygame.org  
Author: A community project.  
Author-email: pygame@pygame.org  
License: LGPL  
Location: c:\users\energy\appdata\local\programs\python\python39\lib  
\site-packages  
Requires:  
Required-by:  
C:\Users\energy>
```

**show INFO
about ANY
module that we
have already
installed!**

**Shows location
of the package**

Select Command Prompt

```
C:\Users\energy>pip show pygame
Name: pygame
Version: 2.0.1
Summary: Python Game Development
Home-page: https://www.pygame.org
Author: A community project.
Author-email: pygame@pygame.org
License: LGPL
Location: c:\users\energy\appdata\local\programs\python\python39\lib
\site-packages
```

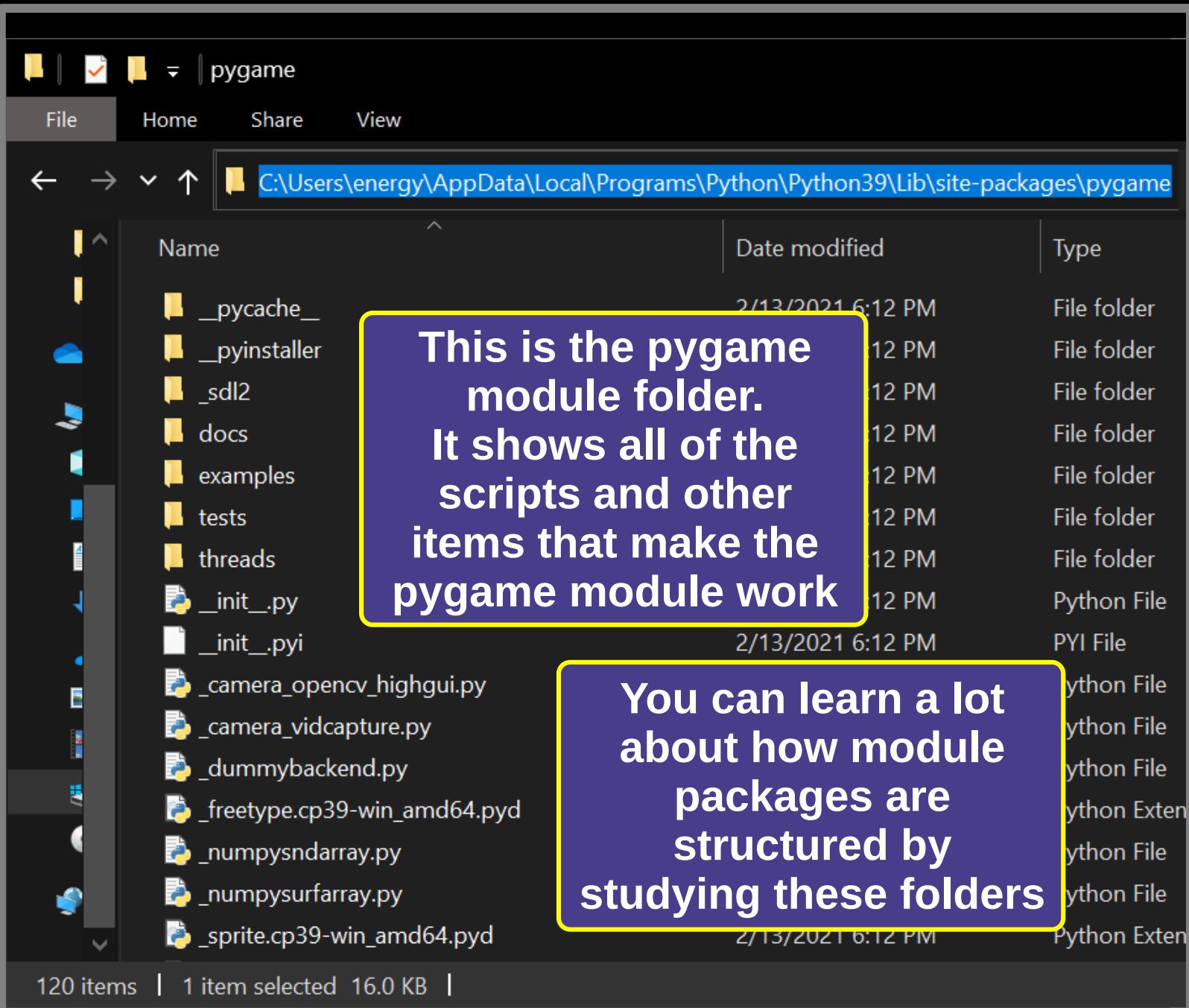
Highlight
the
Location
of the
packages

Press
Control + C
to Copy

Control + V to Paste the
location into the address
bar of any folder

We can now Open the
pygame folder, and
SEE all of the scripts
that make it work!

Study the pygame Folder Contents



VIDEO

College of Scripting
Music & Science

Movie Module

```
C:\ Command Prompt
Microsoft Windows [Version 10.0.19041.804]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\energy>pip install moviepy
```

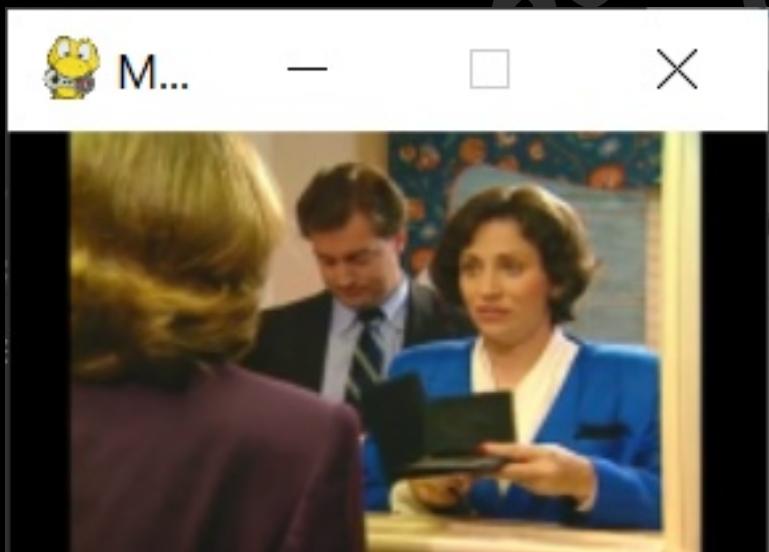
pip install moviepy

```
C:\ Command Prompt
Running setup.py install for moviepy ... done
Successfully installed chardet-4.0.0 decorator-4.4.2
idna-2.10 imageio-2.9.0 imageio-ffmpeg-0.4.3 moviepy-
1.0.3 proglog-0.1.9 requests-2.25.1 tqdm-4.57.0

C:\Users\energy>
```

pygame, moviepy, play movie, size by ratio

```
import moviepy  
from moviepy.editor import *  
import pygame  
  
ourVid = VideoFileClip('testMovie.mp4').resize(0.3)  
  
ourVid.preview()  
  
pygame.quit()
```



Make sure to have
pygame
and
moviepy
installed,
as shown before.

pygame, moviepy, play movie, size by x, y

```
import moviepy  
from moviepy.editor import *  
import pygame
```

```
x = 700  
y = 500
```

```
ourVid = VideoFileClip('testMovie.mp4').resize((x, y))
```

```
ourVid.preview()
```

```
pygame.quit()
```



pygame, moviepy, Start and End time

```
import moviepy  
from moviepy.editor import *  
import pygame
```

```
ourVid = VideoFileClip('testMovie.mp4').resize(1.4)
```

```
aPortion = ourVid.subclip(5, 20)
```

```
aPortion.preview()
```

```
pygame.quit()
```



pygame, moviepy, slow motion

```
import moviepy  
from moviepy.editor import *  
import pygame  
  
ourVid = VideoFileClip('testMovie.mp4').resize(1.4)  
  
aPortion = ourVid.subclip(5, 20)  
ourVid = aPortion.speedx(0.3)  
  
ourVid.preview()  
  
pygame.quit()
```



Nested Dictionaries

College of Scripting
Music & Science

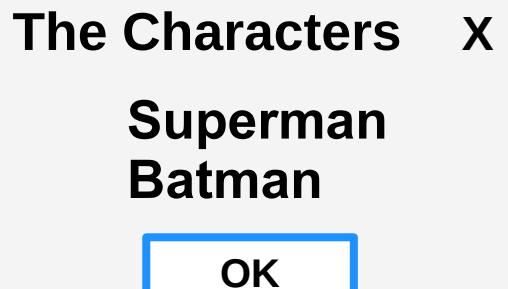
Nested Dictionaries - Show All

Christopher Topalian

```
import ctypes
characters = {
    'Superman':
    {
        'location': 'Metropolis',
        'planet': 'Krypton'
    },
    'Batman':
    {
        'location': 'Gotham',
        'planet': 'Earth'
    }
}
answer = ""
```

```
for z in characters:
    answer += z + '\n'
```

```
ctypes.windll.user32.MessageBoxW(0,
answer, "The Characters", 0)
```



Loop through all entries
of our Nested Dictionaries



Nested Dictionaries - Filter for String

```

import ctypes
characters = {
    'Superman':
    {
        'location': 'Metropolis',
        'planet': 'Krypton'
    },
    'Batman':
    {
        'location': 'Gotham',
        'planet': 'Earth'
    }
}
answer = " "
for z in characters:
    if (characters[z]['location'] == 'Metropolis') :
        answer += z + '\n'

ctypes.windll.user32.MessageBoxW(0,
answer, "The Characters", 0)

```

The Characters X

Superman

OK

only if location
is Metropolis

Nested Dictionaries in Window

```
from tkinter import *
```

```
characters = {
```

```
    'Superman':
```

```
{
```

```
    'location': 'Metropolis',
```

```
    'planet': 'Krypton'
```

```
},
```

```
'Batman':
```

```
{
```

```
    'location': 'Gotham',
```

```
    'planet': 'Earth'
```

```
}
```

```
}
```

```
ourWindow = Tk()
```

```
ourWindow.geometry('215x175+300+150')
```

```
ourWindow.title('Our App')
```

```
x = 75
```

```
y = 5
```

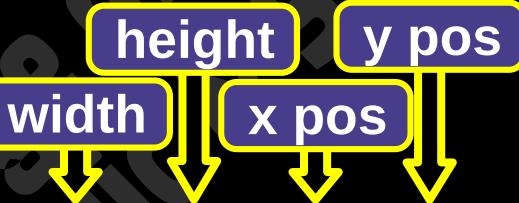
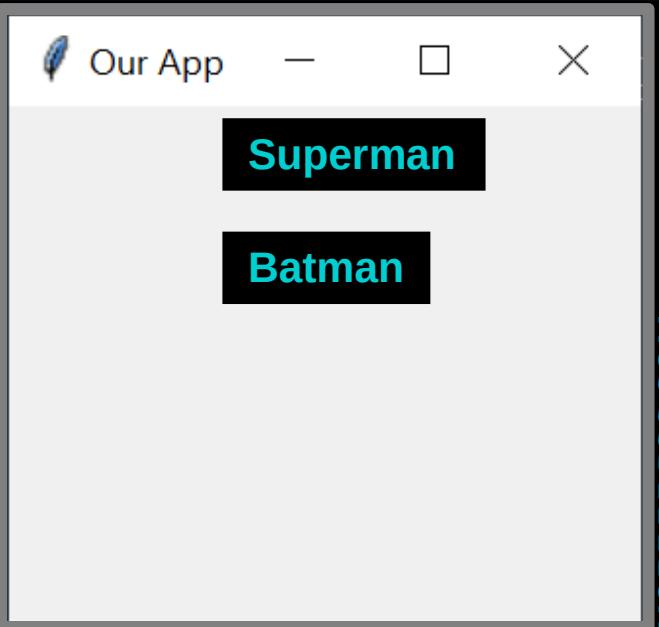
```
for z in characters:
```

```
    ourFirstLabel = Label(ourWindow,
```

```
        text = z, fg = 'aqua', bg = 'black').place(x = x, y = y)
```

```
        y += 40
```

```
ourWindow.mainloop()
```



Loop through all entries
of our Nested Dictionaries

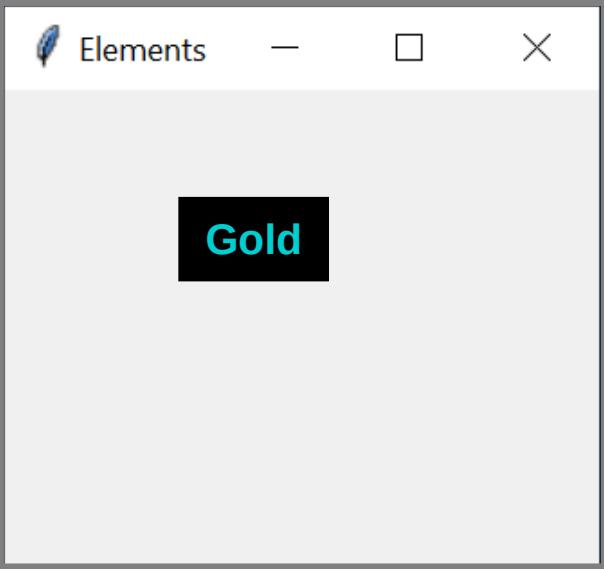
Nested Dictionaries - Make Labels

Christopher Tapalian

```
from tkinter import *
periodicTable= {
    'Copper':
    {
        'abbreviation': 'Cu',
        'electrons': '29'
    },
    'Gold':
    {
        'abbreviation': 'Au',
        'electrons': '79'
    }
}
ourWindow = Tk()
ourWindow.geometry('220x175+200+100')
ourWindow.title('Elements')

x = 75
y = 5

for z in periodicTable:
    if (periodicTable[z]['electrons'] > '35'):
        ourFirstLabel = Label(ourWindow, text = z, fg = 'aqua',
        bg = 'black').place(x = x, y = y)
        y += 40
ourWindow.mainloop()
```



The image shows a screenshot of a Tkinter application window titled "Elements". Inside the window, there is a single black rectangular label containing the word "Gold" in white text.

Annotations:

- Yellow boxes with arrows point from the variable names `height`, `width`, `x pos`, and `y pos` in the code to their corresponding properties on the "Gold" label.
- A yellow box contains the text "only if electrons are more than 35" with a yellow arrow pointing down to the `if` statement in the code.

Nested Dictionaries - Show All

```
languages = {  
    'Javascript':  
    {  
        'year': '1995',  
        'extension': '.js'  
    },  
    'Python':  
    {  
        'year': '1991',  
        'extension': '.py'  
    }  
}  
for x in languages:  
    print('\nTitle', x)  
    print('Created in', languages[x]['year'])  
    print('Extension is', languages[x]['extension'])  
  
input('Press Enter to Exit')
```

Console Version

Result

Title Javascript
Created in 1995
Extension is .js

Title Python
Created in 1991
Extension is .py

Loop through all entries
of our Nested Dictionaries

Nested Dictionaries - Show All

```
periodicTable= {
```

```
    'Silver':
```

```
{
```

```
    'abbreviation': 'Ag',  
    'electrons': '47'
```

```
,
```

```
    'Iron':
```

```
{
```

```
    'abbreviation': 'Fe',  
    'electrons': '26'
```

```
}
```

```
}
```

```
for x in periodicTable:
```

```
    print('\nTitle:', x)
```

```
    print('Letters:', periodicTable[x]['abbreviation'])
```

```
    print('Atomic #:', periodicTable[x]['electrons'])
```

```
input('Press Enter to Exit')
```

Console Version

Result

Title: Silver
Letters: Ag
Atomic # 47

Title: Iron
Letters: Fe
Atomic #: 26

Loop through all entries
of our Nested Dictionaries

Nested Dictionaries - Show All

```
periodicTable= {
```

Console Version

```
    'Copper':
```

```
{
```

```
        'abbreviation': 'Cu',
```

```
        'url': 'https://en.wikipedia.org/wiki/Copper',
```

```
        'electrons': '29'
```

```
,
```

```
    'Gold':
```

```
{
```

```
        'abbreviation': 'Au',
```

```
        'url': 'https://en.wikipedia.org/wiki/Gold',
```

```
        'electrons': '79'
```

```
}
```

```
}
```

```
for x in periodicTable:
```

```
    print('\nTitle:', x)
```

```
    print('Letters', periodicTable[x]['abbreviation'])
```

```
    print('Url', periodicTable[x]['url'])
```

```
    print('Atomic Number', periodicTable[x]['electrons'])
```

```
input('Press Enter to Exit')
```

Loop through all entries
of our Nested Dictionaries

Nested Dictionaries - Filter by Number

```
periodicTable= {
```

```
    'Copper':
```

```
{
```

```
    'abbreviation': 'Cu',
```

```
    'url': 'https://en.wikipedia.org/wiki/Copper',
```

```
    'electrons': '29'
```

```
,
```

```
    'Gold':
```

```
{
```

```
    'abbreviation': 'Au',
```

```
    'url': 'https://en.wikipedia.org/wiki/Gold',
```

```
    'electrons': '79'
```

```
}
```

```
}
```

```
for x in periodicTable:
```

```
    if (periodicTable[x]['electrons'] < '75') :
```

```
        print('\nTitle:', x)
```

```
        print('Letters:', periodicTable[x]['abbreviation'])
```

```
        print('Url:', periodicTable[x]['url'])
```

```
        print('Atomic Number:', periodicTable[x]['electrons'])
```

```
input('Press Enter to Exit')
```

Result is shown
on NEXT PAGE=>

Show if electrons
are less than 75



Result of Previous Page Shown

IDLE Shell 3.9.1

File Edit Shell Debug Options Window Help

```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec  7  
2020, 17:08:21) [MSC v.1927 64 bit (AMD64)]  
] on win32
```

```
Type "help", "copyright", "credits" or "license()" for more information.
```

```
>>>
```

```
===== RESTART: C:/Users/energy/Desktop/elements.py =====
```

Title: Copper

Letters: Cu

Url: <https://en.wikipedia.org/wiki/Copper>

Atomic Number: 29

Ln: 6 Col: 4

Title: Copper

Letters: Cu

Url: <https://en.wikipedia.org/wiki/Copper>

Atomic Number: 29

Nested Dictionaries - Filter by String

```

investigation = {
    'Suspect One':
    {
        'name': 'John Doe',
        'date': '2021/09/01 12:00 AM',
        'location': 'Boston, MA'
    },
    'Suspect Two':
    {
        'name': 'Jane Doe',
        'date': '2020/08/12 12:00 AM',
        'location': 'Denver, Colorado'
    }
}
for x in investigation:
    if (investigation[x]['location'] == 'Boston, MA'):
        print('\nTitle:', x)
        print('Name:', investigation[x]['name'])
        print('Date:', investigation[x]['date'])
        print('Location:', investigation[x]['location'])


```

Result also shown
on NEXT PAGE=>

Result

Title: Suspect One

Name: John Doe

Date: 2021/09/01

12:00 AM

Location: Boston, MA

Show if location
is Boston, MA



Result of Previous Page Shown

IDLE Shell 3.9.1

File Edit Shell Debug Options Window Help

```
CREATES OR LICENSE()
for more information.

