

Python Coding Practice

Muhammad Izham

Universiti Malaysia Perlis

izham@unimap.edu.my

sugita5019@gmail.com

<https://github.com/izham-sugita>



Figure 1: Python's logo.

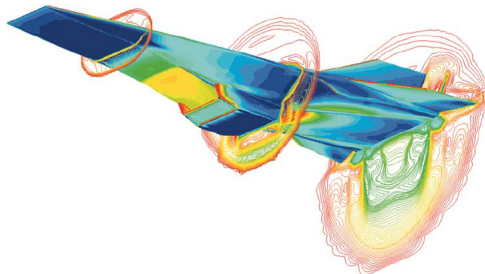


Figure 2: CFD result of HyperX at Mach 7 from NASA

A Little Background

- Created by: Guido van Rossum, 1989-1991
- Why: The creator wanted something easy to use.
- Is it really that easy? Yes (and no)
- Very readable with little memory management.
- <https://people.sc.fsu.edu/~jburkardt/>
- <https://docs.python.org/3/tutorial/index.html>
- <https://www.tutorialsteacher.com/python>
- <https://www.w3schools.com/python/default.asp>
- <https://www.tutorialspoint.com/python/> ← great place to start.

A Little More About Python

- Python2 → Python3; support for Python2 will end 2020.
- This workshop is exclusively on Python3 → refer to just Python
- Python is fully object oriented. Everything is considered object.
- Famous for AI and machine learning → Pytorch, Keras, TensorFlow
- Interpreter language but can be compiled → Cython, Numba
- Very well documented. Every module/libraries are documented online.
- Package management by package installer → pip, pip3
- pip → <https://pypi.org/project/pip/>
- **Python Package Index** → pypi, <https://pypi.org/>
- <https://github.com/> ← another place to look.

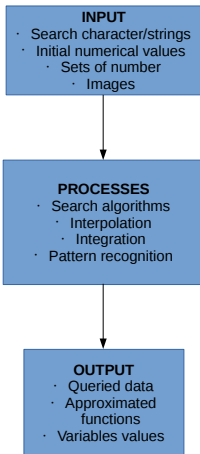
About Anaconda

- No, its not a different programming language.
- Anaconda is a *complete environment* for Python programming.
- Most major scientific package (NumPy, SciPy etc) are included.
- Package installer *conda*

```
user@pc-name:~/conda install any_package
```

- <https://www.anaconda.com/>

About programming



About programming

- Coding paradigm
 - programming language == English (sorry, Mandarin not required)
 - syntax is based on English
 - coding is a reduction of English instructions
- Syntax must be remembered
 - read the manual → documentations are vital
 - memorize THE MOST COMMONLY USED syntax only
 - good algorithm will always beats bad algorithm
- I don't remember every syntax so you have to bare with me



Installation

- Download from <https://www.python.org/>
- For Ubuntu download from repository:

```
user@pc-name:~/apt$ install python3
```

- For Windows, download from <https://www.python.org/downloads/windows/>
- For Ubuntu installing packages:

```
user@pc-name:~/pip3$ install numpy
```

- For Windows installing packages if `C : \Python\Scripts\` is in the path:

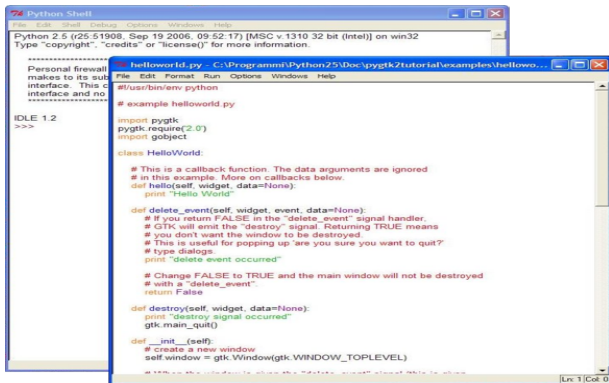
```
C:\User\dummy\> pip3 install numpy
```

- For Windows installing packages:

```
C:\User\dummy\> python3 -m pip3 install numpy
```

