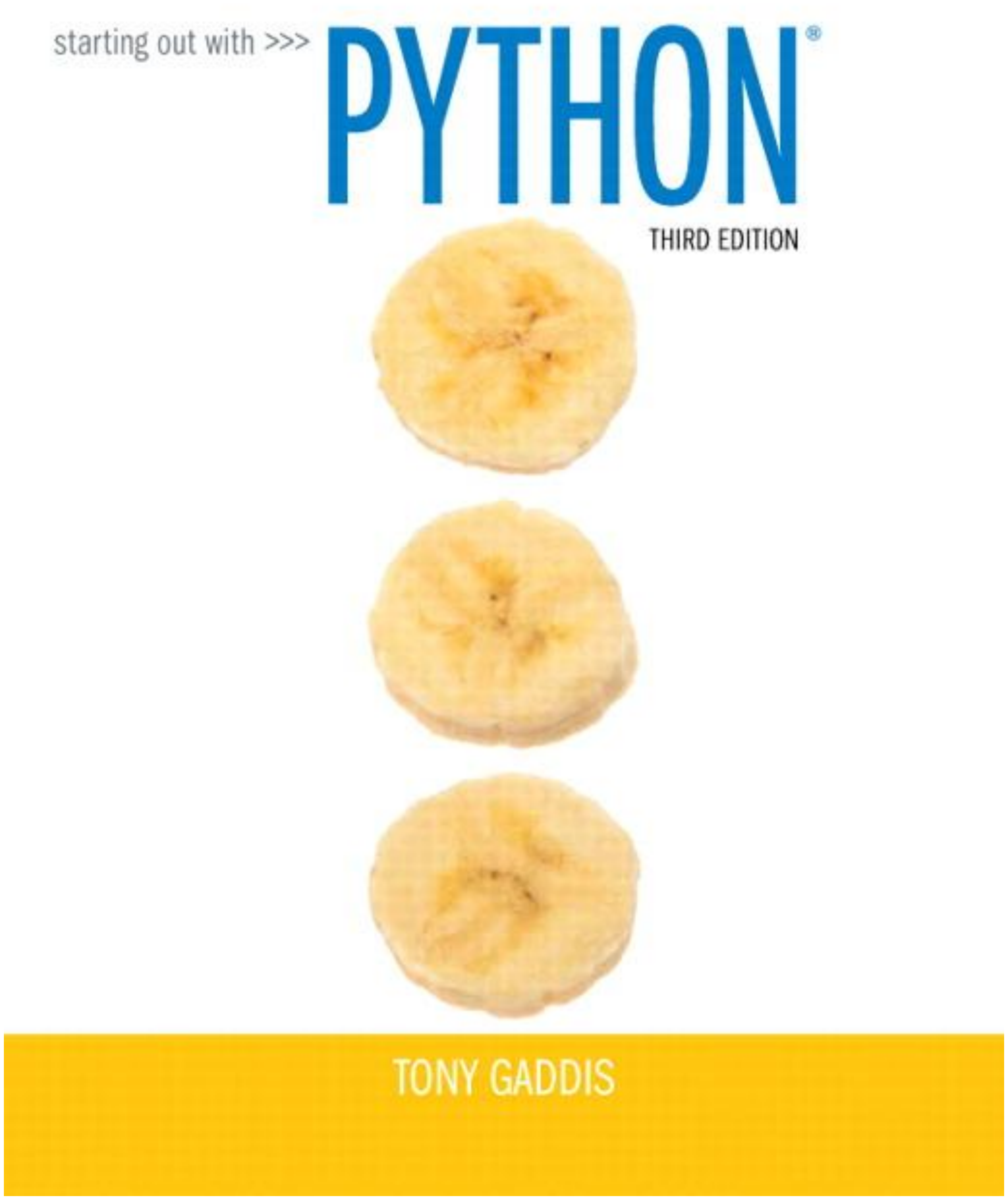
APPENDIX B

Introduction to Python and IDLE



Introduction

- In this class, we will be using Python.

- Furthermore, we'll be using IDLE, Python's own IDE (Integrated Development Environment) — combined source code editor and Python interpreter GUI.

- It will save a lots of time if everyone has IDLE installed and is familiar how to open, edit, and run a script (which is just a text file) using Python.

- However, Python is not IDLE. Python is a programming language and IDLE is used to execute statements written in Python.