>>>
=====
: C:/Users/energy/Desktop
/suspects.py =====
=====
```

Title: Suspect One
name: John Doe
Date: 09/1/2021 12:00 AM
Location: Boston, MA

Ln: 1 Col: 55

Nested Dictionaries - Filter by String

```
spyDevices = {  
    'Alexa':  
    {  
        'company': 'Amazon',  
        'date': '2014/11/06 12:00 AM'  
    },  
    'Google Home':  
    {  
        'company': 'Alphabet Inc',  
        'date': '2016/11/04 12:00 AM'  
    }  
}  
  
for x in spyDevices:  
    if (spyDevices[x]['company'] != 'Amazon'):  
        print('\nTitle:', x)  
        print('Company', spyDevices[x]['company'])  
        print('Date', spyDevices[x]['date'])  
  
input('Press Enter to Exit')
```

Result

Title: Google Home
Company: Alphabet Inc
Date: 11/4/2016

Show if company
is NOT Amazon

Nested Dictionaries - Filter by String

```
collegeVideos= {
    'Evidence Viewer for Investigators':
    {
        'date': '2020/09/15 12:00 AM',
        'url': 'https://youtu.be/jMBKRbG_bXw',
        'description': 'javascript'
    },
    'Evidence Application':
    {
        'date': '2020/09/13 12:00 AM',
        'url': 'https://youtu.be/9QWhAHqxB_Q',
        'description': 'javascript'
    }
}
for x in collegeVideos:
    if (collegeVideos[x]['description'] == 'javascript'):
        print('\nTitle:', x)
        print('Date', collegeVideos[x]['date'])
        print('Url', collegeVideos[x]['url'])
        print('Description', collegeVideos[x]['description'])
input('Press Enter to Exit')
```

Show if description
is javascript



**Filter
by DATE
LESS
THAN**

College of Scripting Music & Science

Nested Dictionaries - Filter by Date

```
videoGames = {
```

```
    'Cities Skylines':
```

```
{
```

```
        'engine': 'Unity',
```

```
        'date': '2015/03/15 12:00 AM'
```

```
,
```



```
'Garry\''s Mod':
```

```
{
```

```
        'engine': 'Source',
```

```
        'date': '2004/12/24 12:00 AM'
```

```
}
```

```
}
```

```
for x in videoGames:
```

```
    if (videoGames[x]['date'] < '2005/03/14'):
```

```
        print('\nTitle:', x)
```

```
        print('Engine:', videoGames[x]['engine'])
```

```
        print('Date:', videoGames[x]['date'])
```

```
input('Press Enter to Exit')
```

Console Version

Result

Title: Garry's Mod

Engine: Source

Date: 2004/12/24

' is used to escape
the apostrophe

Show if date is less
than 2005 March 14th



Nested Dictionaries - Filter by Date

```
import ctypes
```

```
videoGames = {
    'Cities Skylines':
```

```
{
```

```
    'engine': 'Unity',
```

```
    'date': '2015/03/15 12:00 AM'
```

```
},
```



```
'Garry\l's Mod':
```

```
{
```

```
    'engine': 'Source',
```

```
    'date': '2004/12/24 12:00 AM'
```

```
}
```

```
}
```

```
answer = " "
```

```
for z in videoGames:
```

```
    if (videoGames[z]['date'] < '2005/03/14') :
```

```
        answer += z + '\n'
```

```
        answer += videoGames[z]['engine'] + '\n'
```

```
        answer += videoGames[z]['date'] + '\n'
```

```
ctypes.windll.user32.MessageBoxW(0,
    answer, "Simulation", 0)
```

Simulation

X

Gary's Mod

Source

2004/12/24 12:00AM

OK

MessageBoxW
version

' is used to
escape the
apostrophe

Show if date is less
than 2005 March 14th



Filter by
DATE
GREATER
THAN

Nested Dictionaries - Filter by Date

Console Version

```
collegeVideos= {
    'LSL Scripting Tutorial, How to Make a New Script':
    {
        'date': '2020/09/15 12:00 AM',
        'url': 'https://youtu.be/JyTxamxK3E8',
        'description': 'lsl scripting'
    },
    'Spreadsheets Become JS 2D VIRTUAL WORLDS':
    {
        'date': '2020/09/13 12:00 AM',
        'url': 'https://youtu.be/CsijyVNLwQc',
        'description': 'javascript'
    }
}
for x in collegeVideos:
    if (collegeVideos[x]['date'] > '2020/09/14') :
        print('\nTitle:', x)
        print('Date', collegeVideos[x]['date'])
        print('Url', collegeVideos[x]['url'])
        print('Description', collegeVideos[x]['description'])
input('Press Enter to Exit')
```

Show if date is greater
than 2020 Sept 14th



Nested Dictionaries - Filter by Date

```

import ctypes
collegeVideos= {
    'LSL Scripting Tutorial, How to Make a New Script':
    {
        'date': '2020/09/15 12:00 AM',
        'url': 'https://youtu.be/JyTxamxK3E8',
    },
    'Spreadsheets Become JS 2D VIRTUAL WORLDS':
    {
        'date': '2020/09/13 12:00 AM',
        'url': 'https://youtu.be/CsijyVNLwQc',
    }
}
answer = " "
for z in collegeVideos:
    if (collegeVideos[z]['date'] > '2020/09/14') :
        answer += z + '\n'
        answer += collegeVideos[z]['date'] + '\n'
        answer += collegeVideos[z]['url'] + '\n'

ctypes.windll.user32.MessageBoxW(0,
answer, "The College Videos", 0)

```

Show if date is greater
than 2020 Sept 14th



ADD Nested Dictionary

Nested Dictionaries - ADD Dictionary

```
periodicTable = {
    'Copper':
    {
        'abbreviation': 'Cu',
        'electrons': '29'
    },
    'Gold':
    {
        'abbreviation': 'Au',
        'electrons': '79'
    }
}
```

```
periodicTable['Platinum'] = {
```

```
    'abbreviation': 'Pt',
    'electrons': '78'
}
```

```
for x in periodicTable:
```

```
    print('\nTitle:', x)
```

```
    print('Letters:', periodicTable[x]['abbreviation'])
```

```
    print('Atomic #:', periodicTable[x]['electrons'])
```

```
input('Press Enter to Exit')
```

Result

Title: Copper
Letters: Cu
Atomic #: 29

Title: Gold
Letters: Au
Atomic #: 79

Title: Platinum
Letters: Pt
Atomic #: 78

console version

Add Dictionary

Loop through all entries
of our Nested Dictionaries

Nested Dictionaries - ADD Dictionary

Christopher Topalian

```
import ctypes
periodicTable= {
    'Copper':
    {
        'abbreviation': 'Cu',
        'electrons': '29'
    },
    'Gold':
    {
        'abbreviation': 'Au',
        'electrons': '79'
    }
}
periodicTable['Platinum'] = {
    'abbreviation': 'Pt',
    'electrons': '78'
}
answer = " "
for z in periodicTable:
    answer += z + ' '
    answer += periodicTable[z]['abbreviation'] + ': '
    answer += periodicTable[z]['electrons'] + '\n'
ctypes.windll.user32.MessageBoxW(0,
answer, "The Elements", 0)
```

The Elements

X

Copper Cu: 29

Gold Au: 79

Platinum Pt: 78

OK

MessageBoxW
version

Add Dictionary

DELETE Nested Dictionary

Nested Dictionaries - Delete Dictionary

```
periodicTable= {  
    'Copper':  
    {  
        'abbreviation': 'Cu',  
        'electrons':'29'  
    },  
    'Gold':  
    {  
        'abbreviation': 'Au',  
        'electrons':'79'  
    }  
}
```

```
del periodicTable['Gold']
```

```
for x in periodicTable:
```

```
    print('\nTitle:', x)
```

```
    print('Letters:', periodicTable[x]['abbreviation'])
```

```
    print('Atomic #:', periodicTable[x]['electrons'])
```

```
input('Press Enter to Exit')
```

Result

Title: Copper
Letters: Cu
Atomic #: 29

console
version

Delete Dictionary

Loop through all entries
of our Nested Dictionaries

Nested Dictionaries - Delete Dictionary

```

import ctypes
periodicTable= {
    'Copper':
    {
        'abbreviation': 'Cu',
        'electrons': '29'
    },
    'Gold':
    {
        'abbreviation': 'Au',
        'electrons': '79'
    }
}
del periodicTable['Gold']
answer = " "
for z in periodicTable:
    answer += z + ' '
    answer += periodicTable[z]['abbreviation'] + ': '
    answer += periodicTable[z]['electrons'] + '\n'
ctypes.windll.user32.MessageBoxW(0,
answer, "The Elements", 0)

```

The Elements

X

Copper Cu: 29

OK

MessageBoxW
version

Delete Dictionary



DELETE Last Entry of a Nested Dictionary

Nested Dictionaries - Delete Last Dictionary

```
periodicTable= {  
    'Copper':  
    {  
        'abbreviation': 'Cu',  
        'electrons': '29'  
    },  
    'Gold':  
    {  
        'abbreviation': 'Au',  
        'electrons': '79'  
    }  
}
```

Delete Last Entry

```
periodicTable.popitem()  
for x in periodicTable:  
    print('\nTitle:', x)  
    print('Letters:', periodicTable[x]['abbreviation'])  
    print('Atomic #:', periodicTable[x]['electrons'])  
  
input('Press Enter to Exit')
```

Loop through all entries
of our Nested Dictionaries

Result
Title: Copper
Letters: Cu
Atomic #: 29

console
version

Nested Dictionaries - Delete Last Dictionary

```
import ctypes
periodicTable= {
    'Copper':
    {
        'abbreviation': 'Cu',
        'electrons': '29'
    },
    'Gold':
    {
        'abbreviation': 'Au',
        'electrons': '79'
    }
}
```

```
periodicTable.popitem()
```

```
answer = ""
```

```
for z in periodicTable:
```

```
    answer += z + ''
```

```
    answer += periodicTable[z]['abbreviation'] + ': '
```

```
    answer += periodicTable[z]['electrons'] + '\n'
```

```
ctypes.windll.user32.MessageBoxW(0,
answer, "The Elements", 0)
```

The Elements

X

Copper Cu: 29

OK

MessageBoxW
version

Delete Last Entry

Loop through all entries
of our Nested Dictionaries



DELETE First Entry of a Nested Dictionary

Nested Dictionaries - Delete 1st Dictionary `pop`

Christopher Topalian

```
periodicTable= {
```

```
    'Copper':
```

```
{
```

```
    'abbreviation': 'Cu',
```

```
    'electrons': '29'
```

```
,
```

```
    'Gold':
```

```
{
```

```
    'abbreviation': 'Au',
```

```
    'electrons': '79'
```

```
}
```

```
}
```

```
periodicTable.pop(list(periodicTable)[0])
```

```
for x in periodicTable:
```

```
    print('\nTitle:', x)
```

```
    print('Letters:', periodicTable[x]['abbreviation'])
```

```
    print('Atomic #:', periodicTable[x]['electrons'])
```

```
input('Press Enter to Exit')
```

Result

Title: Gold

Letters: Au

Atomic #: 79

console
version

Delete First Entry



Nested Dictionaries - Delete 1st Dictionary pop

```

import ctypes
periodicTable= {
    'Copper':
    {
        'abbreviation': 'Cu',
        'electrons': '29'
    },
    'Gold':
    {
        'abbreviation': 'Au',
        'electrons': '79'
    }
}
periodicTable.pop(list(periodicTable)[0])
answer = " "
for z in periodicTable:
    answer += z + ' '
    answer += periodicTable[z]['abbreviation'] + ': '
    answer += periodicTable[z]['electrons'] + '\n'
ctypes.windll.user32.MessageBoxW(0,
    answer, "The Elements", 0)

```

The Elements

X

Gold Au: 79

OK

MessageBoxW
version

Delete First Entry



Nested Dictionaries - Delete 1st Dictionary `del`

```
periodicTable= {
```

```
    'Copper':
```

```
{
```

```
    'abbreviation': 'Cu',
```

```
    'electrons':'29'
```

```
,
```

```
'Gold':
```

```
{
```

```
    'abbreviation': 'Au',
```

```
    'electrons':'79'
```

```
}
```

```
}
```

```
del periodicTable[ list(periodicTable)[0] ]
```

```
for x in periodicTable:
```

```
    print('\nTitle:', x)
```

```
    print('Letters:', periodicTable[x]['abbreviation'])
```

```
    print('Atomic #:', periodicTable[x]['electrons'])
```

```
input('Press Enter to Exit')
```

Result

Title: Gold

Letters: Au

Atomic #: 79

**console
version**

Delete First Entry



Nested Dictionaries - Delete 1st Dictionary del

```

import ctypes
periodicTable= {
    'Copper':
    {
        'abbreviation': 'Cu',
        'electrons': '29'
    },
    'Gold':
    {
        'abbreviation': 'Au',
        'electrons': '79'
    }
}
del periodicTable[ list(periodicTable)[0] ]
answer = " "
for z in periodicTable:
    answer += z + ' '
    answer += periodicTable[z]['abbreviation'] + ': '
    answer += periodicTable[z]['electrons'] + '\n'
ctypes.windll.user32.MessageBoxW(0,
answer, "The Elements", 0)

```

The Elements

X

Gold Au: 79

OK

MessageBoxW
version

Delete First Entry



DELETE Second Entry of a Nested Dictionary

Nested Dictionaries - Delete 2nd Dictionary `pop`

Christopher Topalian

```
periodicTable= {
```

```
    'Copper':
```

```
{
```

```
    'abbreviation': 'Cu',
```

```
    'electrons':'29'
```

```
,
```

```
    'Gold':
```

```
{
```

```
    'abbreviation': 'Au',
```

```
    'electrons':'79'
```

```
}
```

```
}
```

```
periodicTable.pop(list(periodicTable)[1])
```

```
for x in periodicTable:
```

```
    print('\nTitle:', x)
```

```
    print('Letters:', periodicTable[x]['abbreviation'])
```

```
    print('Atomic #:', periodicTable[x]['electrons'])
```

```
input('Press Enter to Exit')
```

Result

Title: Copper

Letters: Cu

Atomic #: 29

console
version

Delete Second Entry



Nested Dictionaries - Delete 2nd Dictionary `pop`

```

import ctypes
periodicTable= {
    'Copper':
    {
        'abbreviation': 'Cu',
        'electrons': '29'
    },
    'Gold':
    {
        'abbreviation': 'Au',
        'electrons': '79'
    }
}
periodicTable.pop(list(periodicTable)[1])
answer = ""

for z in periodicTable:
    answer += z + ' '
    answer += periodicTable[z]['abbreviation'] + ': '
    answer += periodicTable[z]['electrons'] + '\n'

ctypes.windll.user32.MessageBoxW(0,
    answer, "The Elements", 0)

```

The Elements

X

Copper Cu: 29

OK

MessageBoxW
version

Delete Second Entry



EDIT an Entry by Index Nested Dictionary

Nested Dictionaries - Edit Dictionary by Index

```

periodicTable= {
    'Copper':
    {
        'abbreviation': 'Cu',
        'electrons':'29'
    },
    'Gold':
    {
        'abbreviation': 'Au',
        'electrons':'79'
    }
}
periodicTable[ list(periodicTable)[0] ]= {
    'abbreviation':'Cupric', 'electrons':'Twenty Nine' }

for x in periodicTable:
    print('\nTitle:', x)
    print('Letters:', periodicTable[x]['abbreviation'])
    print('Atomic #:', periodicTable[x]['electrons'])

input('Press Enter to Exit')

```

Result

```

Title: Copper
Letters: Cuprum
Atomic #: Twenty Nine

Title: Gold
Letters: Au
Atomic #: 79

```

**console
version**

Edit Data of first entry



Nested Dictionaries - Edit Dictionary by Index

```

import ctypes
periodicTable= {
    'Copper':
    {
        'abbreviation': 'Cu',
        'electrons': '29'
    },
    'Gold':
    {
        'abbreviation': 'Au',
        'electrons': '79'
    }
}
periodicTable[ list(periodicTable)[0] ]= {
    'abbreviation': ' Cuprum',
    'electrons': ' Twenty Nine' }
answer = " "
for z in periodicTable:
    answer += z + ''
    answer += periodicTable[z]['abbreviation'] + ': '
    answer += periodicTable[z]['electrons'] + '\n'
ctypes.windll.user32.MessageBoxW(0,
    answer, "The Elements", 0)

```

The Elements

X

Copper Cuprum: 29
Gold Au: 79

OK

MessageBoxW
version

Edit Data of first entry



EDIT an Entry by Key Nested Dictionary

Nested Dictionaries - Edit Dictionary by Key

```
periodicTable= {
```

```
    'Copper':
```

```
{
```

```
    'abbreviation': 'Cu',
```

```
    'electrons': '29'
```

```
,
```

```
'Gold':
```

```
{
```

```
    'abbreviation': 'Au',
```

```
    'electrons': '79'
```

```
}
```

```
}
```

```
periodicTable['Copper'] = {
```

```
    'abbreviation':'Cuprum',
```

```
    'electrons':'Twenty Nine'
```

```
}
```

```
for x in periodicTable:
```

```
    print('\nTitle:', x)
```

```
    print('Letters:', periodicTable[x]['abbreviation'])
```

```
    print('Atomic #:', periodicTable[x]['electrons'])
```

```
input('Press Enter to Exit')
```

Result

Title: Copper

Letters: Cuprum

Atomic #: Twenty Nine

Title: Gold

Letters: Au

Atomic #: 79

**console
version**

Edit Data of
Copper entry

Loop through all entries
of our Nested Dictionaries

Nested Dictionaries - Edit Dictionary by Key

```

import ctypes
periodicTable= {
    'Copper':
    {
        'abbreviation': 'Cu',
        'electrons': '29'
    },
    'Gold':
    {
        'abbreviation': 'Au',
        'electrons': '79'
    }
}
periodicTable['Copper'] = {
    'abbreviation': 'Cuprum',
    'electrons': 'Twenty Nine'
}
answer = " "
for z in periodicTable:
    answer += z + ' '
    answer += periodicTable[z]['abbreviation'] + ': '
    answer += periodicTable[z]['electrons'] + '\n'
ctypes.windll.user32.MessageBoxW(0,
    answer, "The Elements", 0)

```

The Elements

X

Copper Cuprum: 29
Gold Au: 79

OK

MessageBoxW
version

Edit Copper
abbreviation



Nested Dictionaries - Edit Value by Key

Christopher Topalian

```
periodicTable= {  
    'Copper':  
    {  
        'abbreviation': 'Cu',  
        'electrons': '29'  
    },  
    'Gold':  
    {  
        'abbreviation': 'Au',  
        'electrons': '79'  
    }  
}
```

```
periodicTable['Copper']['abbreviation'] = ''
```

```
for x in periodicTable:  
    print('\nTitle:', x)  
    print('Letters:', periodicTable[x]['abbreviation'])  
    print('Atomic #:', periodicTable[x]['electrons'])  
  
input('Press Enter to Exit')
```

Result

Title: Copper

Letters:

Atomic #: 29

Title: Gold

Letters: Au

Atomic #: 79

**console
version**

Edit Copper
abbreviation



empty string

Nested Dictionaries - Edit Value by Key

Christopher Topalian

```
import ctypes
periodicTable= {
    'Copper':
    {
        'abbreviation': 'Cu',
        'electrons': '29'
    },
    'Gold':
    {
        'abbreviation': 'Au',
        'electrons': '79'
    }
}
periodicTable['Copper']['abbreviation'] = ''
answer = ""
for z in periodicTable:
    answer += z + ''
    answer += periodicTable[z]['abbreviation'] + ':'
    answer += periodicTable[z]['electrons'] + '\n'
ctypes.windll.user32.MessageBoxW(0,
    answer, "The Elements", 0)
```

The Elements

X

Copper: 29

Gold Au: 79

OK

MessageBoxW
Version

Edit Copper
abbreviation



periodicTable['Copper']['abbreviation'] = ''

answer = ""

for z in periodicTable:

answer += z + ''

answer += periodicTable[z]['abbreviation'] + ':'

answer += periodicTable[z]['electrons'] + '\n'



empty string

Nested Dictionaries using or **to Filter using Multiple Criteria**

Nested Dictionaries - Filter for A or B

Christopher Topalian

```
import ctypes
elements = {
    'Zinc':
    {
        'electrons': '30',
        'type': 'metal'
    },
    'Silver':
    {
        'electrons': '47',
        'type': 'metal'
    }
}
answer = ""
for z in elements:
    if (elements[z]['electrons'] == '30'
        or elements[z]['electrons'] == '47'):
        answer += z + '\n'

ctypes.windll.user32.MessageBoxW(0,
    answer, "The Elements", 0)
```

The Elements

X

Zinc
Silver

OK

show if
electrons
has 30 or 47

show only if electrons
has 30 or 47

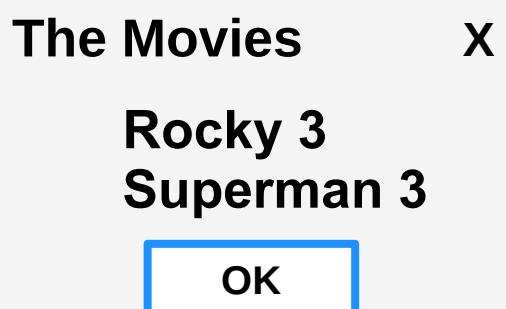


Nested Dictionaries - Filter for A or B

Christopher Topalian

```
import ctypes
movies = {
    'Rocky 3':
    {
        'kind': 'action',
        'date': '1982/05/28 12:00 AM'
    },
    'Superman 3':
    {
        'kind': 'adventure',
        'date': '1983/06/17 12:00 AM'
    }
}
answer = ""
for z in movies:
    if (movies[z]['kind'] == 'action'
        or movies[z]['kind'] == 'adventure'):
        answer += z + '\n'

ctypes.windll.user32.MessageBoxW(0,
    answer, "The Movies", 0)
```



show if kind
is action or
adventure

kind
action

kind
adventure

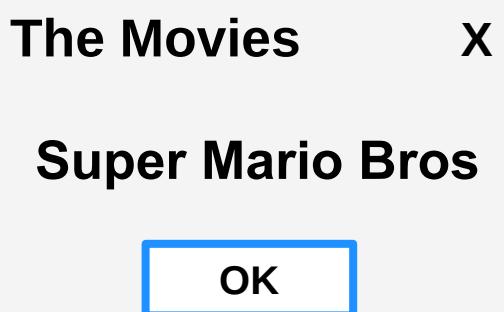


Nested Dictionaries using and **to Filter using Multiple Criteria**

Nested Dictionaries - Filter for A and B