Introduction

- For Windows, the default editor is IDLE



The screenshot shows the IDLE Python Shell and a Python script editor. The Python Shell window displays the Python version (2.5) and a message about the personal firewall. The script editor shows a file named 'helloworld.py' with the following code:

```
#!/usr/bin/env python

# example helloworld.py

import pygtk
pygtk.require(2.0)
import gobject

class HelloWorld:

    # This is a callback function. The data arguments are ignored
    # in this example. More on callbacks below.
    def hello(self, widget, data=None):
        print "Hello World"

    def delete_event(self, widget, event, data=None):
        # If you return FALSE in the "delete_event" signal handler,
        # GTK will emit the "destroy" signal. Returning TRUE means
        # you don't want the window to be destroyed.
        # This is useful for popping up 'are you sure you want to quit?'
        # type dialogs.
        print "delete event occurred"

        # Change FALSE to TRUE and the main window will not be destroyed
        # with a "delete_event".
        return False

    def destroy(self, widget, data=None):
        print "destroy signal occurred"
        gtk.main_quit()

    def __init__(self):
        # Create a new window
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)

        # Add a "delete_event" signal handler to the window. This means that
```

Hello, Python!

- Symbols to remember.

```
1 #everything about the basics
2
3 '''
4 This is how you put a long comment.
5 : <- colon
6 ; <- semi-colon
7 () <-parentheses
8 [] <-brackets
9 {} <- braces
10 # <- hash sign
11 ' <- apostrophe
12 " <- quote mark
13 .....
14 Thats about everything you need to know/remember
15 '''
```

Hello, Python!

- The print() function.

```
1 print(" Hello , world!" )
2 print(" Hello , Python 3\n" ) # "\n"; next line
3 money=4.0
4 print(" I have %f B, in a yacht somewhere"%money)
5 var0 = 1
6 var1 = 1.07
7 print("To print float number %f and integer %d"%(
    var1 , var0))
8 str1 = "Another way "
9 str2 = "to add something."
10 print(str1+str2)
11 print(str1+"\t"+str2) # "\t" for tab spacing
```

file: py-hello.py

Basic I/O including files.

- Basic I/O from terminal.

```
1 #Basic terminal input
2 var0 = input("Enter something:")
3 print(var0) #input by default is STRING ONLY!
4
5 var1 = input("Enter integer: ")
6 var1 = int(var1)
7 print(var1, var1+var1)
8 print(var1, var1, var1, var1) #you can print a lot
9
10 var2 = input("Enter real number: ")
11 var2 = float(var2)
```

file: py-basic-io.py

Basic I/O including files.

- Write to file and close.

```
1 fo = open("test.txt", "w+")
2 fo.write("Test file\n")
3 fo.close()
```

- Write to file with different access mode.

```
1 fo = open("test.txt", "a+")
2 anim_list = ["cat", "tiger", "bear", "elephant", "mouse",
3             ", "dear"]
4
5 for animal in anim_list:
6     print(animal)
7     fo.write(animal+"\n")
8
9 fo.close()
```

file: py-file-io.py

Loop in Python

- for-loop

```
1 #Example of simple loop
2 import numpy as np
3 imax = 64
4 a = np.ndarray(shape=(imax), dtype=float)
5 for i in range(imax):
6     a[i] = 1.0
```

Loop in Python

- Loop with string data.

```
1 words = ['I', 'me', 'you', 'him']  
2 for w in words:  
3     print(w, len(w))
```


Loop in Python

- while-loop.

```
1 i=1
2 while i < 6:
3     print(i)
4     i += 1
5
6 i=1
7 while i < 6:
8     print(i)
9     if i == 4:
10        break
11    i += 1
```

file: py-while.py

More precise loop in Python

- for-loop for forward-backward sweep.

```
1 #print precise
2 for i in range(6,1,-1):
3     print("a[%d]=%.4f"%(i , a [ i ]))
4
5 print()
6 for i in range(1,6):
7     print("a[%d]=%.4f"%(i , a [ i ]))
```

file: py-precise-loop.py

Standard template

- This is a standard template slide.
- Modify by adding items.

Finite difference method for heat equation

- The 1D equation : $\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2}$

```
1 import numpy as np
2 from numba import jit , njit
3 import time
4
5 @jit(nopython=True, nogil=True, cache=True)
6 def theloop(itermax ,imax ,icenter ,coeff ,dt ,fx ,fxnew ,
7             peak ,tminus):
8     for iter in range(itermax):
9         for i in range(1,imax):
10             fxnew[i] = fx[i] + coeff*( fx[i-1] -
11                                         2.0*fx[i] + fx[i+1] )
12
13             peak.append(fxnew[icenter])
14             tminus.append(( iter+1)*dt)
15             fx = fxnew
16
17 pi = np.pi
```

Thank You!
Questions?