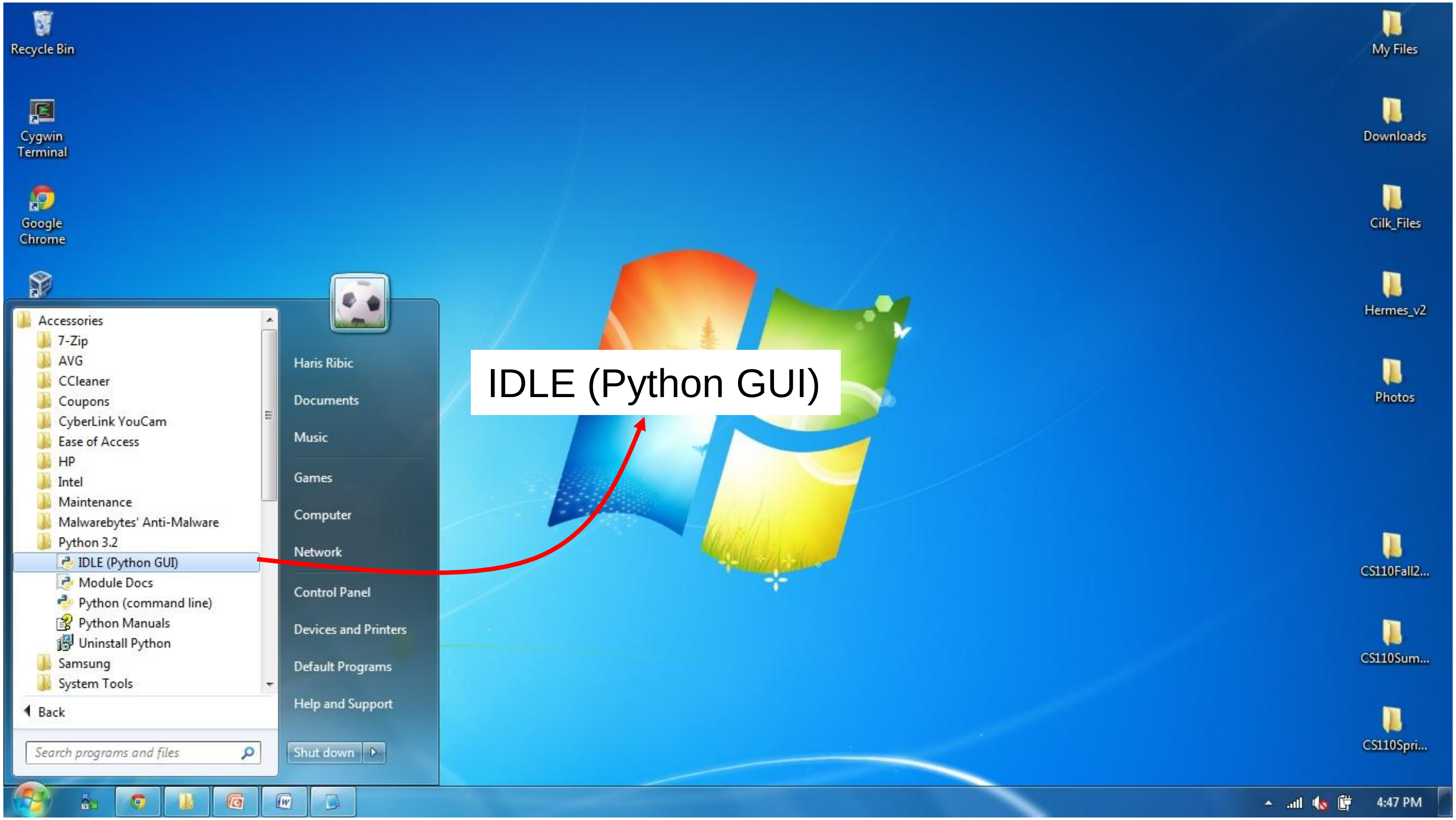
Introduction

- We will be using 3.3.x or 3.4.x versions
  - Do not download any version below 3.3.x
  - https://www.python.org/