```
import ctypes
movies = {
    'Super Mario Bros':
    {
        'type': 'comedy',
        'date': '1993/05/28 12:00 AM'
    },
    'Superman 1':
    {
        'type': 'adventure',
        'date': '1978/12/15 12:00 AM'
    }
}
answer = ""
for z in movies:
    if (movies[z]['date'] > '1979/01/01'
        and movies[z]['date'] < '1995/01/01'):
        answer += z + '\n'

ctypes.windll.user32.MessageBoxW(0,
    answer, "The Movies", 0)
```



show if date
is between
start date
and end date

start date
1979/01/01

end date
1995/01/01

Nested Dictionaries - Filter for A and B

```

import ctypes
speakers = {
    'Jane Doe':
    {
        'title': 'Genetics Lecture',
        'date': '2022/04/20 07:00 AM'
    },
    'John Doe':
    {
        'title': 'Biology Innovations',
        'date': '2022/04/20 11:00 AM'
    }
}
answer = ""
for z in speakers:
    if (speakers[z]['date'] > '2022/04/20 05:00 AM'
        and speakers[z]['date'] < '2022/04/20 10:00 AM'):
        answer += z + '\n'
ctypes.windll.user32.MessageBoxW(0,
    answer, "The Speakers", 0)

```

The Speakers

X

Jane Doe

OK

show if Year,
Month, Day,
Time is
between
start and end
condition

if greater than
2022/04/20
05:00 AM



if less than
2022/04/20
10:00 AM

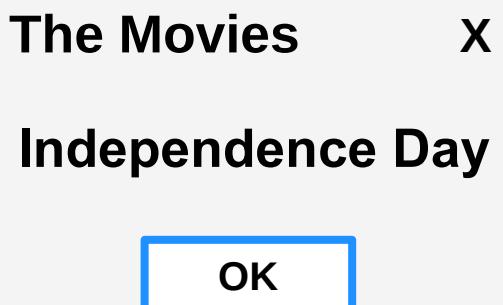
Nested Dictionaries - Filter for A and B

```

import ctypes
movies = {
    'Independence Day':
    {
        'kind': 'action',
        'date': '1996/07/03 12:00 AM'
    },
    'Star Trek First Contact':
    {
        'kind': 'fantasy',
        'date': '1996/11/22 12:00 AM'
    }
}
answer = ""
for z in movies:
    if (movies[z]['kind'] == 'action'
        and movies[z]['date'] < '1996/10/01'):
        answer += z + '\n'

ctypes.windll.user32.MessageBoxW(0,
    answer, "The Movies", 0)

```



show if date
is between
start date
and end date

kind
action

if date
less than
1996/10/01

COMBINE Nested Dictionaries

College of Scripting Music & Science

Nested Dictionaries - Combine Dictionaries

```
import ctypes
speakersUSA = {
    'Jane Doe':
    {
        'title': 'Genetics Lecture',
        'date': '2022/04/20 07:00 AM'
    }
}
speakersEU = {
    'Rose Doe':
    {
        'title': 'Botany Lecture',
        'date': '2022/04/20 11:00 AM'
    }
}
speakersUSA.update(speakersEU)
for x in speakersUSA:
    print('\nTitle:', x)
    print('Lecture:', speakersUSA[x]['title'])
    print('Date:', speakersUSA[x]['date'])

input('Press Enter to Exit')
```

Result

```
Title: Jane Doe
Lecture: Genetics
Lecture
Date: 2022/04/20
07:00 AM

Title: Rose Doe
Lecture: Botany
Lecture
Date: 2022/04/20
11:00 AM
```

ADD
Dictionary

Nested Dictionaries - Combine Dictionaries

```
import ctypes
speakersUSA = {
    'Jane Doe':
    {
        'title': 'Genetics Lecture',
        'date': '2022/04/20 07:00 AM'
    }
}
speakersEU = {
    'Rose Doe':
    {
        'title': 'Botany Lecture',
        'date': '2022/04/20 11:00 AM'
    }
}
speakersUSA.update(speakersEU)
answer = " "
for z in speakersUSA:
    answer += z + '\n'
ctypes.windll.user32.MessageBoxW(0,
    answer, "The Speakers", 0)
```

The Speakers X

Jane Doe
Rose Doe

OK

Added
Dictionary

Nested Dictionaries - Combine Dictionaries

The Complete Python

```
import ctypes
speakersUSA = {
    'Jane Doe':
    {
        'title': 'Genetics Lecture',
        'date': '2022/04/20 07:00 AM'
    }
}
speakersEU = {
    'Rose Doe':
    {
        'title': 'Botany Lecture',
        'date': '2022/04/20 11:00 AM'
    }
}
speakersUSA.update(speakersEU)
answer = ""
for z in speakersUSA:
    answer += z + ''
    answer += speakersUSA[z]['title'] + ': '
    answer += speakersUSA[z]['date'] + '\n'
ctypes.windll.user32.MessageBoxW(0,
    answer, "The Speakers", 0)
```

Result is shown
on NEXT PAGE=>

Each Dictionary
can have as many
entries as we want.

In this tutorial we
show only
one entry per
dictionary to keep
text easy to read.

Added
Dictionary

COPY the RESULT of our Message Box

The Speakers

X

Jane Doe Genetics Lecture 2022/04/20 07:00 AM
Rose Doe Botany Lecture: 2022/04/20 11:00AM

OK

We Left Click our Mouse Arrow anywhere on the Message Box.

We then press Ctrl + C to COPY.

We hear a sound, which indicates the action.

We can now paste the copied text into ANY text editor for easy viewing, pressing Ctrl + V to PASTE.

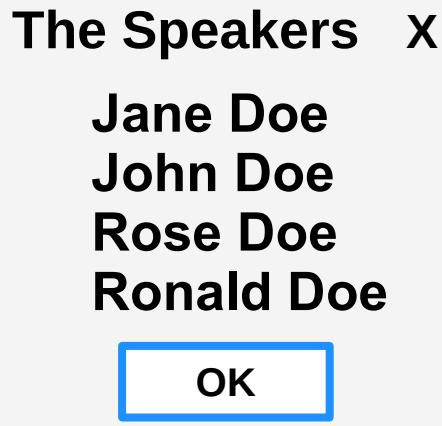
The Speakers

Jane Doe Genetics Lecture: 2022/04/20 07:00 AM
Rose Doe Botany Lecture: 2022/04/20 11:00 AM

OK

Nested Dictionaries - Combine Dictionaries

```
import ctypes
speakersUSA = {
    'Jane Doe':
    {
        'title': 'Genetics Lecture',
        'date': '2022/04/20 07:00 AM'
    },
    'John Doe':
    {
        'title': 'Mechanics Lecture',
        'date': '2022/04/20 09:00 AM'
    }
}
speakersEU = {
    'Rose Doe':
    {
        'title': 'Botany Lecture',
        'date': '2022/04/20 11:00 AM'
    },
    'Ronald Doe':
    {
        'title': 'Electronics Lecture',
        'date': '2022/04/20 01:00 PM'
    }
}
speakersUSA.update(speakersEU)
answer = " "
for z in speakersUSA:
    answer += z + '\n'
ctypes.windll.user32.MessageBoxW(0,
    answer, "The Speakers", 0)
```



We have added more entries to each dictionary, but the data is starting to take up a lot of space!

In the Next tutorial, we put our DATA in an EXTERNAL FILE, for easy readability!

Added Dictionary



EXTERNAL FILE Example

College of Scripting Music & Science

Our External Data File saved as **ourData.py**

```
speakersUSA = {  
    'Jane Doe':  
    {  
        'title': 'Genetics Lecture',  
        'date': '2022/04/20 07:00 AM'  
    },  
    'John Doe':  
    {  
        'title': 'Mechanics Lecture',  
        'date': '2022/04/20 09:00 AM'  
    }  
}  
  
speakersEU = {  
    'Rose Doe':  
    {  
        'title': 'Botany Lecture',  
        'date': '2022/04/20 11:00 AM'  
    },  
    'Ronald Doe':  
    {  
        'title': 'Electronics Lecture',  
        'date': '2022/04/20 01:00 PM'  
    }  
}
```

We save this script as **ourData.py**

On the next tutorial page, the script named **ourScript.py** uses the data from this page.

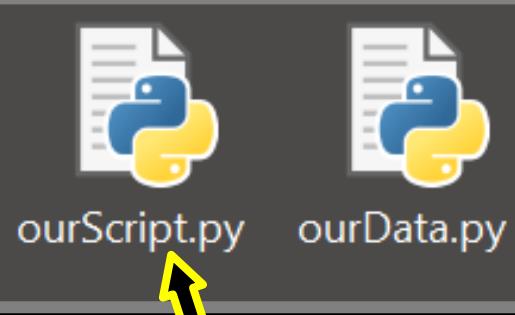
We place BOTH **ourData.py** and **ourScript.py** in the **SAME FOLDER!**

First Way of using ourData.py

```
import ctypes
from ourData import*
speakersUSA.update(speakersEU)
answer = " "
for z in speakersUSA:
    answer += z + '\n'
ctypes.windll.user32.MessageBoxW(0,
answer, "The Speakers", 0)
```

We save this
script as
ourScript.py

This script
USES the DATA
from the
previous page!



We place BOTH
ourScript.py
and
ourData.py
in the
SAME FOLDER!

Double Click
ourScript.py
and notice
how it **USES**
the Data from
ourData.py

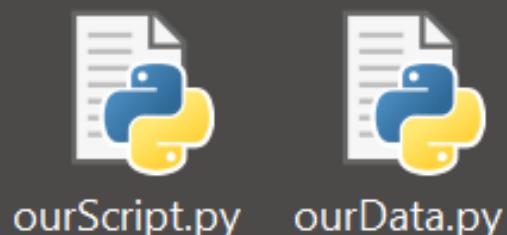
Second Way of using ourData.py

```
import ctypes, ourData  
ourData.speakersUSA.update(ourData.speakersEU)  
answer = ""  
for z in ourData.speakersUSA:  
    answer += z + '\n'  
ctypes.windll.user32.MessageBoxW(0,  
answer, "The Speakers", 0)
```

Dot is used to access the data

We save this script as ourScript.py

This script USES the DATA from the previous page!



We place BOTH ourScript.py and ourData.py in the SAME FOLDER!

Double Click ourScript.py and notice how it USES the Data from ourData.py

EXTERNAL FILE Example

College of Scripting Music & Science

Save this data script as `ourData.py`

```
periodicTable= {  
    'Copper':  
    {  
        'abbreviation': 'Cu',  
        'electrons': '29'  
    },  
    'Gold':  
    {  
        'abbreviation': 'Au',  
        'electrons': '79'  
    }  
}
```

We save this data script as `ourData.py`

This is the data that the script on the next page will reference!

The Script on the next page uses this data!

Make sure both scripts are located in the same folder.



`ourScript.py`



`ourData.py`

Window

shows data
after button
is pressed

```
from tkinter import *
```

```
from ourData import*
```

```
ourWindow = Tk()
```

```
ourWindow.geometry('220x175+200+100')
```

```
ourWindow.title('Elements')
```

```
theX = "
```

```
theY = "
```

```
def ourFunction():
```

```
    theX = 65
```

```
    theY = 5
```

```
    for z in periodicTable:
```

```
        if (periodicTable[z]['electrons'] > '27' ):
```

```
            ourLabel = Label(ourWindow, text = z, fg = 'aqua',
```

```
            bg = 'black', font=("Courier", 25)).place(x=theX,y=theY)
```

```
        theY += 40
```

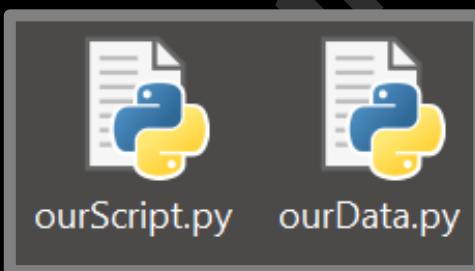
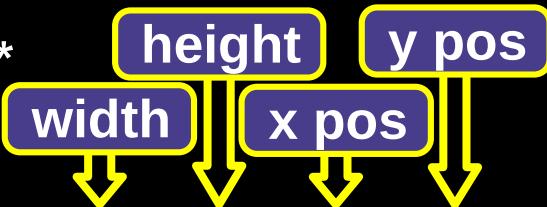
```
ourButton = Button(ourWindow,
```

```
text = 'Show Elements',
```

```
command = ourFunction,
```

```
fg = 'white', bg = 'black').place(x = 65, y = 120)
```

```
ourWindow.mainloop()
```



This script uses the file from the previous page named ourData.py

Both Scripts are in the Same Folder

**Next page
shows why
separating
Data from
Code is Useful**

Window shows data after button is pressed

```

from tkinter import *
periodicTable= {
    'Copper':
    {
        'abbreviation': 'Cu',
        'electrons': '29'
    },
    'Gold':
    {
        'abbreviation': 'Au',
        'electrons': '79'
    }
}
ourWindow = Tk()
ourWindow.geometry('220x175+200+100')
ourWindow.title('Elements')
theX = "
theY = "
def ourFunction():
    theX = 65
    theY = 5
    for z in periodicTable:
        if (periodicTable[z]['electrons'] > '27' ):
            ourLabel = Label(ourWindow, text = z, fg = 'aqua',
                bg = 'black', font=("Courier", 25)).place(x=theX,y=theY)
            theY += 40
ourButton = Button(ourWindow,
text = 'Show Elements',
command = ourFunction,
fg = 'white', bg = 'black').place(x = 65, y = 120)
ourWindow.mainloop()

```

As more data is added, it becomes more obvious why using an external file is a good choice.

This page shows the original script that we had turned into two files on the previous pages.

Down Down Menu

College of Scripting
Music & Science

Drop Down Menu

```
from tkinter import *  
MONTHS  
theMonths = [  
    'January', 'February', 'March', 'April',  
    'May', 'June', 'July', 'August', 'September',  
    'October', 'November', 'December'  
]  
ourWindow = Tk()  
theChoice = StringVar(ourWindow)  
theChoice.set(theMonths[0])  
theDropDownMenu = OptionMenu(ourWindow,  
    theChoice, *theMonths)  
theDropDownMenu.pack()  
def ConfirmIt():  
    print(theChoice.get() + ' is the month chosen')  
ourButton1 = Button(ourWindow,  
    text = "Select", command = ConfirmIt)  
ourButton1.pack()  
ourWindow.mainloop()
```

LOGIC

College of Scripting
Music & Science

AND

```
from ctypes import*
```

```
A = 1
```

```
B = 1
```

```
if (A == 1 and B == 1):
```

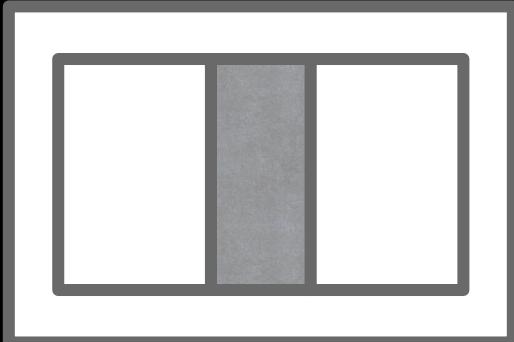
```
    print('Both are True')
```

```
windll.user32.MessageBoxW(0,
```

```
'Both are True', 'AND Gate', 0)
```

```
input('Press Enter to Exit')
```

A	B	=
0	0	0
0	1	0
1	0	0
1	1	1



Activates Only if
Both are True

NAND

```
from ctypes import*
```

```
A = 0
```

```
B = 0
```

```
if (A == 0 or B == 0):
```

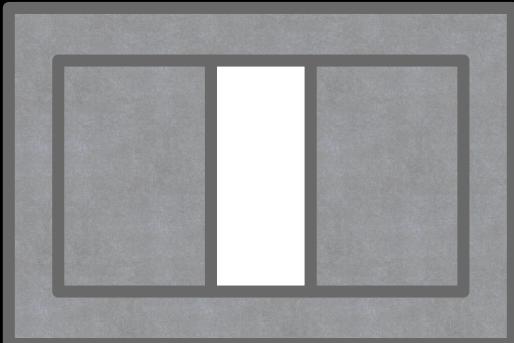
```
    print('A True or B True or Both False')
```

```
windll.user32.MessageBoxW(0,
```

```
'A True, B True, Both False', 'NAND', 0)
```

```
input('Press Enter to Exit')
```

A	B	=
0	0	= 1
0	1	= 1
1	0	= 1
1	1	= 0



Activates Only if
A True or B True,
or Both are False

OR

```
from ctypes import*
```

```
A = 1
```

```
B = 0
```

```
if (A == 1 or B == 1):
```

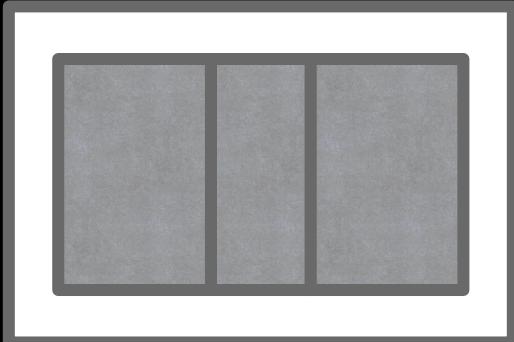
```
    print('One or Both are True')
```

```
    windll.user32.MessageBoxW(0,
```

```
        'One or Both True', 'OR Gate', 0)
```

```
    input('Press Enter to Exit')
```

A	B	=
0	0	= 0
0	1	= 1
1	0	= 1
1	1	= 1



Activates Only if
One or Both
are True

NOR

```
from ctypes import*
```

```
A = 0
```

```
B = 0
```

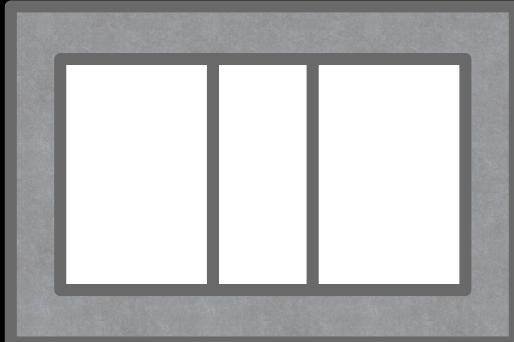
```
if (A == 0 and B == 0):
```

```
    print('Both are False')
```

```
windll.user32.MessageBoxW(0,  
'Both are False', 'NOR Gate', 0)
```

```
input('Press Enter to Exit')
```

A	B	=
0	0	1
0	1	0
1	0	0
1	1	0



Activates Only if
Both are False

XOR

```
from ctypes import*
```

```
A = 1
```

```
B = 0
```

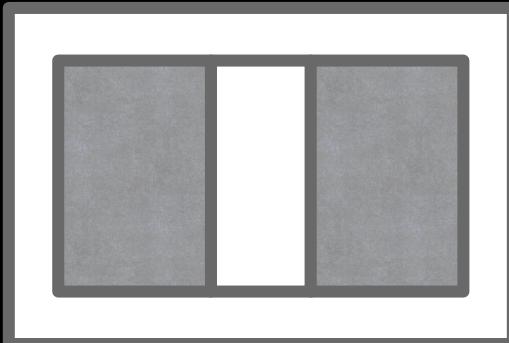
```
if ((A == 0 and B == 1) or  
    (A == 1 and B == 0)):
```

```
    print('A True or B True')
```

```
windll.user32.MessageBoxW(0,  
'A True or B True', 'XOR Gate', 0)
```

```
input('Press Enter to Exit')
```

A	B	=
0	0	0
0	1	1
1	0	1
1	1	0

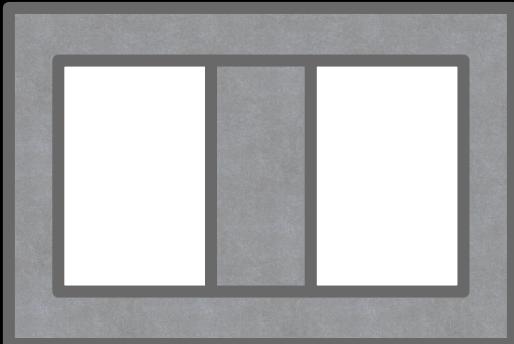


Activates Only if
A True or B True

XNOR

```
from ctypes import*
A = 0
B = 0
if ((A == 0 and B == 0) or
    (A == 1 and B == 1)):
    print('Both True or Both False')
    windll.user32.MessageBoxW(0,
    'Both True or False', 'XNOR Gate', 0)
input('Press Enter to Exit')
```

A	B	=
0	0	1
0	1	0
1	0	0
1	1	1



**Activates Only if
Both are True or
Both are False**

CONVERSE IMPLICATION

```
from ctypes import*
```

```
A = 0
```

```
B = 0
```

```
if (A == 1 or B == 0):
```

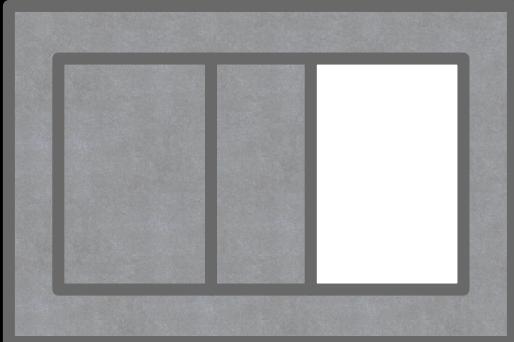
```
    print('Both True, Both False, A True')
```

```
windll.user32.MessageBoxW(0,
```

```
'Both True, False, A True', 'Ci Gate', 0)
```

```
input('Press Enter to Exit')
```

A	B	=
0	0	= 1
0	1	= 0
1	0	= 1
1	1	= 1



**Activates Only if
Both True or Both
False or A True**

CONVERSE NON IMPLICATION

```
from ctypes import*
```

```
A = 0
```

```
B = 1
```

```
if (A == 0 and B == 1):
```

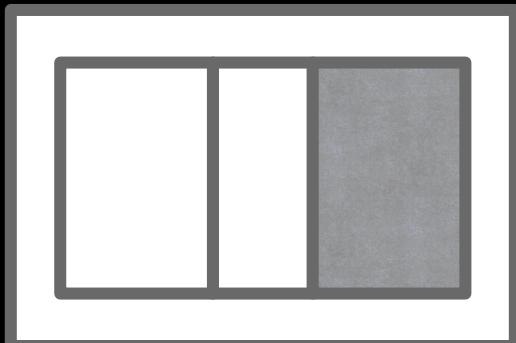
```
    print('B True')
```

```
windll.user32.MessageBoxW(0,
```

```
'B True', 'CNI Gate', 0)
```

```
input('Press Enter to Exit')
```

A	B	
0	0	= 0
0	1	= 1
1	0	= 0
1	1	= 0



Activates Only if
B True

MATERIAL IMPLICATION

```
from ctypes import*
```

```
A = 0
```

```
B = 0
```

```
if (A == 0 or B == 1):
```

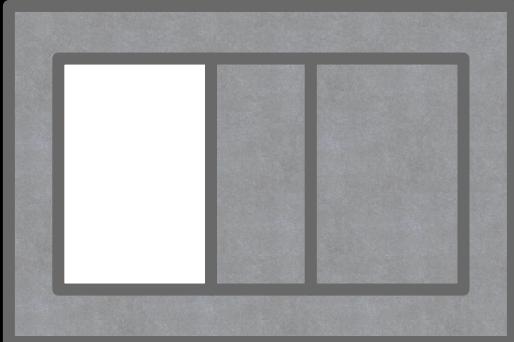
```
    print('Both True, Both False, B True')
```

```
windll.user32.MessageBoxW(0,
```

```
'Both True, False, B True', 'Mi Gate', 0)
```

```
input('Press Enter to Exit')
```

A	B	=
0	0	= 1
0	1	= 1
1	0	= 0
1	1	= 1



**Activates Only if
Both True or Both
False or B True**

MATERIAL NON IMPLICATION

```
from ctypes import*
```

```
A = 1
```

```
B = 0
```

```
if (A == 1 and B == 0):
```

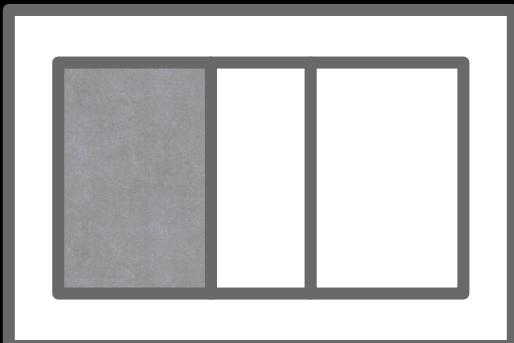
```
    print('A True')
```

```
windll.user32.MessageBoxW(0,
```

```
'A True', 'MNi Gate', 0)
```

```
input('Press Enter to Exit')
```

A	B	=
0	0	0
0	1	0
1	0	1
1	1	0



Activates Only if
A True

Right Projection

```
from ctypes import*
```

```
A = 1
```

```
B = 1
```

```
if (B == 1):
```

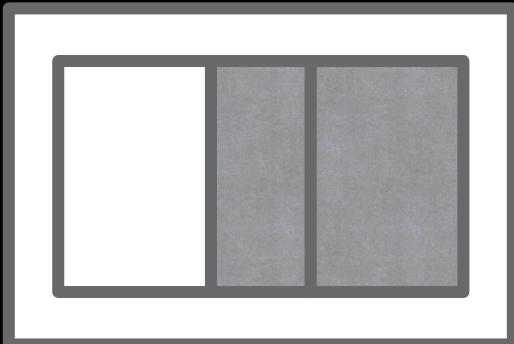
```
    print('Both True or B True')
```

```
windll.user32.MessageBoxW(0,
```

```
'Both True or B True', 'RP Gate', 0)
```

```
input('Press Enter to Exit')
```

A	B	
0	0	= 0
0	1	= 1
1	0	= 0
1	1	= 1



Activates Only if
Both True or
B True

Right Complementation

```
from ctypes import*
```

```
A = 0
```

```
B = 0
```

```
if (B == 0):
```

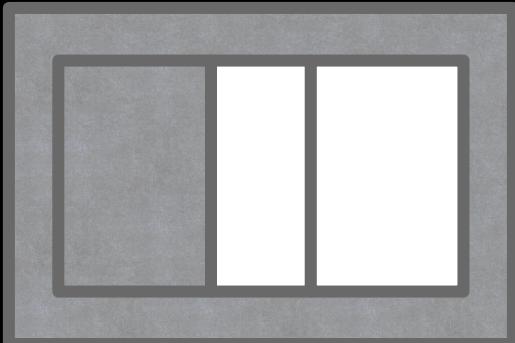
```
    print('Both False or A True')
```

```
windll.user32.MessageBoxW(0,
```

```
'Both False or A True', 'RC Gate', 0)
```

```
input('Press Enter to Exit')
```

A	B	=
0	0	1
0	1	0
1	0	1
1	1	0



Activates Only if
Both are False or
A is True

Left Projection

```
from ctypes import*
```

```
A = 1
```

```
B = 0
```

```
if (A == 1):
```

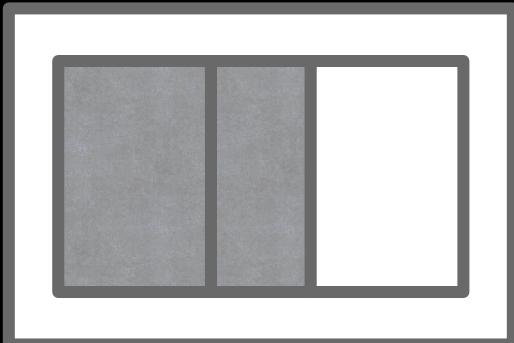
```
    print('Both True or A True')
```

```
windll.user32.MessageBoxW(0,
```

```
'Both True or A True', 'LP Gate', 0)
```

```
input('Press Enter to Exit')
```

A	B	=
0	0	0
0	1	0
1	0	1
1	1	1

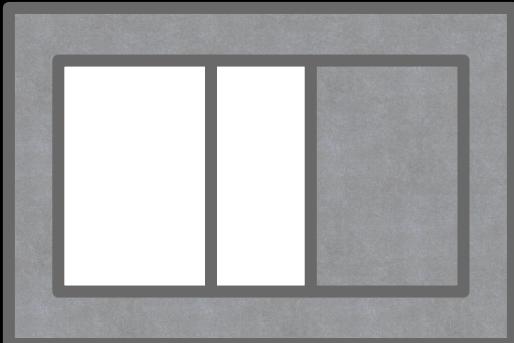


Activates Only if
Both True
or A True

Left Complementation

```
from ctypes import*
A = 0
B = 0
if (A == 0):
    print('Both False or B False')
    windll.user32.MessageBoxW(0,
    'Both False or B False', 'LC Gate', 0)
input('Press Enter to Exit')
```

A	B
0	0 = 1
0	1 = 1
1	0 = 0
1	1 = 0



Activates Only if
Both are False or
B is True

CONTRADICTION

```
from ctypes import*
```

```
A = 0
```

```
B = 0
```

```
if ((A == 0 or B == 0) or  
    (A == 1 or B == 1)):
```

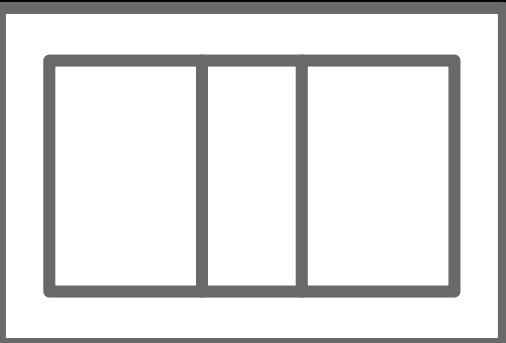
```
    print('Always False')
```

```
windll.user32.MessageBoxW(0,  
'Always False', 'Contradiction Gate', 0)
```

```
input('Press Enter to Exit')
```

A	B	=
0	0	0
0	1	0
1	0	0
1	1	0

For
Tutorial
Purposes



Activates Neg Text if
A is True or False or
B is True or False

TAUTOLOGY

```
from ctypes import*
```

```
A = 0
```

```
B = 0
```

```
if ((A == 0 or B == 0) or  
    (A == 1 or B == 1)):
```

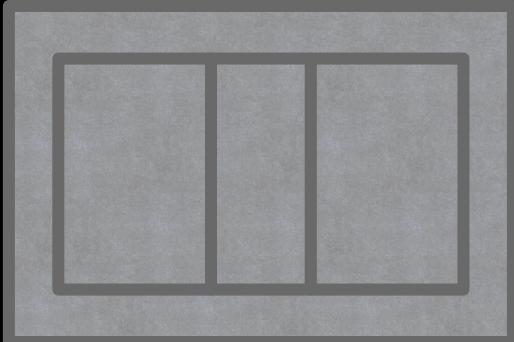
```
    print('Always True')
```