The IDLE Programming Environment

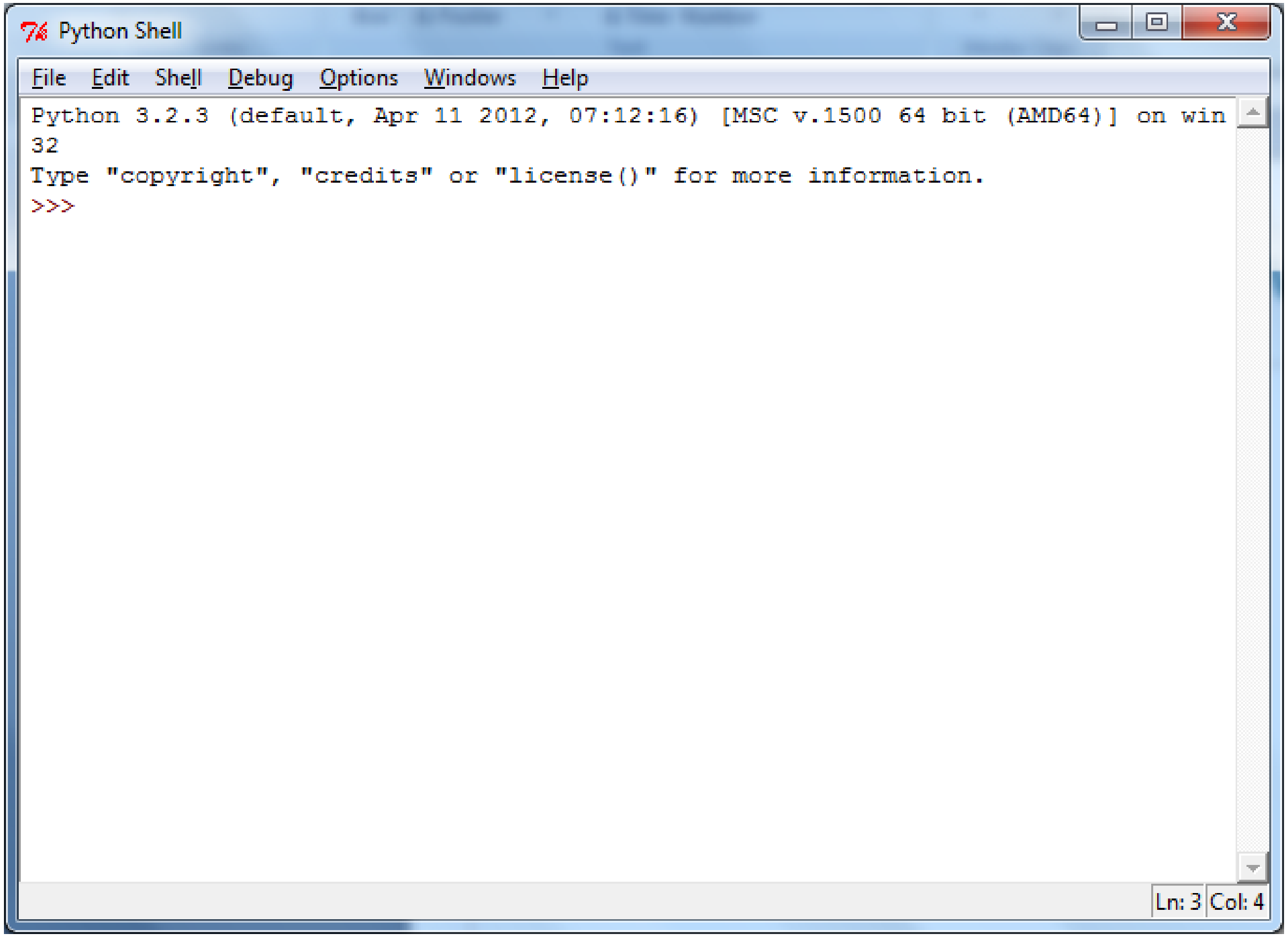
- **IDLE: single program that provides tools to write, execute and test a program**
  - Automatically installed when Python language is installed
  - Runs in interactive mode
  - Has built-in text editor with features designed to help write Python programs

Start IDLE



IDLE (Python GUI)