```
windll.user32.MessageBoxW(0,  
'Always True', 'Tautology Gate', 0)
```

```
input('Press Enter to Exit')
```

A	B	=
0	0	= 1
0	1	= 1
1	0	= 1
1	1	= 1

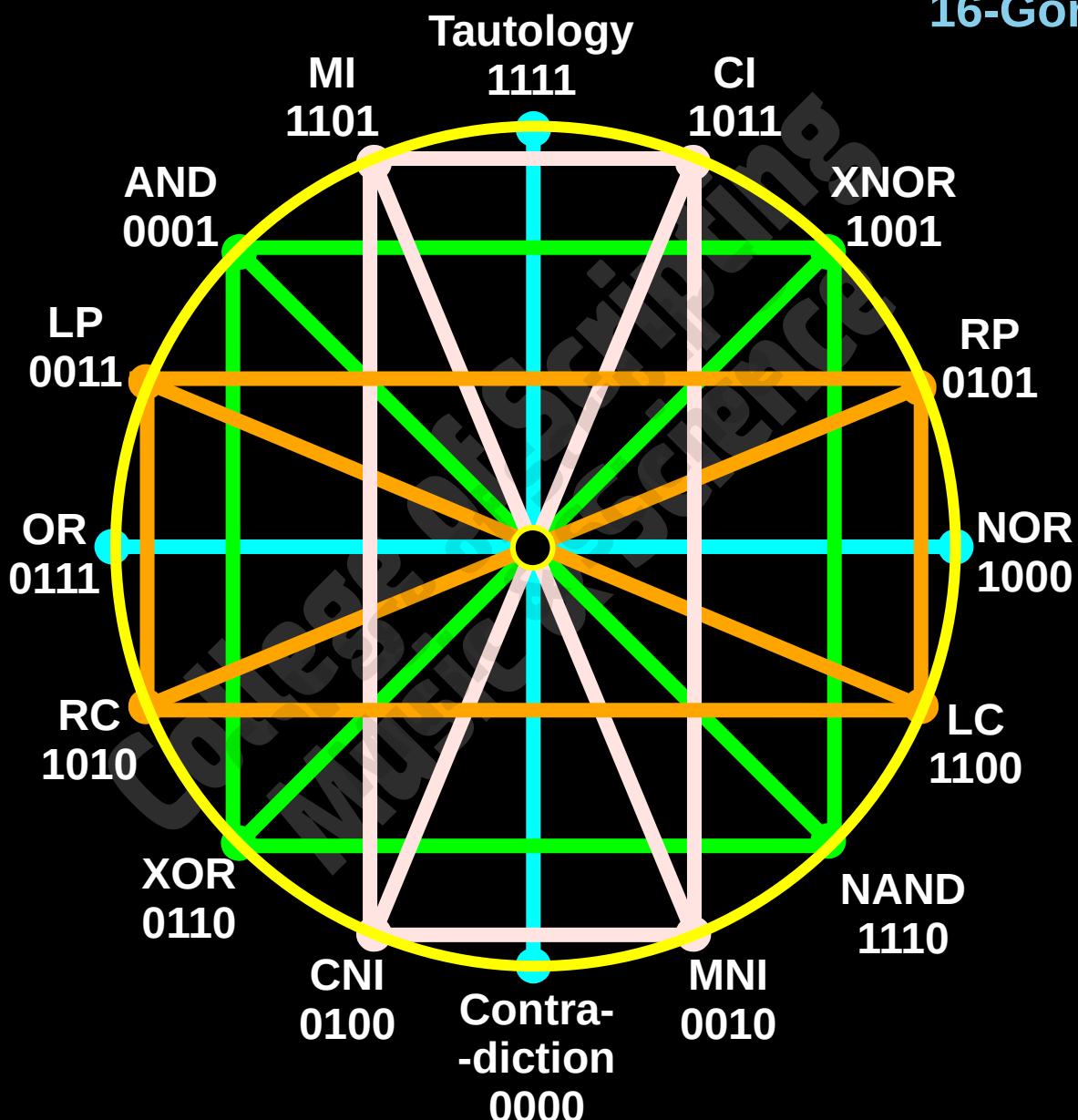
For
Tutorial
Purposes



Activates if
A is True or False or
B is True or False

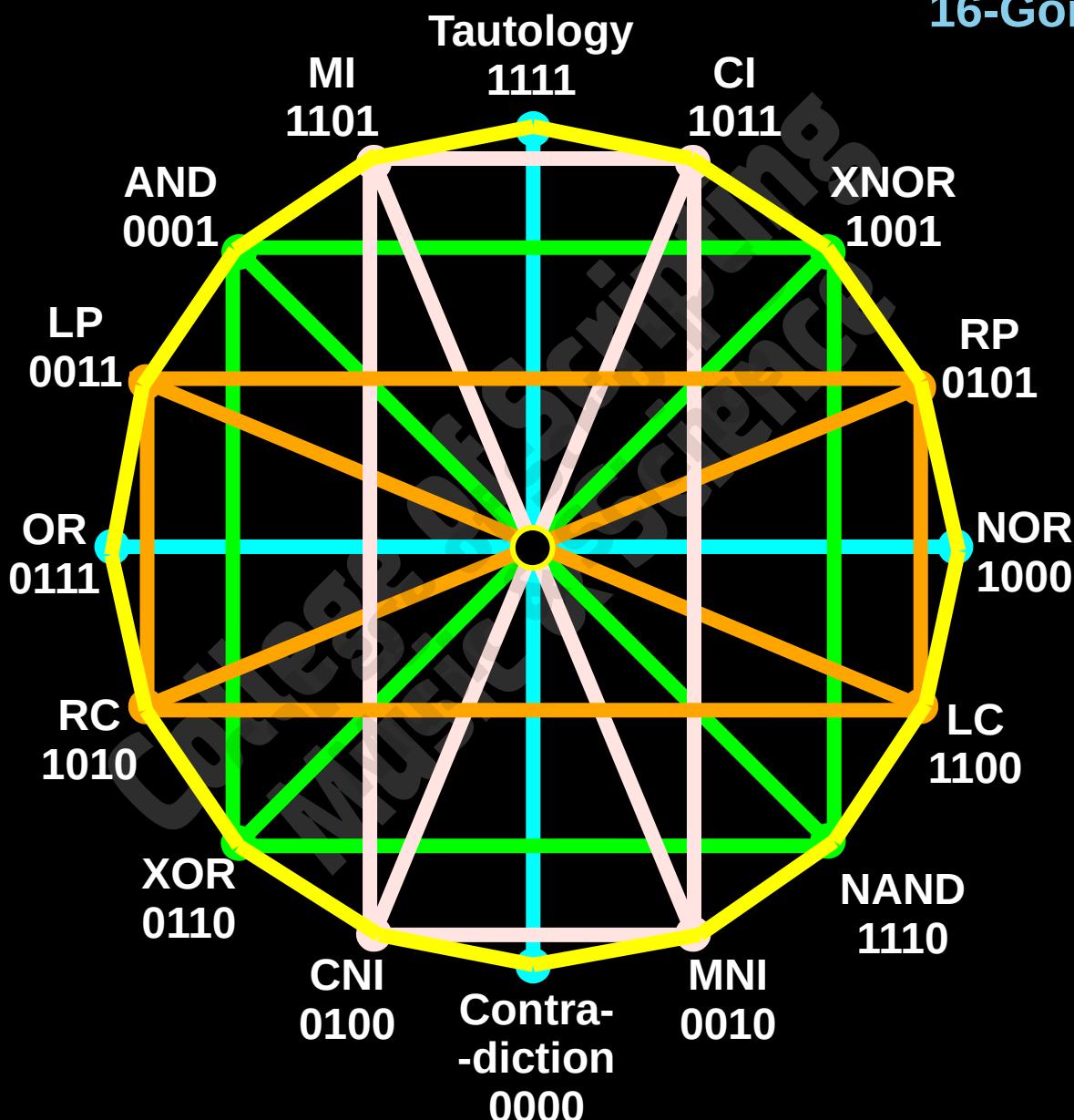
Artificial Intelligence System

16-Gon



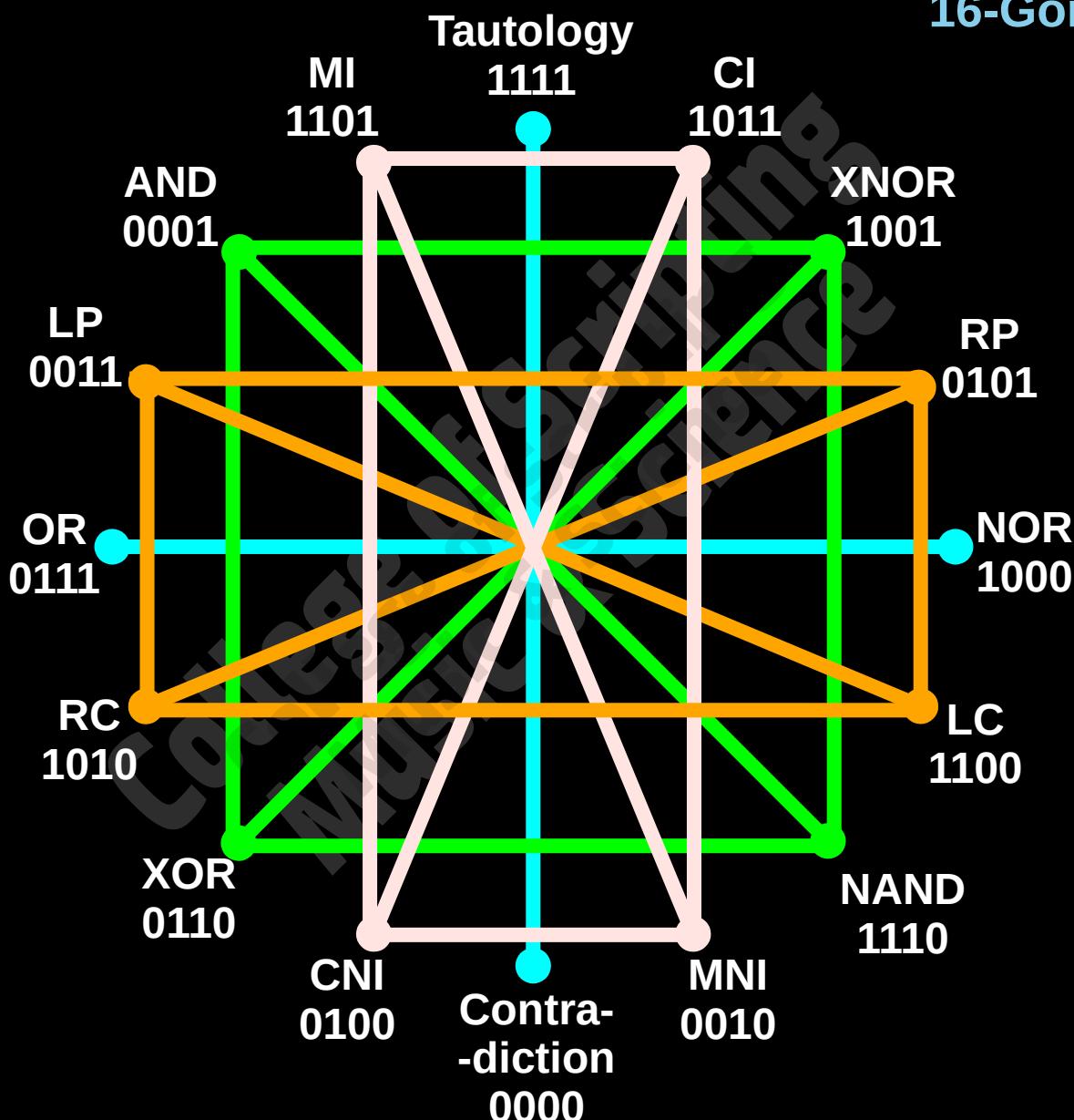
Artificial Intelligence System

16-Gon



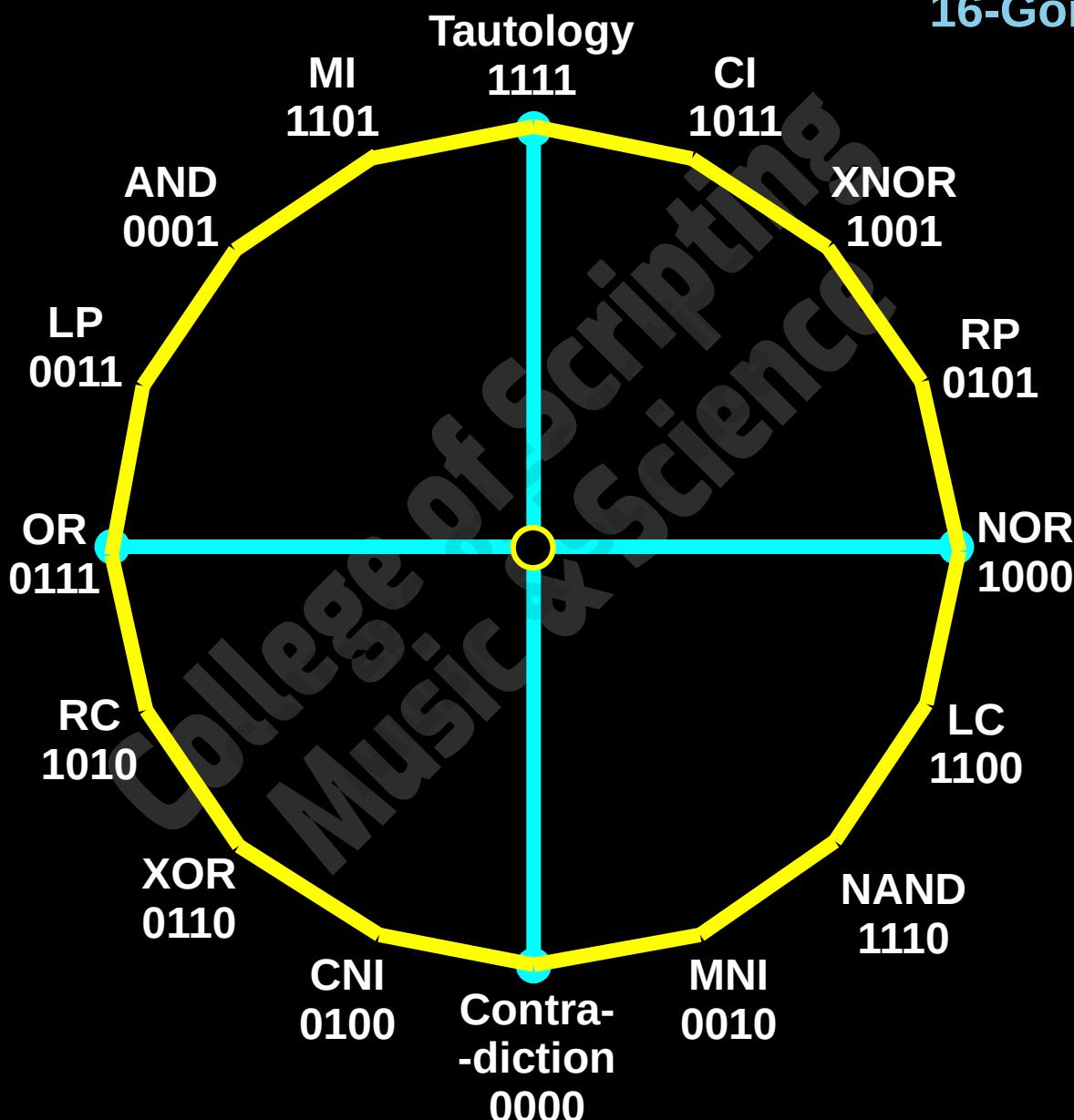
Artificial Intelligence System

16-Gon



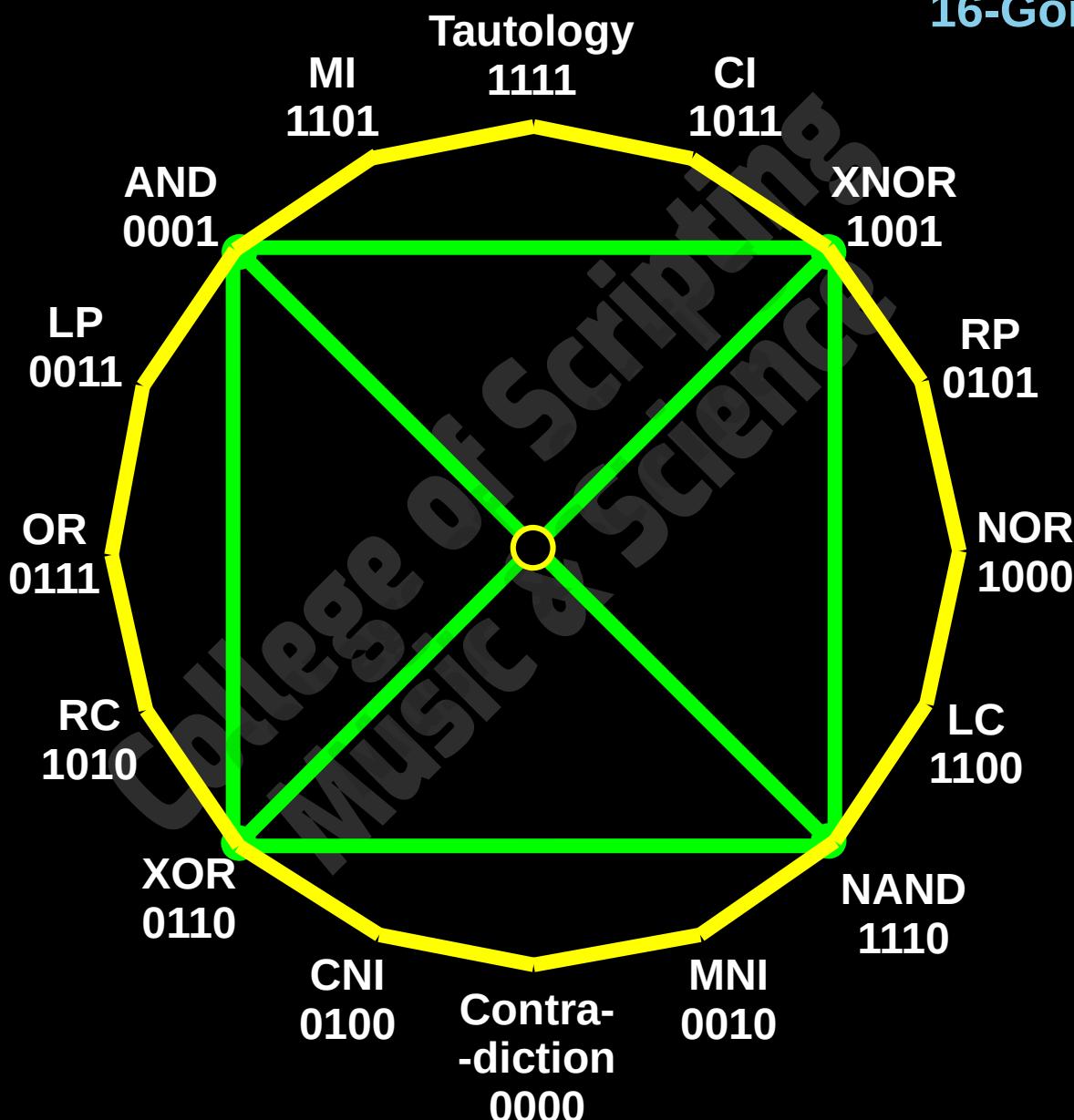
Artificial Intelligence System

16-Gon



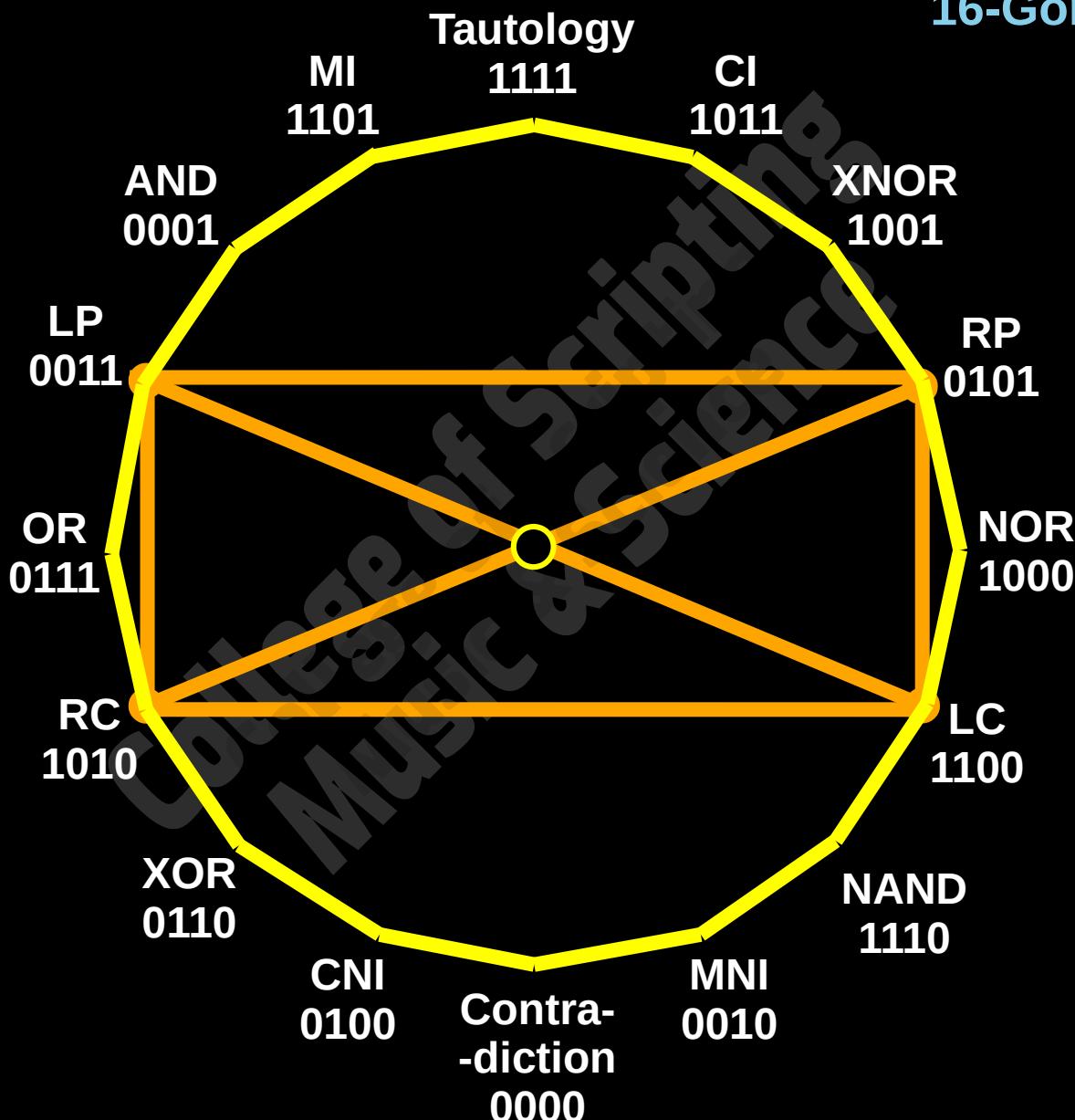
Artificial Intelligence System

16-Gon



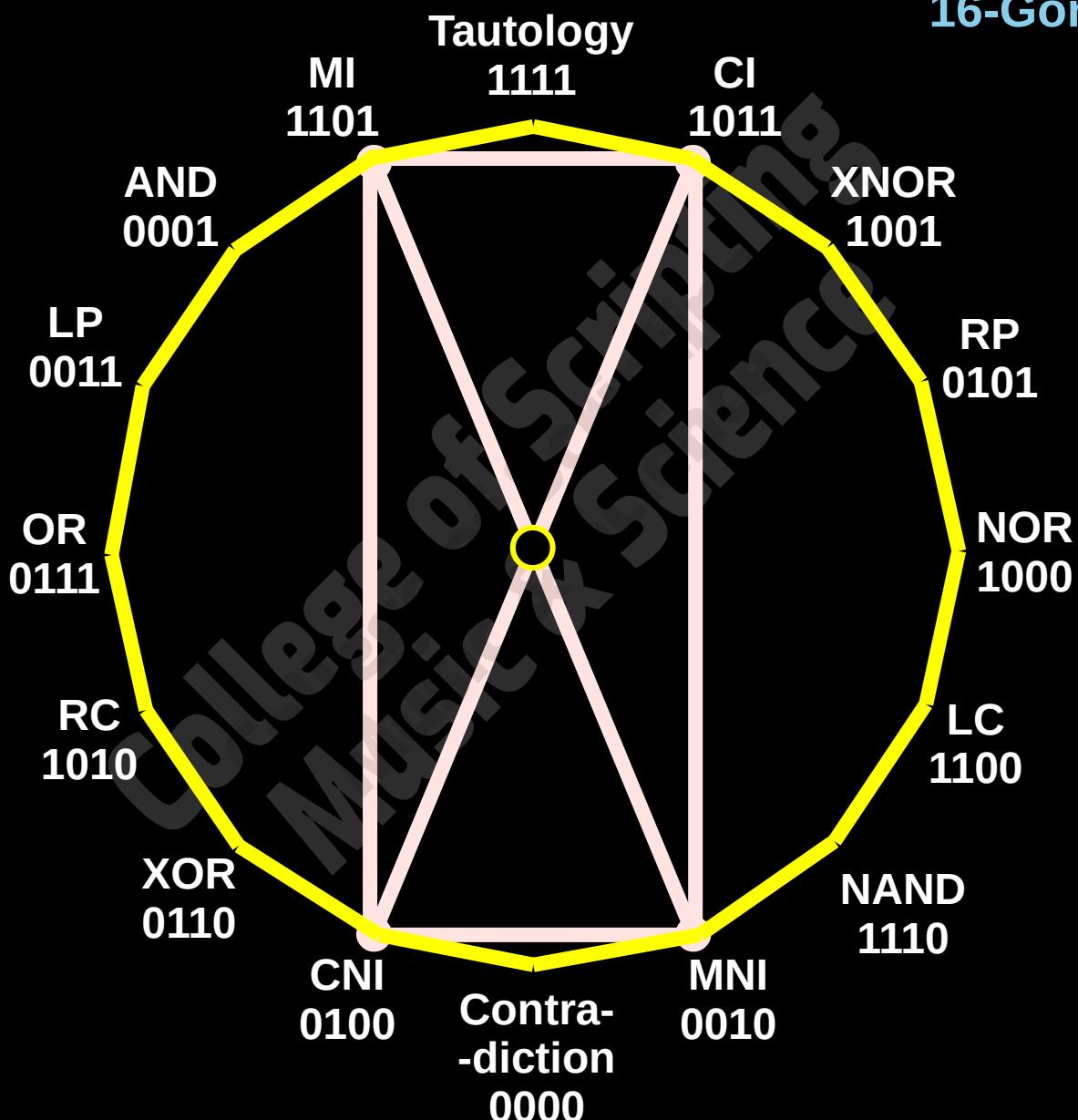
Artificial Intelligence System

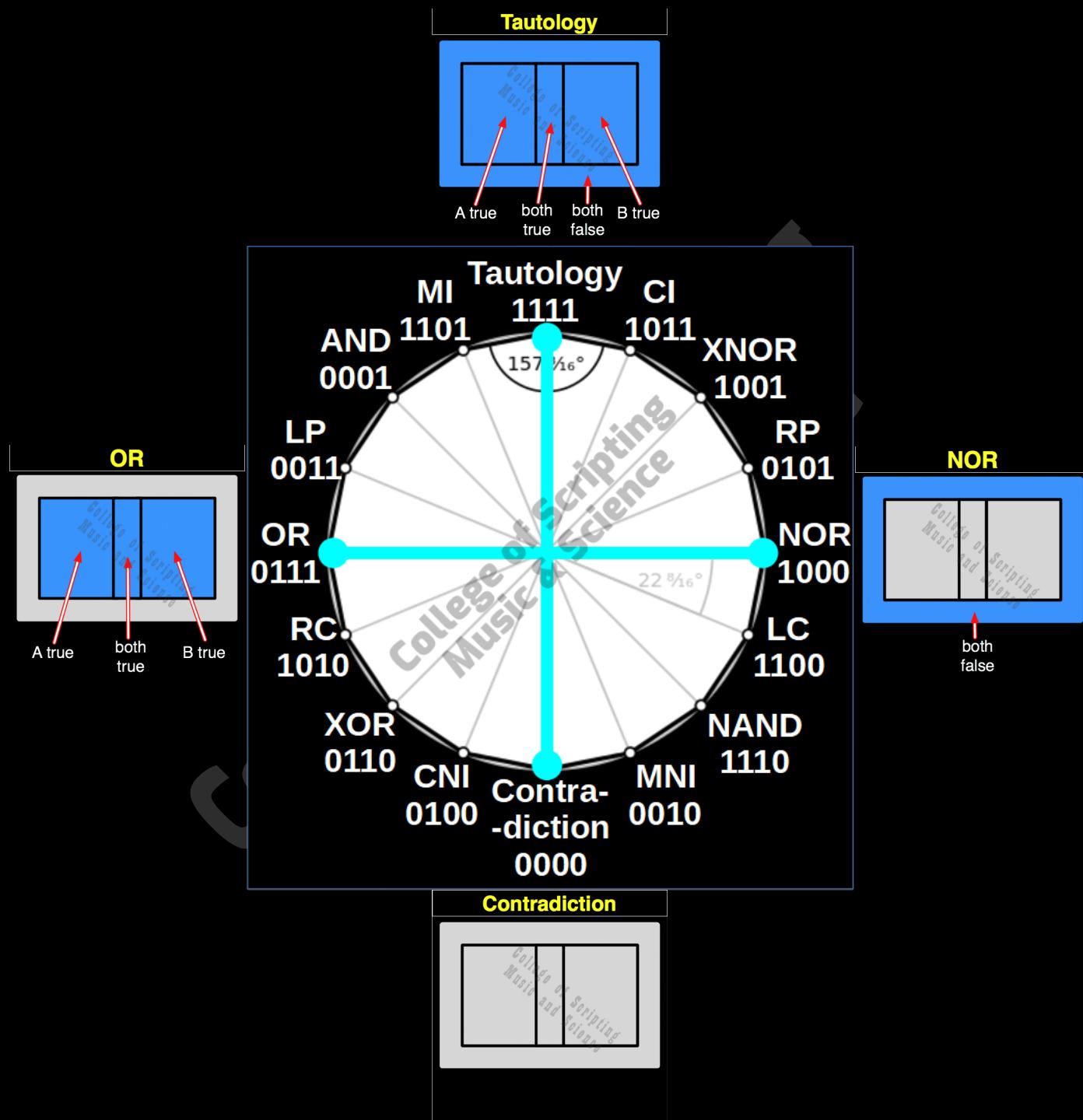
16-Gon

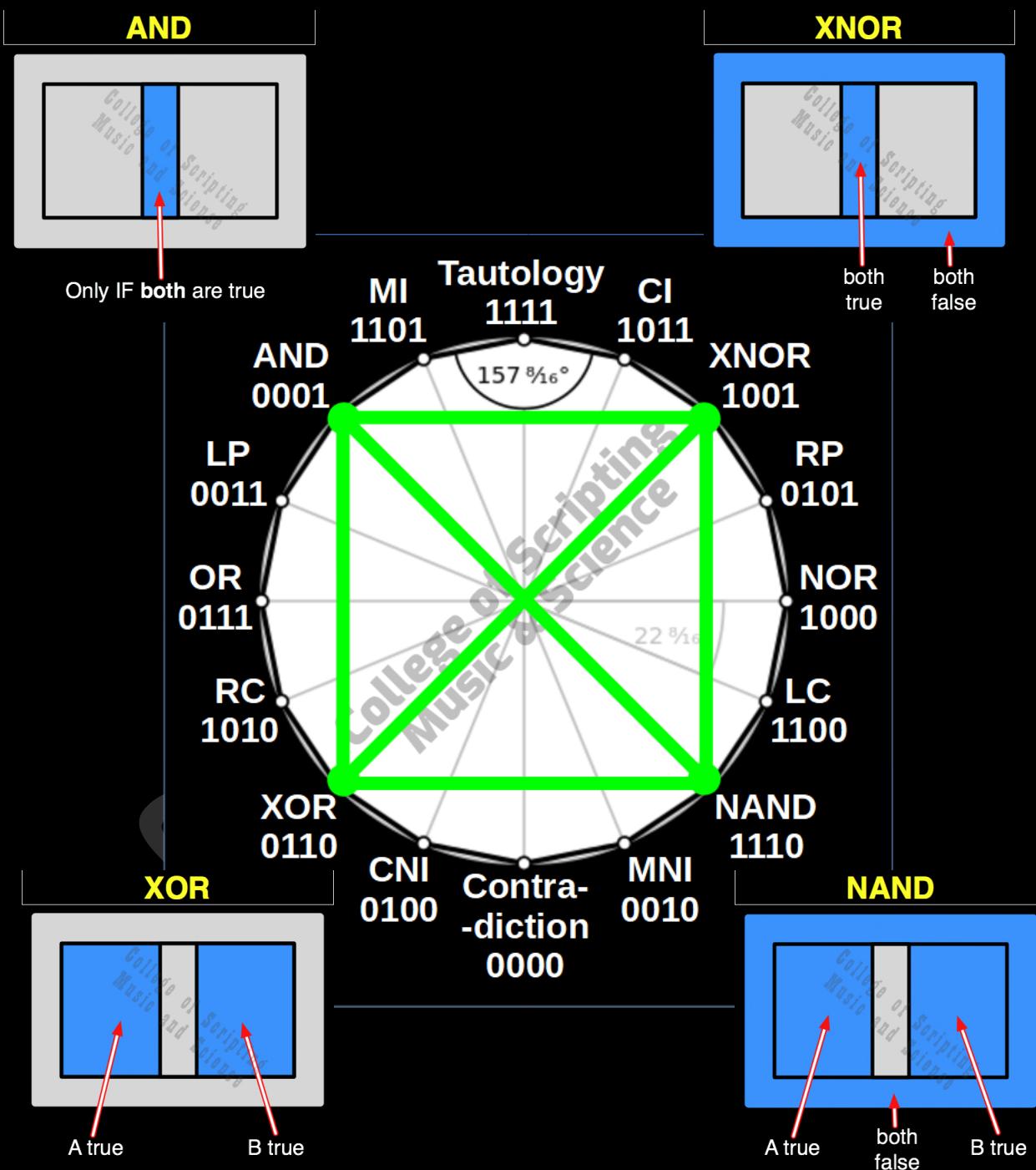


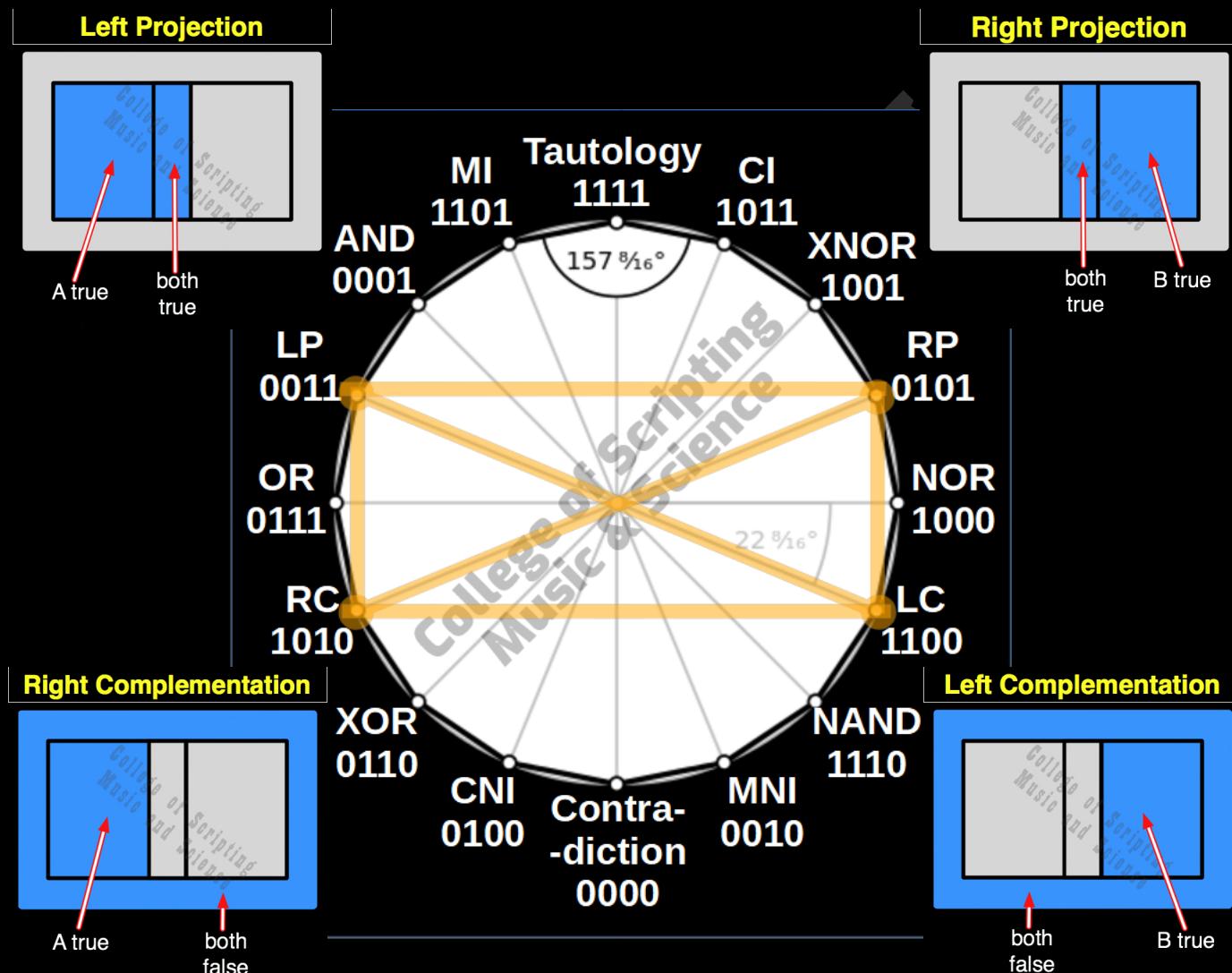
Artificial Intelligence System

16-Gon

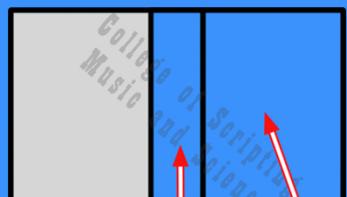








Material Implication



both
false both
true B true

MI
1101 Tautology
1111

AND
0001

LP
0011

OR
0111

RC
1010

XOR
0110

CNI
0100

Contra-
diction
0000

CI
1011

XNOR
1001

RP
0101

NOR
1000

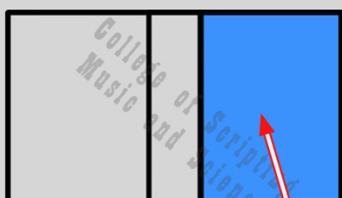
LC
1100

NAND
1110

MNI
0010

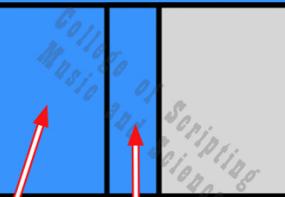
Material NonImplication

Converse NonImplication

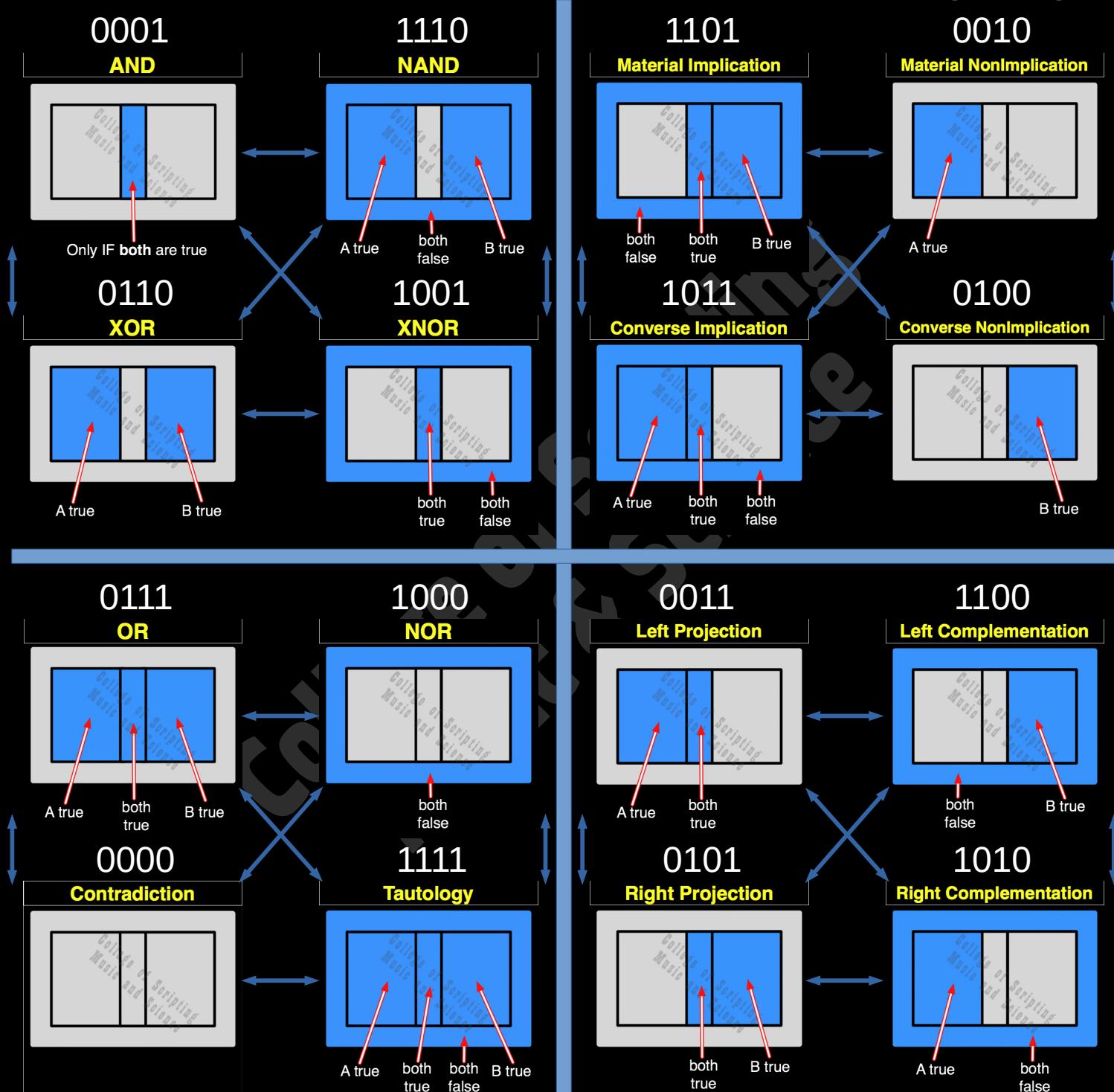


B true

A true



A true
both
true both
false



Reference

College of Scripting
Music & Science

MessageBox user32 options

Alert sound

ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 16)



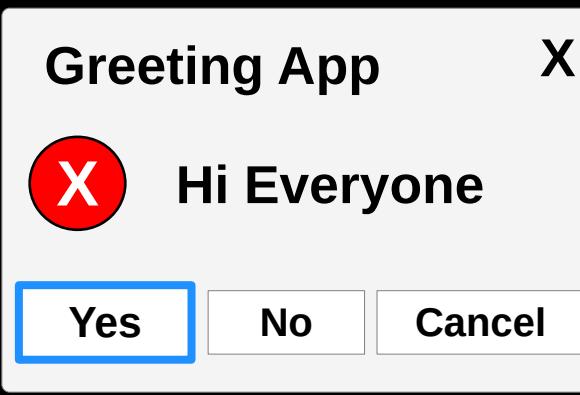
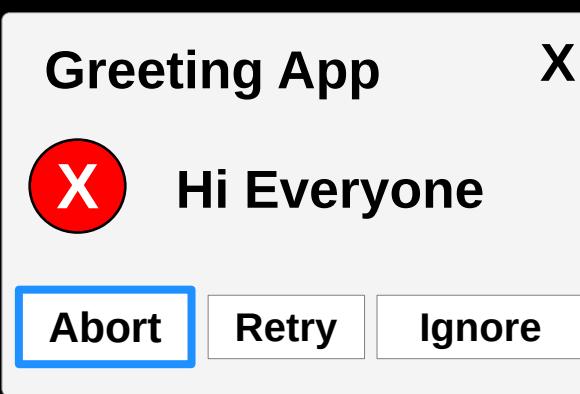
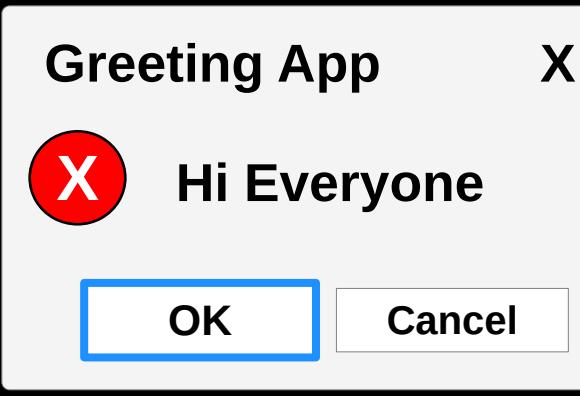
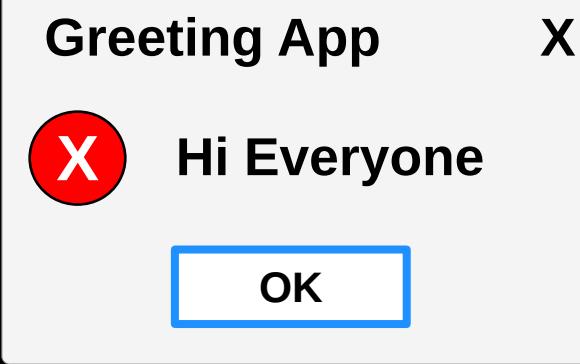
ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 17)

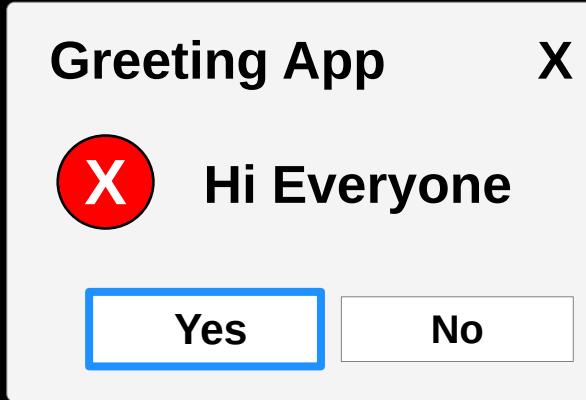


ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 18)

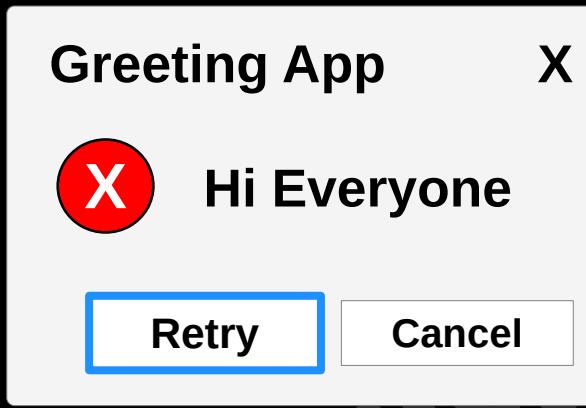


ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 19)

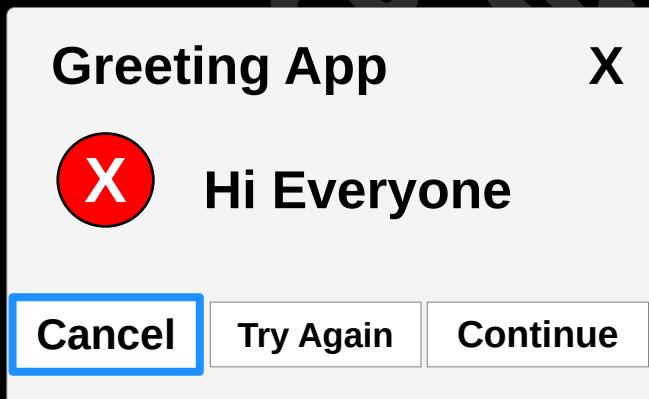




**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 20)**

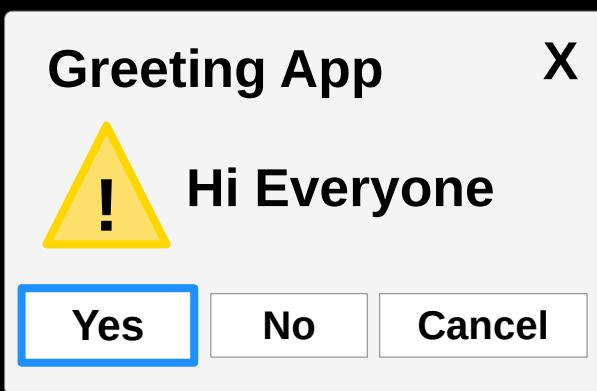
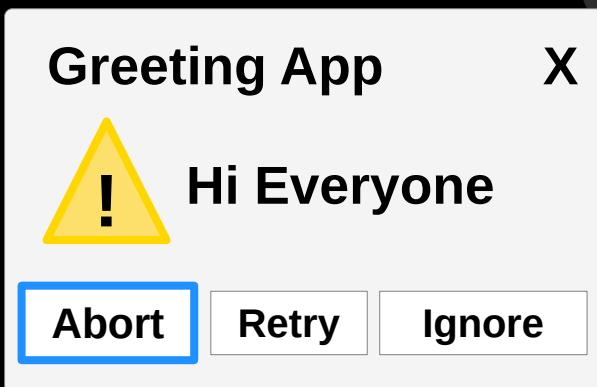
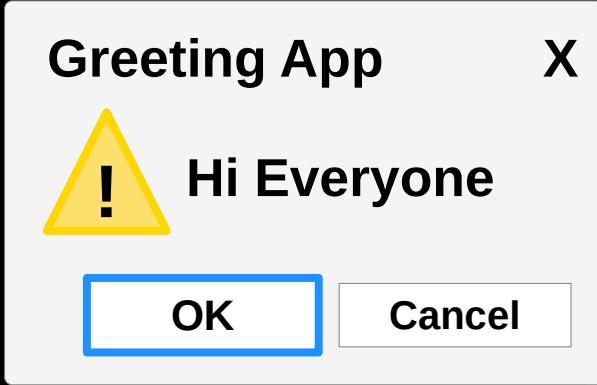
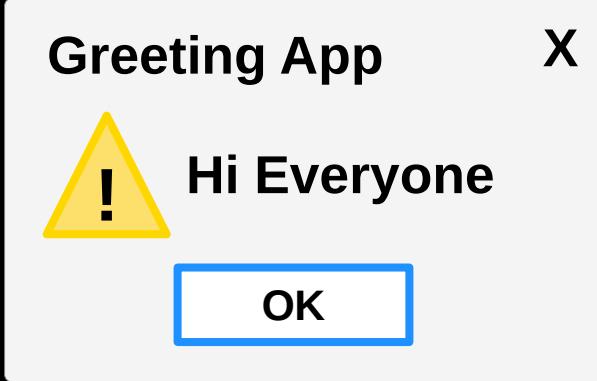


**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 21)**



**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 22)**





ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 48)

↑

ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 49)

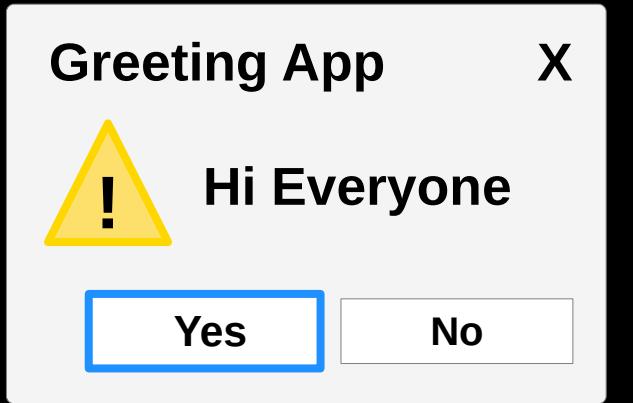
↑

ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 50)

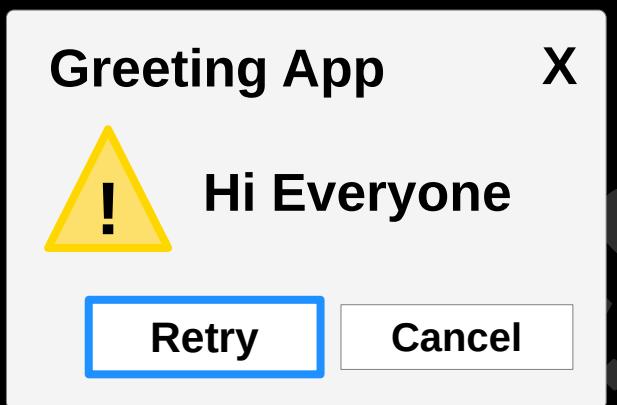
↑

ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 51)

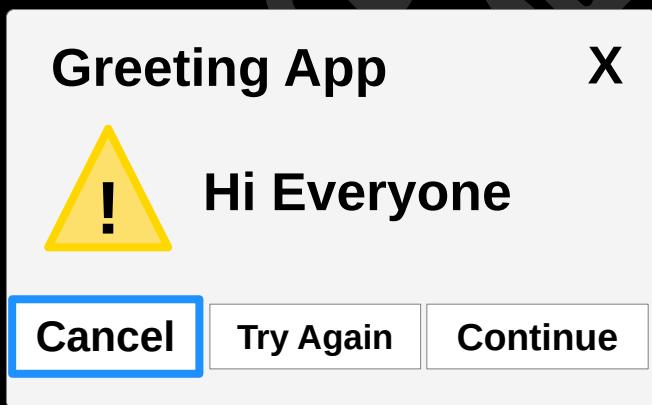
↑



`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 52)`



`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 53)`



`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 54)`



Greeting App

X



Hi Everyone

OK

Greeting App

X



Hi Everyone

OK

Cancel

Greeting App

X



Hi Everyone

Abort

Retry

Ignore

Greeting App

X



Hi Everyone

Yes

No

Cancel

Alert sound

ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 64)



ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 65)

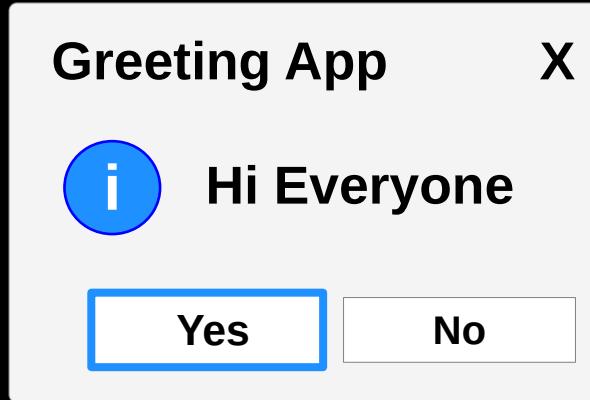


ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 66)

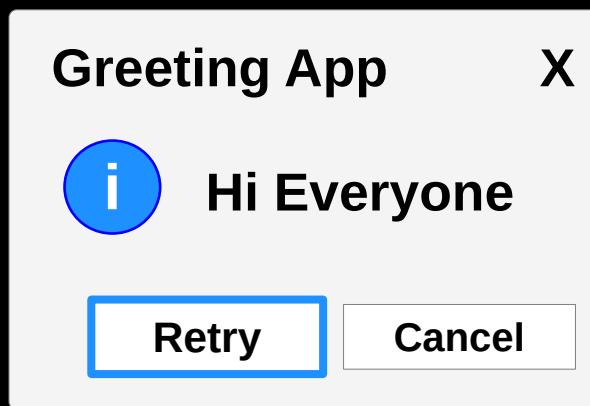


ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 67)

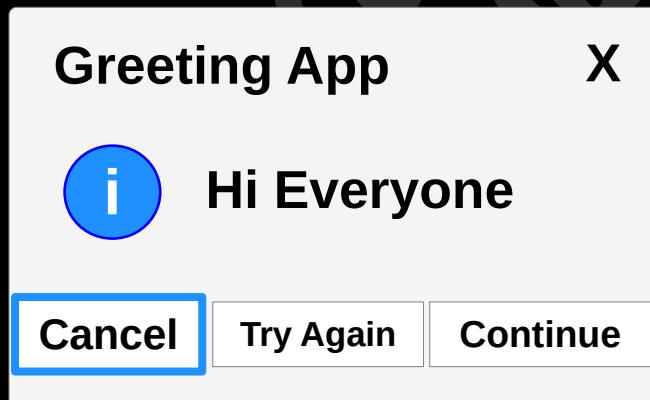




ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 68)



ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 69)



ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 70)



Greeting App

X

Hi Everyone

OK

Greeting App

X

Hi Everyone

OK

Cancel

Greeting App

X

Hi Everyone

Abort

Retry

Ignore

Greeting App

X

Hi Everyone

Yes

No

Cancel

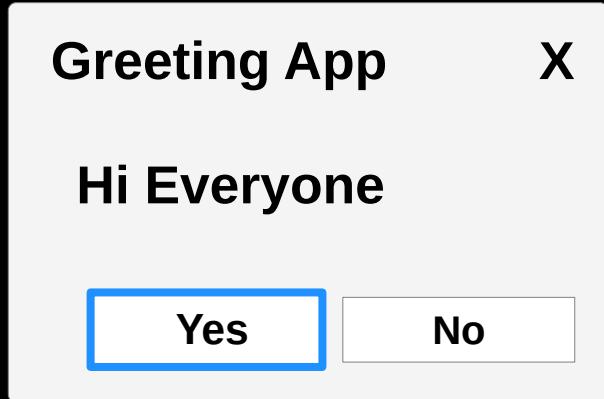
Alert sound

ctypes.windll.user32.
**MessageBoxW(0,
ourText, ourTitle, 80)**

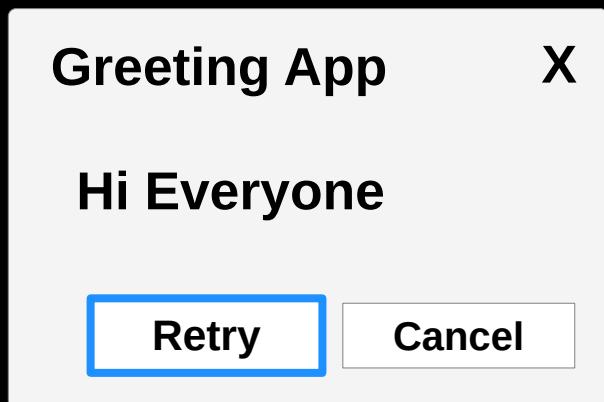
ctypes.windll.user32.
**MessageBoxW(0,
ourText, ourTitle, 81)**

ctypes.windll.user32.
**MessageBoxW(0,
ourText, ourTitle, 82)**

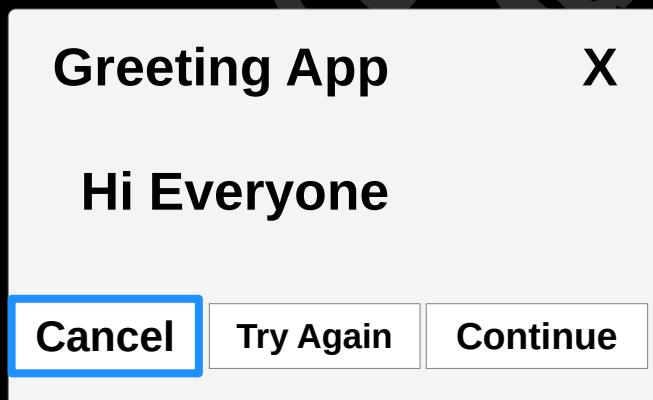
ctypes.windll.user32.
**MessageBoxW(0,
ourText, ourTitle, 83)**



ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 84)



ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 85)



ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 86)



Greeting App

X

Hi Everyone

OK

Greeting App

X

Hi Everyone

OK

Cancel

Greeting App

X

Hi Everyone

Abort

Retry

Ignore

Greeting App

X

Hi Everyone

Yes

No

Cancel

No alert sound

**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 256)**



**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 257)**

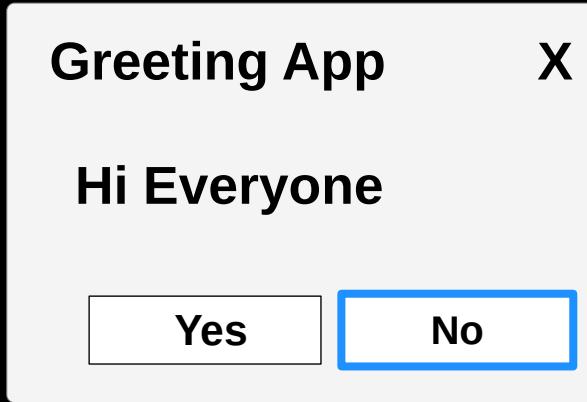


**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 258)**

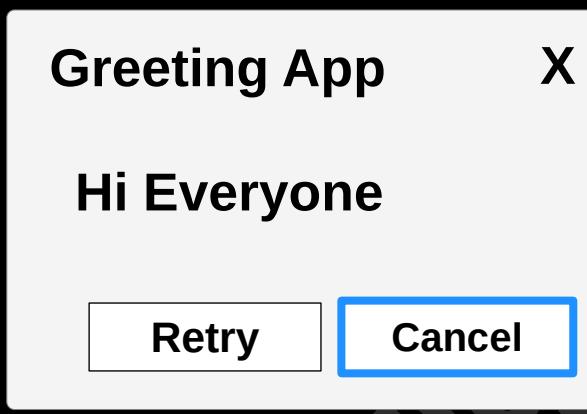


**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 259)**

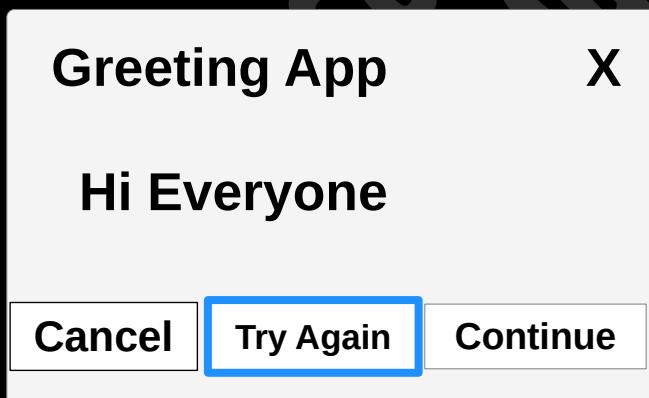




`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 260)`

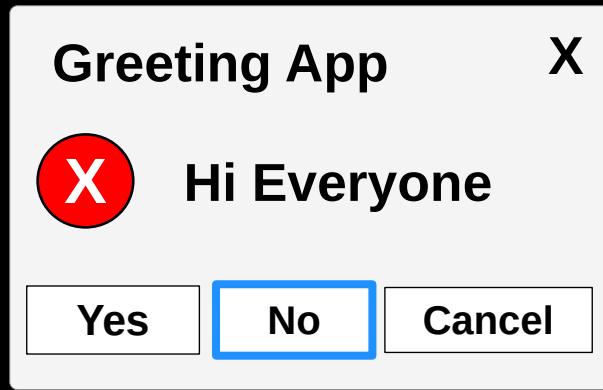
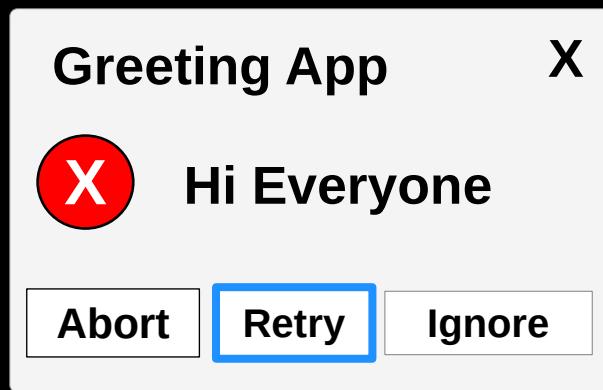
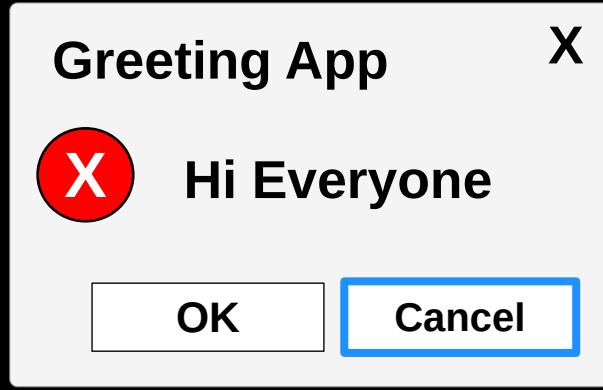
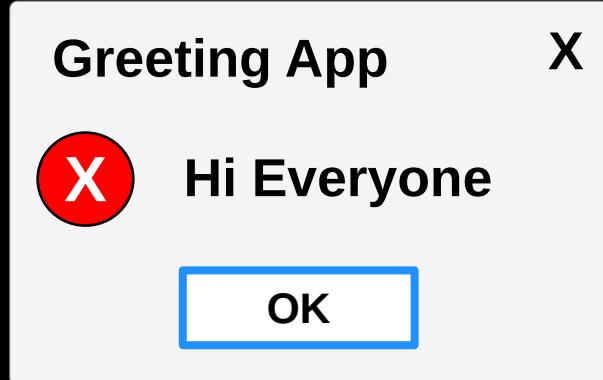


`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 261)`



`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 262)`





ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 272)

A yellow arrow points upwards from the text 'College of Scripting Music & Science' at the bottom of the slide to the numerical value '272' in the third line of the code snippet.

ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 273)

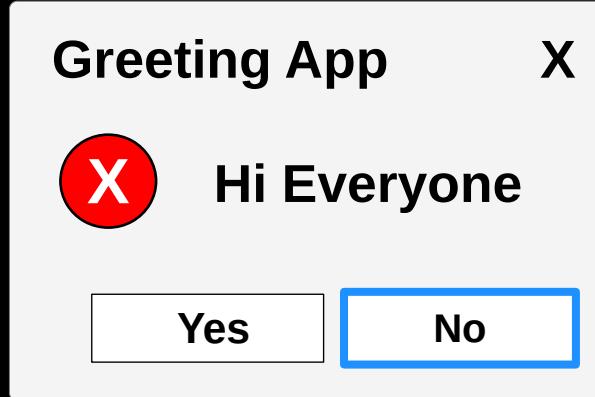
A yellow arrow points upwards from the text 'College of Scripting Music & Science' at the bottom of the slide to the numerical value '273' in the fourth line of the code snippet.

ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 274)

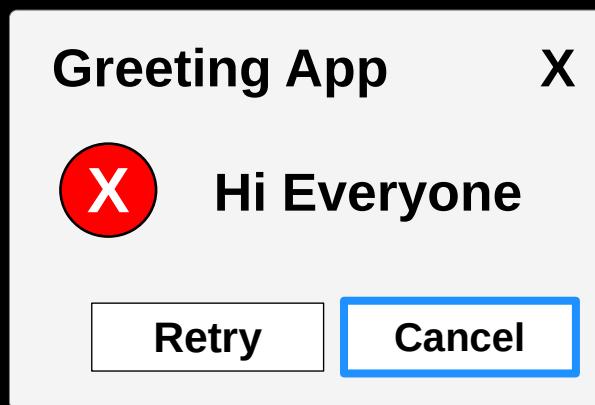
A yellow arrow points upwards from the text 'College of Scripting Music & Science' at the bottom of the slide to the numerical value '274' in the fifth line of the code snippet.

ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 275)

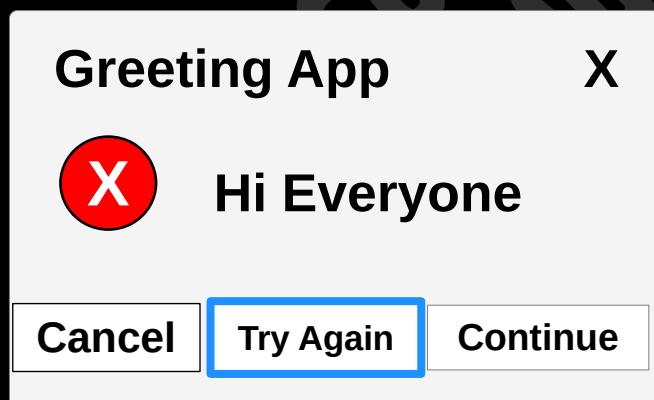
A yellow arrow points upwards from the text 'College of Scripting Music & Science' at the bottom of the slide to the numerical value '275' in the sixth line of the code snippet.



`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 276)`



`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 277)`



`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 278)`



Greeting App