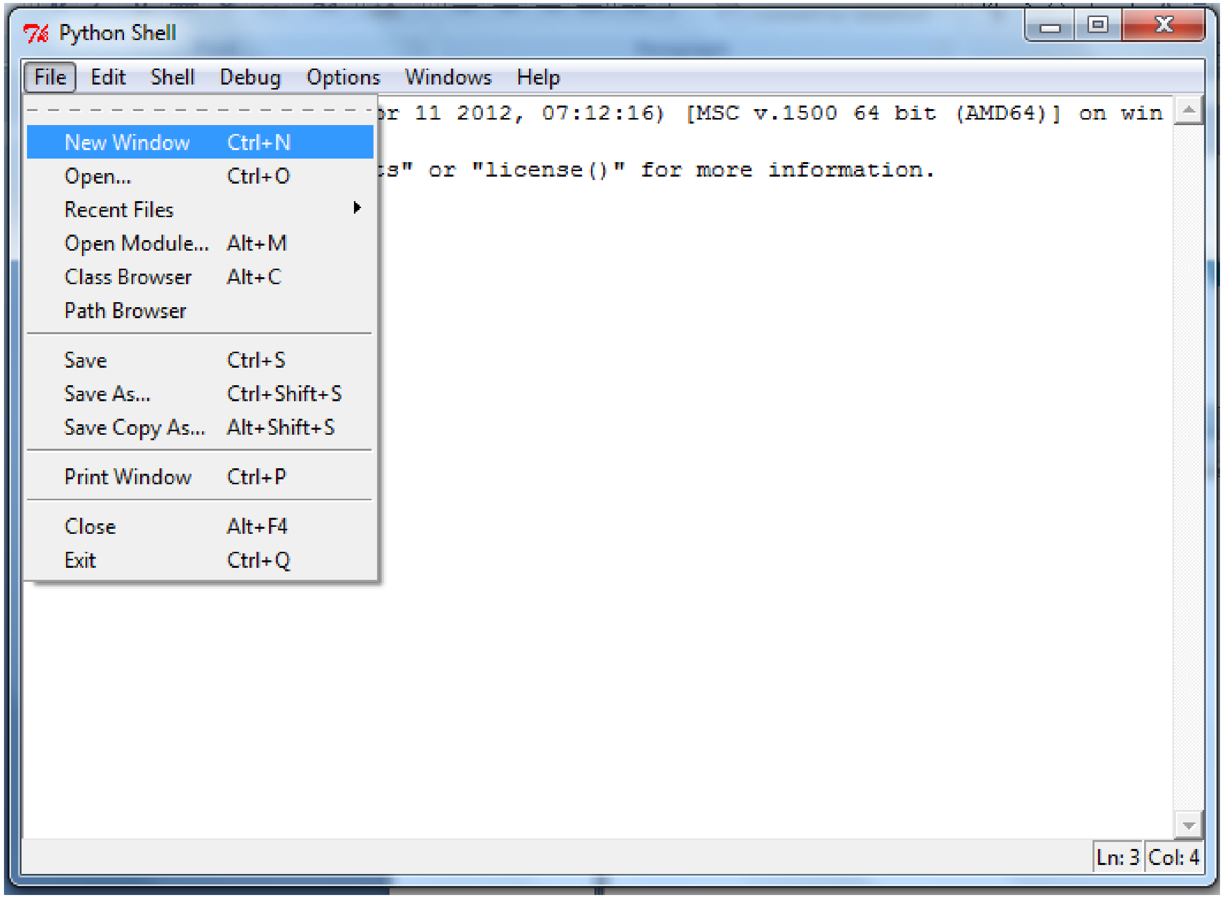
IDLE Shell Window (Interactive mode)



```
Python 3.2.3 (default, Apr 11 2012, 07:12:16) [MSC v.1500 64 bit (AMD64)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>>
```

Writing a Python Program in IDLE Script Window

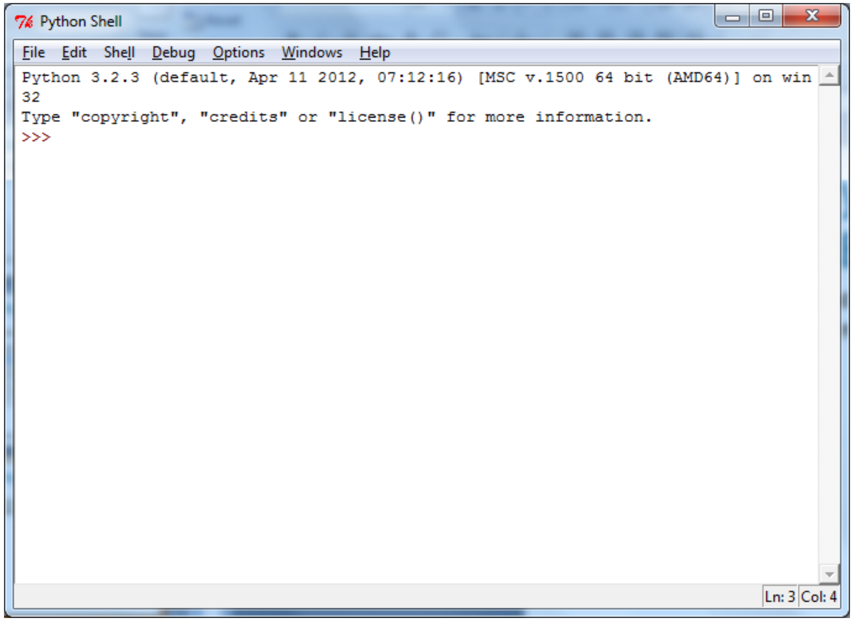
- From the Python Shell window, select New Window from the File menu