X



Hi Everyone

OK

Greeting App

X



Hi Everyone

OK

Cancel

Greeting App

X



Hi Everyone

Abort

Retry

Ignore

Greeting App

X



Hi Everyone

Yes

No

Cancel

No Alert sound

**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 288)**



**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 289)**

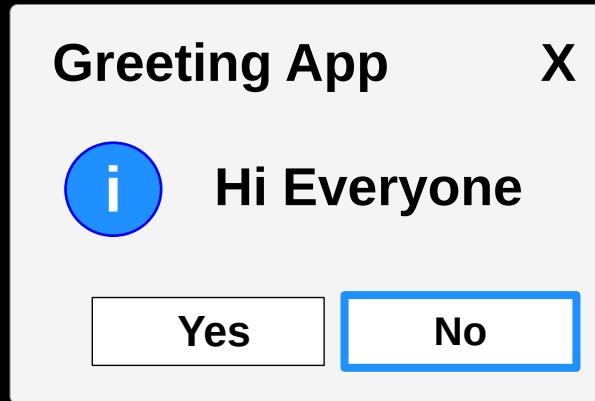


**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 290)**

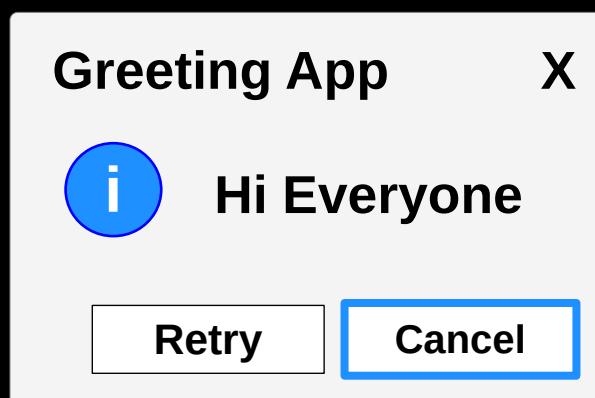


**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 291)**

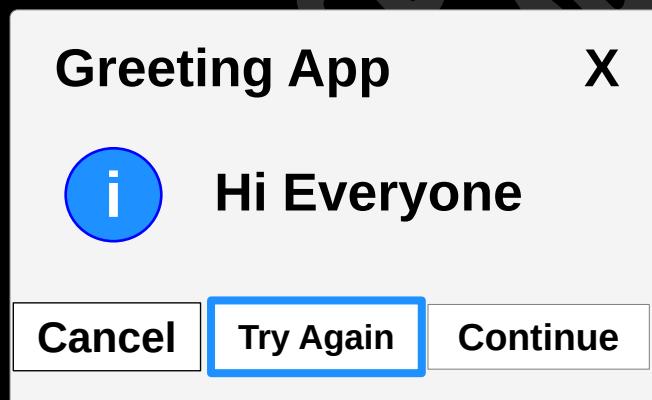




`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 292)`



`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 293)`



`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 294)`



Greeting App

X



Hi Everyone

OK

Greeting App

X



Hi Everyone

OK

Cancel

Greeting App

X



Hi Everyone

Abort

Retry

Ignore

Greeting App

X



Hi Everyone

Yes

No

Cancel

Alert sound

ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 304)



ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 305)

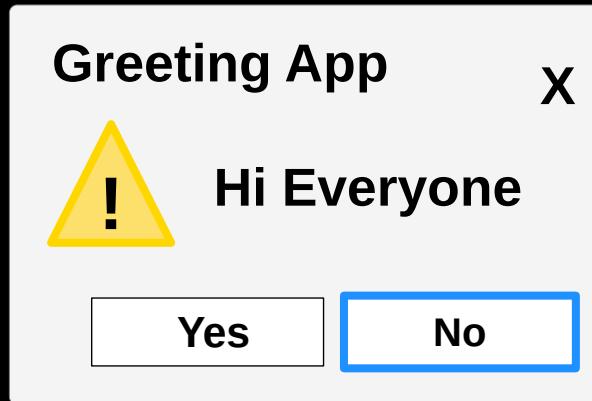


ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 306)

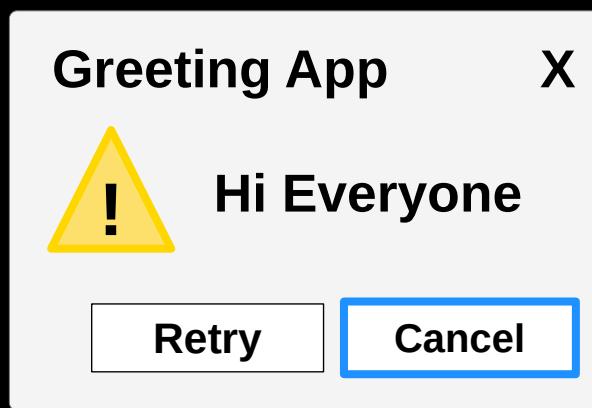


ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 307)

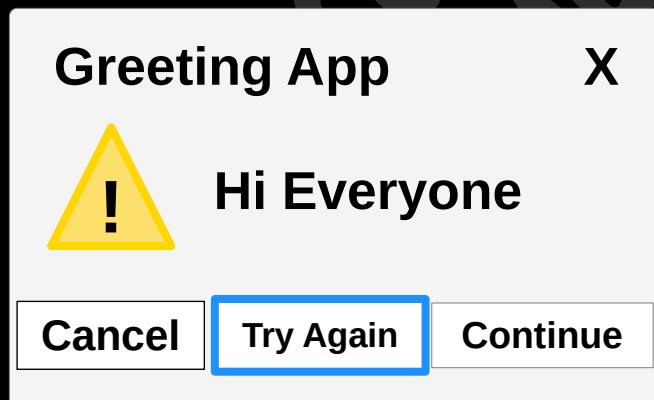




`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 308)`



`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 309)`



`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 310)`



Alert sound

ctypes.windll.user32.
**MessageBoxW(0,
ourText, ourTitle, 320)**



ctypes.windll.user32.
**MessageBoxW(0,
ourText, ourTitle, 321)**



ctypes.windll.user32.
**MessageBoxW(0,
ourText, ourTitle, 322)**



ctypes.windll.user32.
**MessageBoxW(0,
ourText, ourTitle, 323)**



Greeting App

X

i Hi Everyone

OK

Greeting App

X

i Hi Everyone

OK

Cancel

Greeting App

X

i Hi Everyone

Abort

Retry

Ignore

Greeting App

X

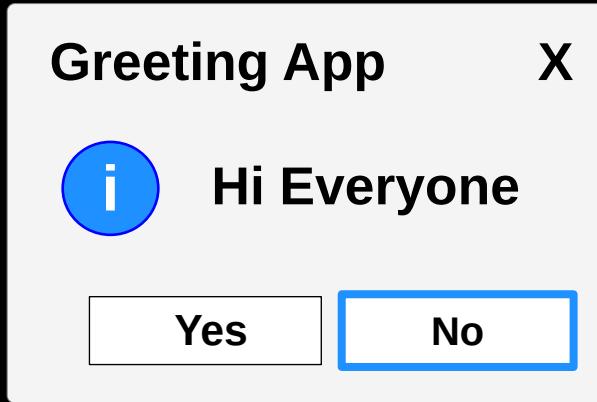
i Hi Everyone

Yes

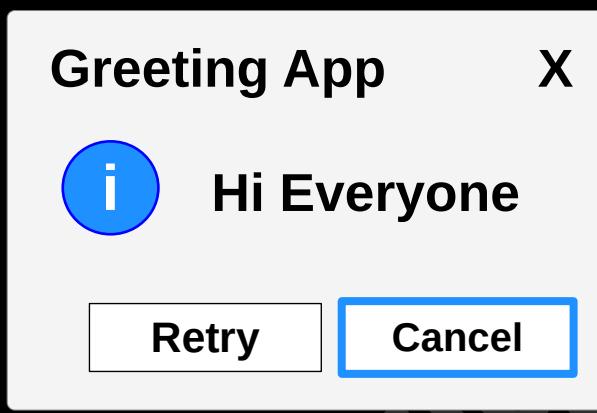
No

Cancel

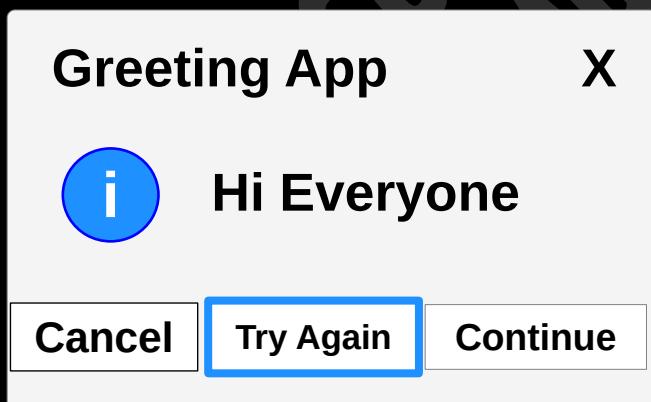
College of Scripting Music & Science



`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 324)`



`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 325)`



`ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 326)`



Alert sound

**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 336)**



**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 337)**



**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 338)**



College of Scripting Music & Science

**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 339)**



Greeting App

X

Hi Everyone

OK

Greeting App

X

Hi Everyone

OK

Cancel

Greeting App

X

Hi Everyone

Abort

Retry

Ignore

Greeting App

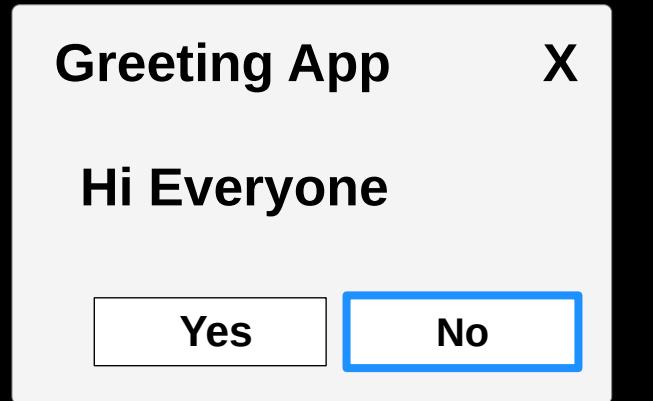
X

Hi Everyone

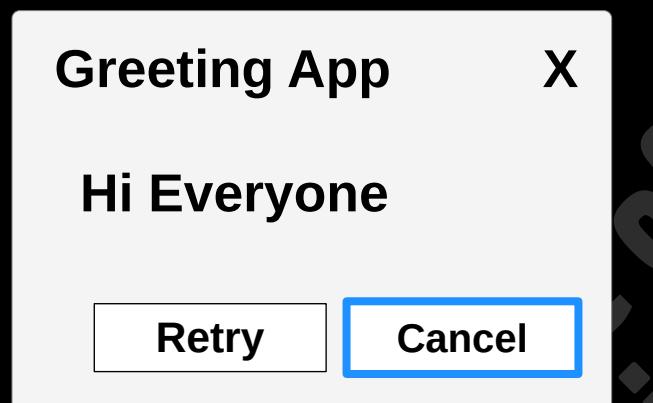
Yes

No

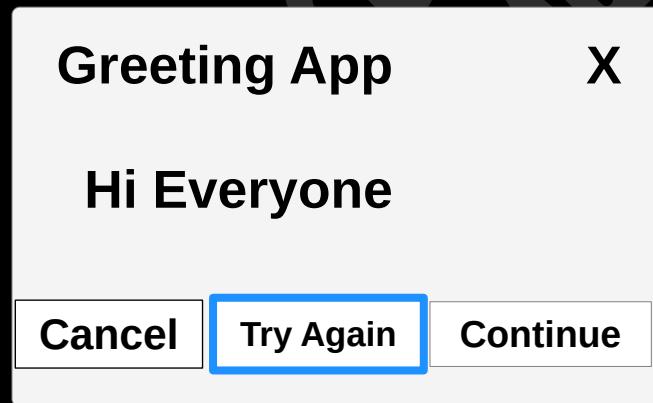
Cancel



**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 340)**



**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 341)**



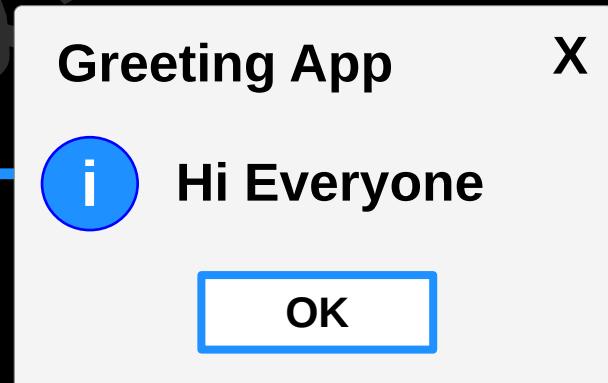
**ctypes.windll.user32.
MessageBoxW(0,
ourText, ourTitle, 342)**



MessageBox tkinter options

Simple Message Box

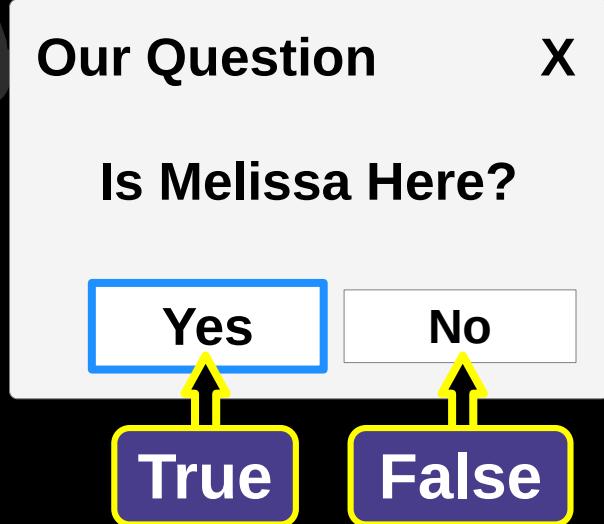
```
import tkinter  
from tkinter import messagebox  
  
ourWindow = tkinter.Tk()  
ourWindow.geometry("50x50")  
  
ourTitle = "Greeting App"  
ourMessage = "Hi Everyone"  
  
ourText = messagebox.showinfo(ourMessage,  
ourMessage)  
  
ourWindow.mainloop()
```



The messagebox appears, with an **info** icon and shows **ourTitle**, which is "Greeting App" and **ourMessage**, which is "Hi Everyone".

Open a Message Box, Yes No

```
import tkinter  
from tkinter import messagebox  
  
ourWindow = tkinter.Tk()  
ourWindow.geometry("50x50")  
  
ourTitle = "Our Question"  
ourMessage = "Is Melissa Here?"  
  
ourText = messagebox.askyesno(ourMessage,  
ourMessage)  
  
#print(ourText)  
  
if(ourText == True):  
    print("Pressed Yes")  
  
if(ourText == False):  
    print("Pressed No")  
  
ourWindow.mainloop()
```



askyesno returns True or False

Open a Message Box, OK, Cancel

```
import tkinter
from tkinter import messagebox
ourWindow = tkinter.Tk()
ourWindow.geometry("50x50")

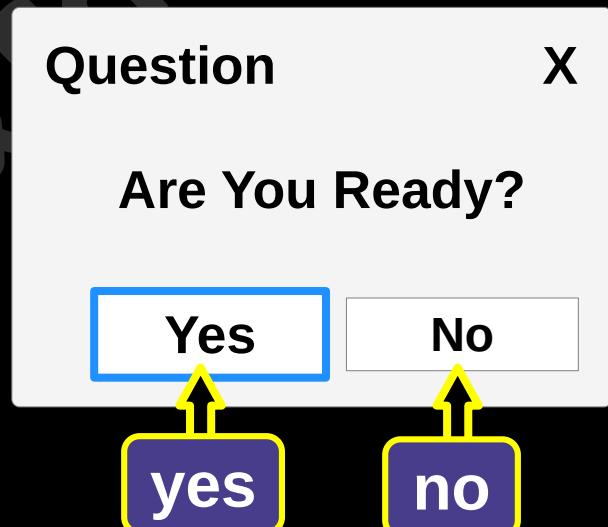
ourText = messagebox.askquestion("Question",
"Are You Ready?")

#print(ourText)

if(ourText == 'yes'):
    print("Pressed Yes")

if(ourText == 'no'):
    print("Pressed No")

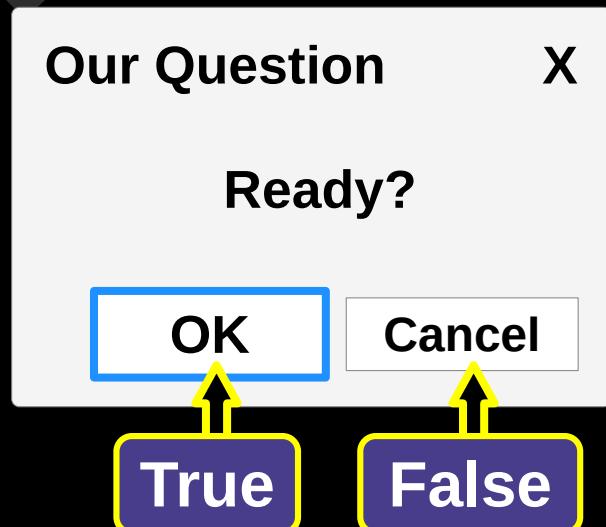
ourWindow.mainloop()
```



askquestion returns yes or no.

Open a Message Box, OK, Cancel

```
import tkinter  
from tkinter import messagebox  
  
ourWindow = tkinter.Tk()  
ourWindow.geometry("50x50")  
  
ourTitle = "Our Question"  
ourMessage = "Ready?"  
  
ourText = messagebox.askokcancel(ourMessage,  
ourMessage)  
  
#print(ourText)  
  
if(ourText == True):  
    print("Pressed OK")  
  
if(ourText == False):  
    print("Pressed Cancel")  
  
ourWindow.mainloop()
```



`askokcancel` returns **True** or **False**.

Open a Message Box, OK, Cancel

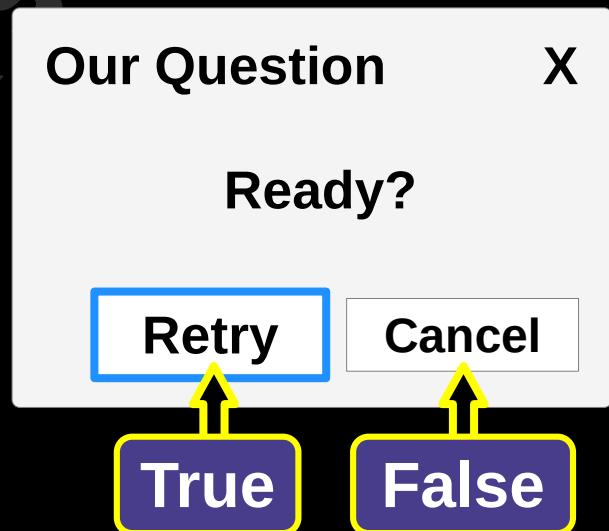
```
import tkinter
from tkinter import messagebox

ourWindow = tkinter.Tk()
ourWindow.geometry("50x50")

ourTitle = "Our Question"
ourMessage = "Ready?"

ourText = messagebox.askretrycancel(ourMessage,
ourMessage)

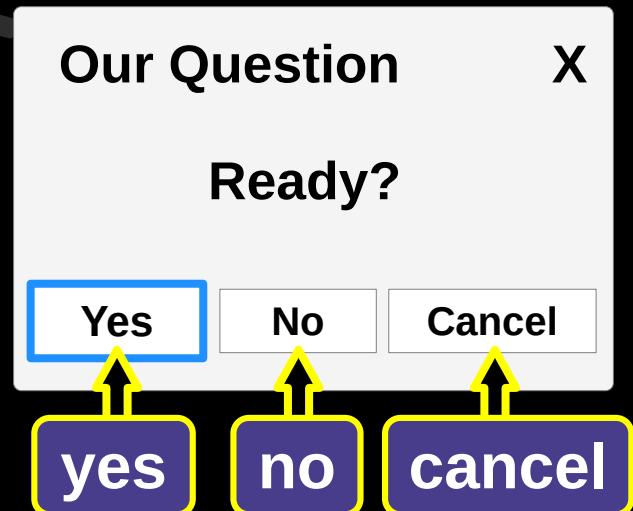
#print(ourText)
if(ourText == True):
    print("Pressed Retry")
if(ourText == False):
    print("Pressed Cancel")
ourWindow.mainloop()
```



`askokcancel` returns `True` or `False`.

Open a Message Box

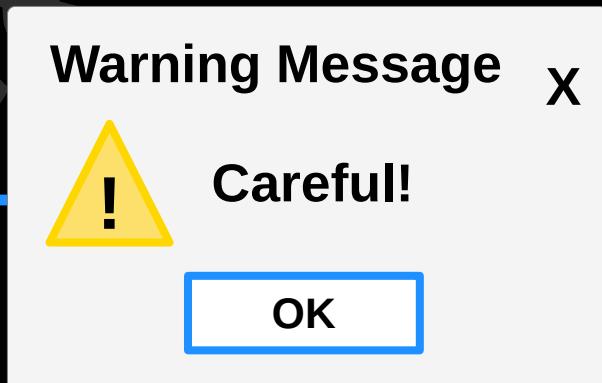
```
import tkinter  
from tkinter import messagebox  
  
ourWindow = tkinter.Tk()  
ourWindow.geometry("50x50")  
  
ourTitle = "Our Question"  
ourMessage = "Ready"  
  
ourText = messagebox.askyesnocancel(ourMessage,  
ourMessage)  
  
#print(ourText)  
  
if(ourText == True):  
    print("Pressed Yes")  
if(ourText == False):  
    print("Pressed No")  
if(ourText == None):  
    print("Pressed Cancel")  
  
ourWindow.mainloop()
```



askyesnocancel returns **True**, **False** or **None**.

Open a Message Box

```
import tkinter  
from tkinter import messagebox  
ourWindow = tkinter.Tk()  
ourWindow.geometry("50x50")  
  
ourTitle = "Warning Message"  
ourMessage = "Careful!"  
  
ourText = messagebox.showwarning(ourMessage,  
ourMessage)  
  
ourWindow.mainloop()
```



showwarning shows a warning symbol,
showinfo shows an info symbol, and
showerror shows an error message.

Resources:

<https://docs.python.org/3/>

<https://www.pygame.org/docs/>

https://docs.opencv.org/master/d6/d00/tutorial_py_root.html

<https://collegeofscripting.weebly.com/>

<https://collegeofscripting.wordpress.com>

Happy Programming!

DEDICATED TO GOD THE FATHER

**Presented by the
College of Scripting
Music & Science**

**We have taught over 2 million people
how to program computers, for free,
since March of 2007!**

**www.youtube.com/ScriptingCollege
www.github.com/ChristopherTopalian
www.CollegeOfScripting.weebly.com**

www.CollegeOfScripting.wordpress.com

**Christopher Topalian Copyright 2007-2021
All Rights Reserved. All content here is protected.**