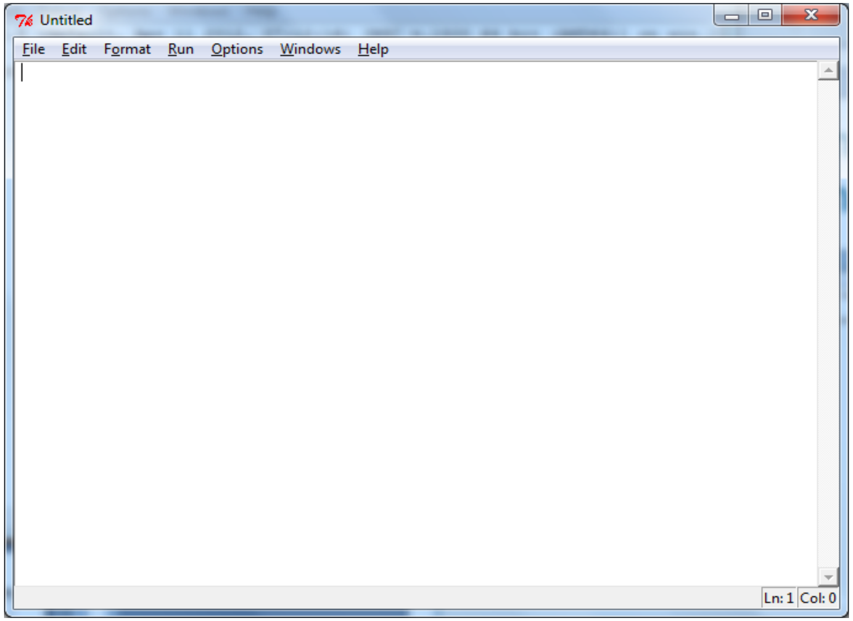
```
or 11 2012, 07:12:16) [MSC v.1500 64 bit (AMD64)] on win
s" or "license()" for more information.
```

Writing a Python Program in IDLE Script Window

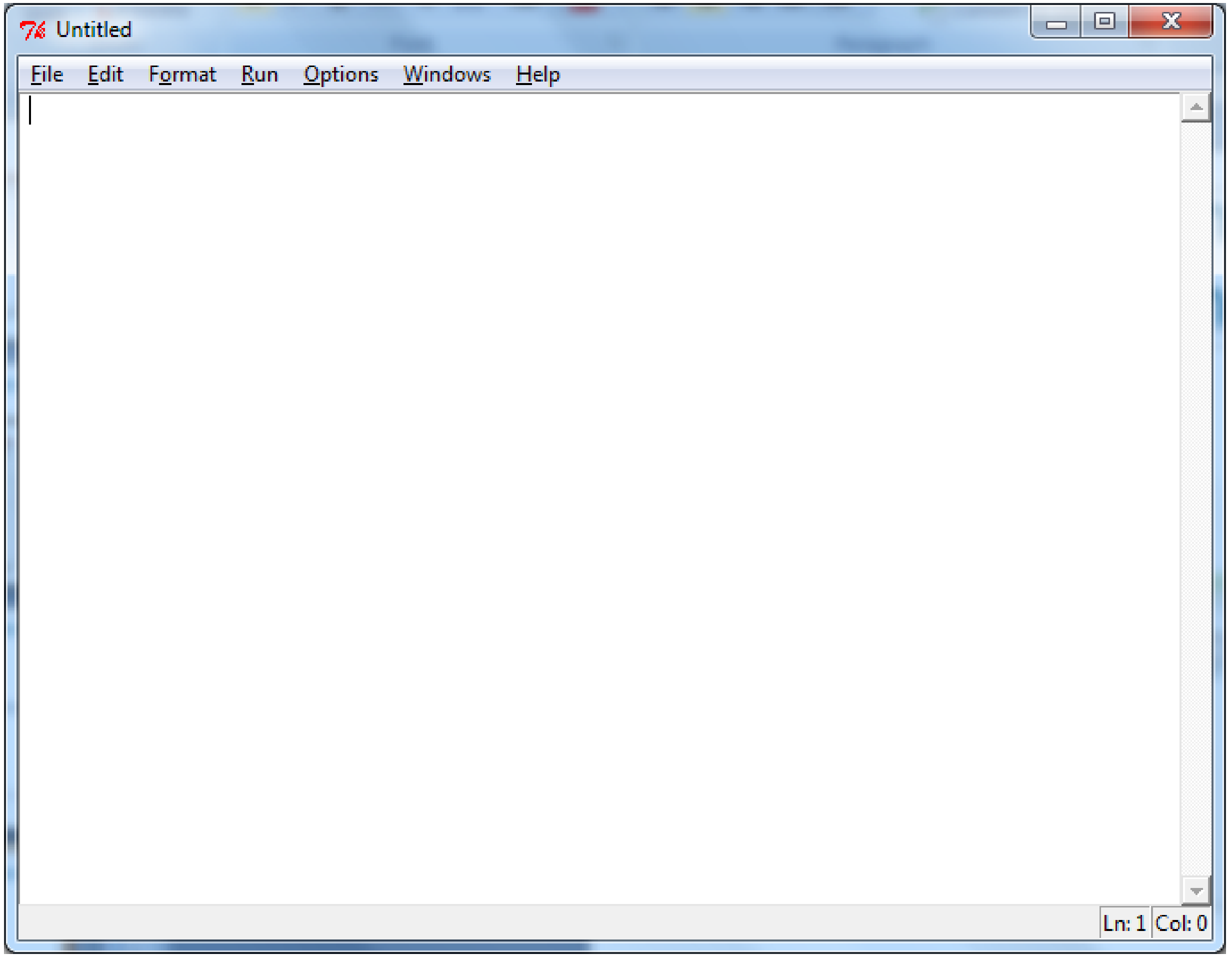
- You will see a window entitled "Untitled"
- Edit window is 80 characters wide
- Do not resize width. You may resize height.



```
Python 3.2.3 (default, Apr 11 2012, 07:12:16) [MSC v.1500 64 bit (AMD64)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>>
```



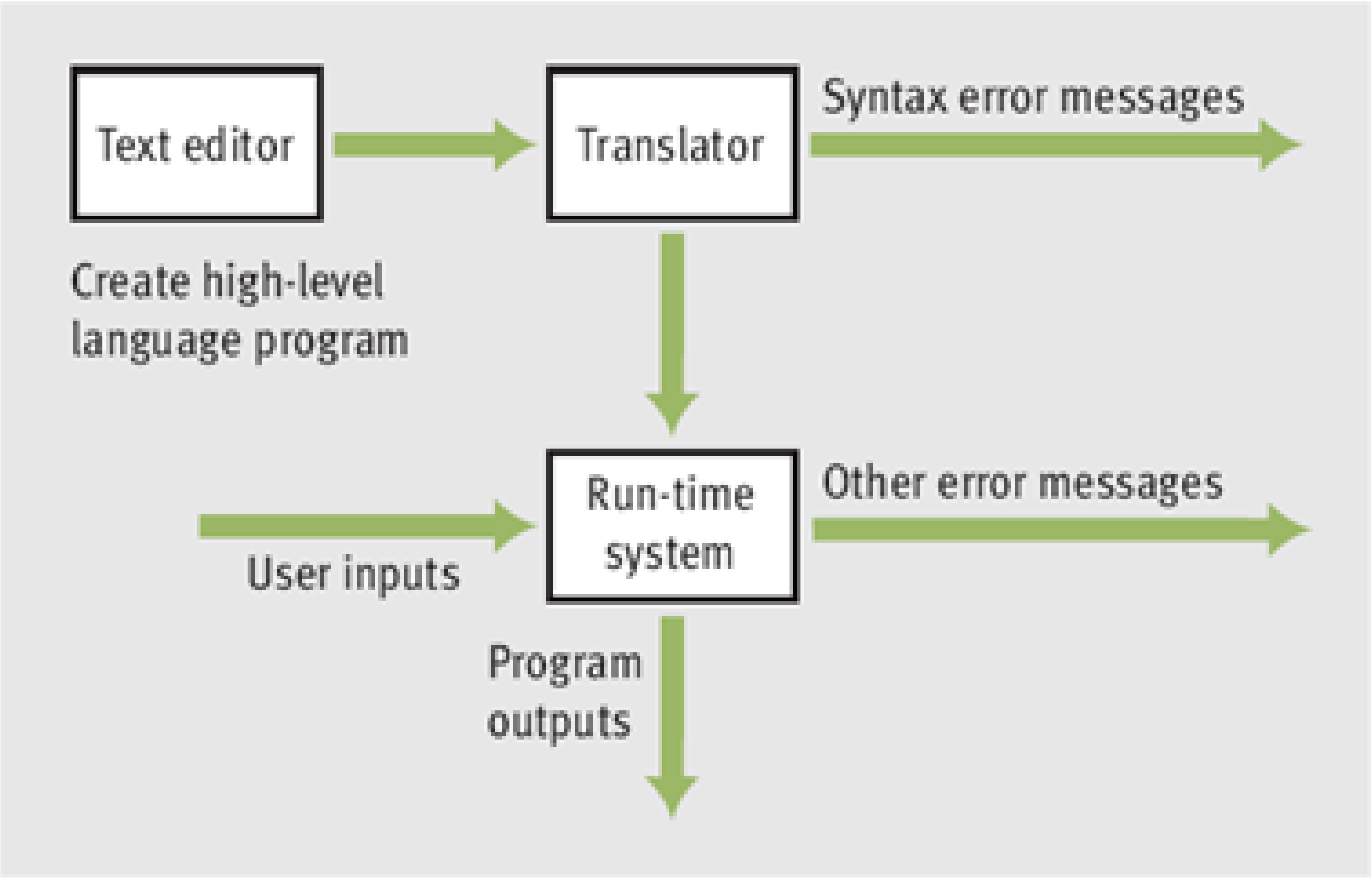
Writing a Python Program in IDLE Script Window



Python

- High-Level language: allows simple creation of powerful and complex programs
  - No need to know how CPU works or write large number of instructions
  - More intuitive to understand
- Interpreter: translates and executes instructions in high-level language program
  - Python is an interpreted language
  - Translates one instruction at a time
  - No separate machine language program
- Source code: statements written by programmer
  - Syntax error: prevents code from being translated

Python



Text editor

Create high-level language program

Translator

Syntax error messages

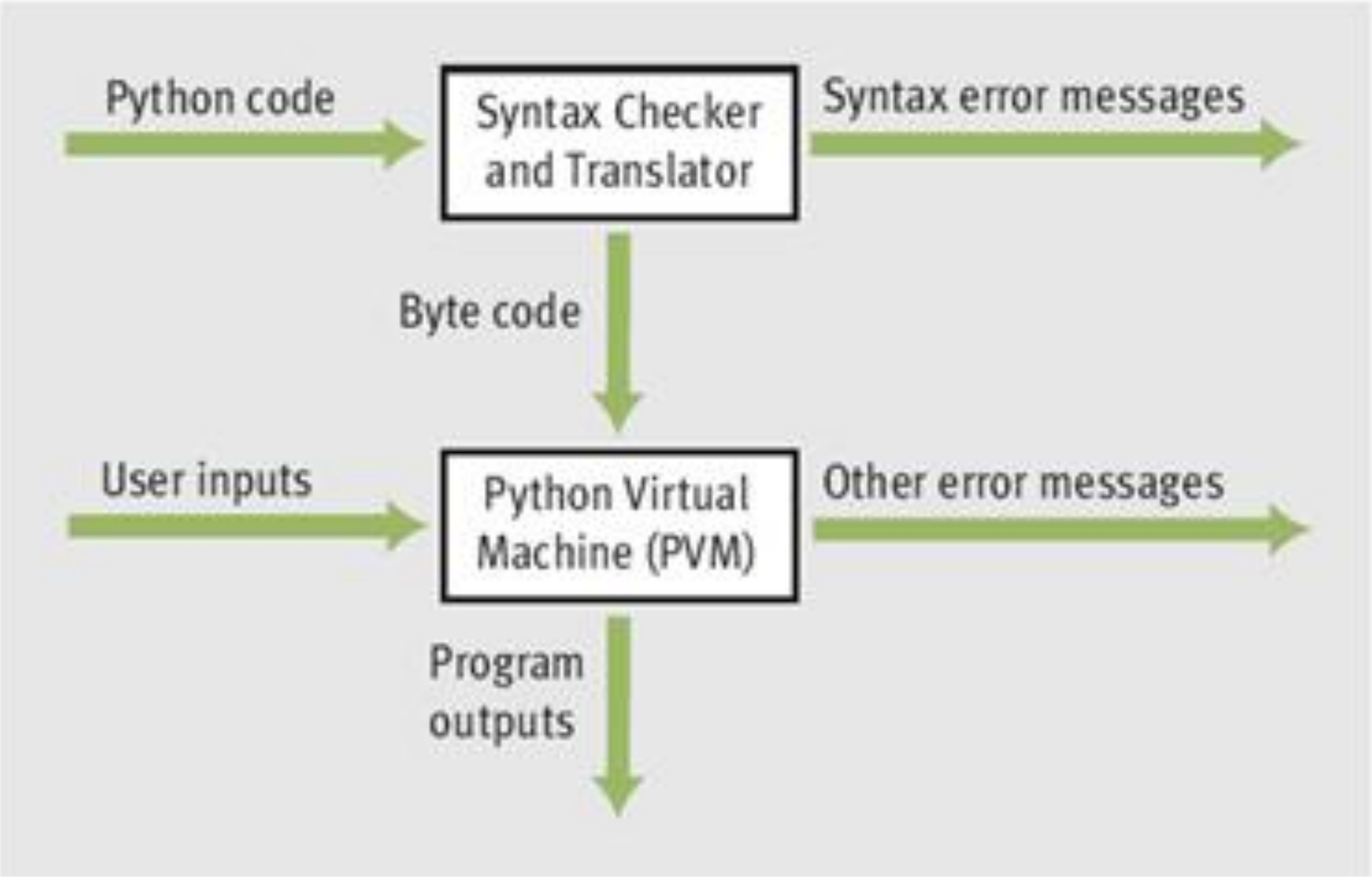
Run-time system

User inputs

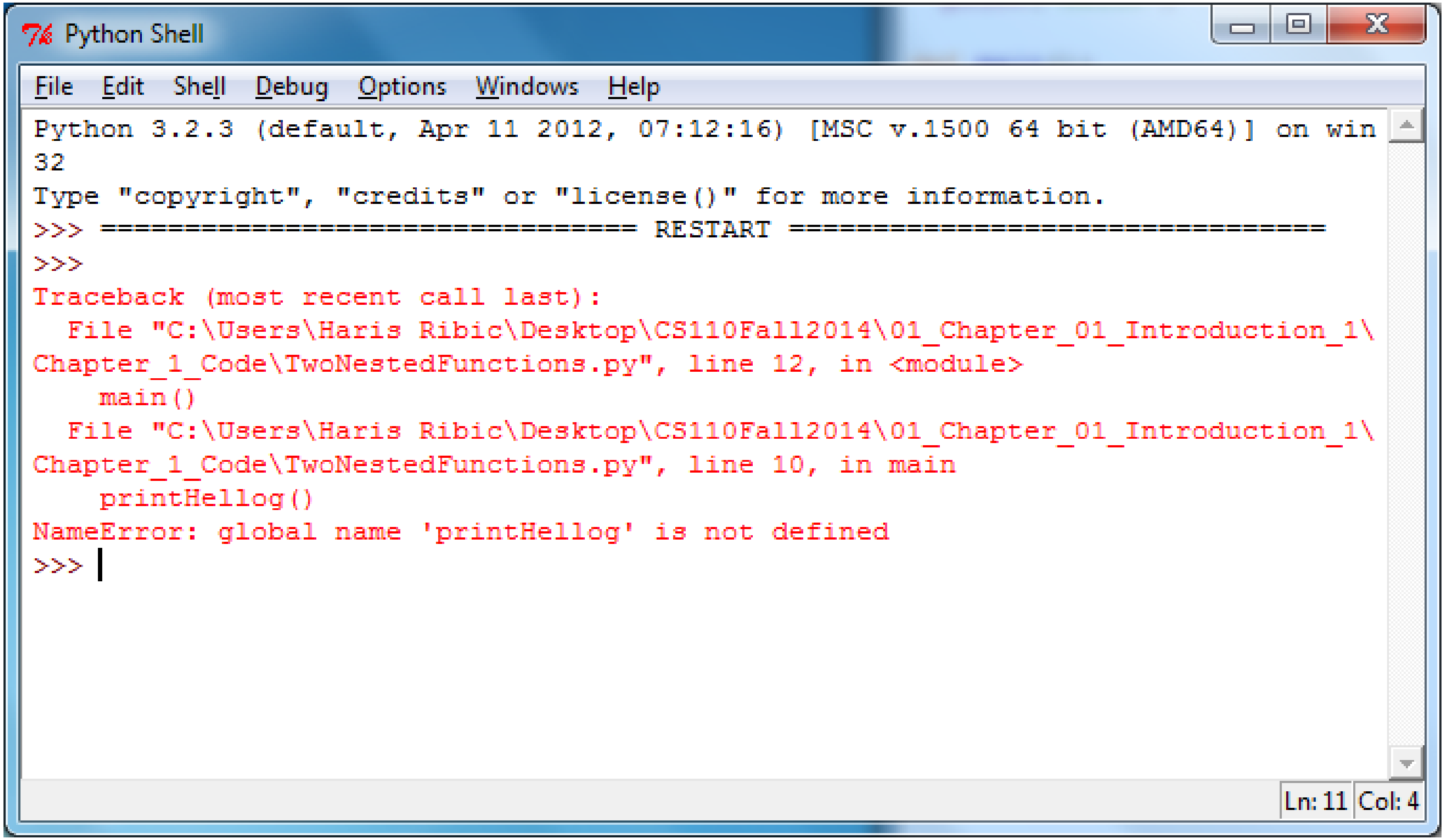
Other error messages

Program outputs

Python


Python code
Syntax Checker and Translator
Syntax error messages
Byte code
User inputs
Python Virtual Machine (PVM)
Other error messages
Program outputs

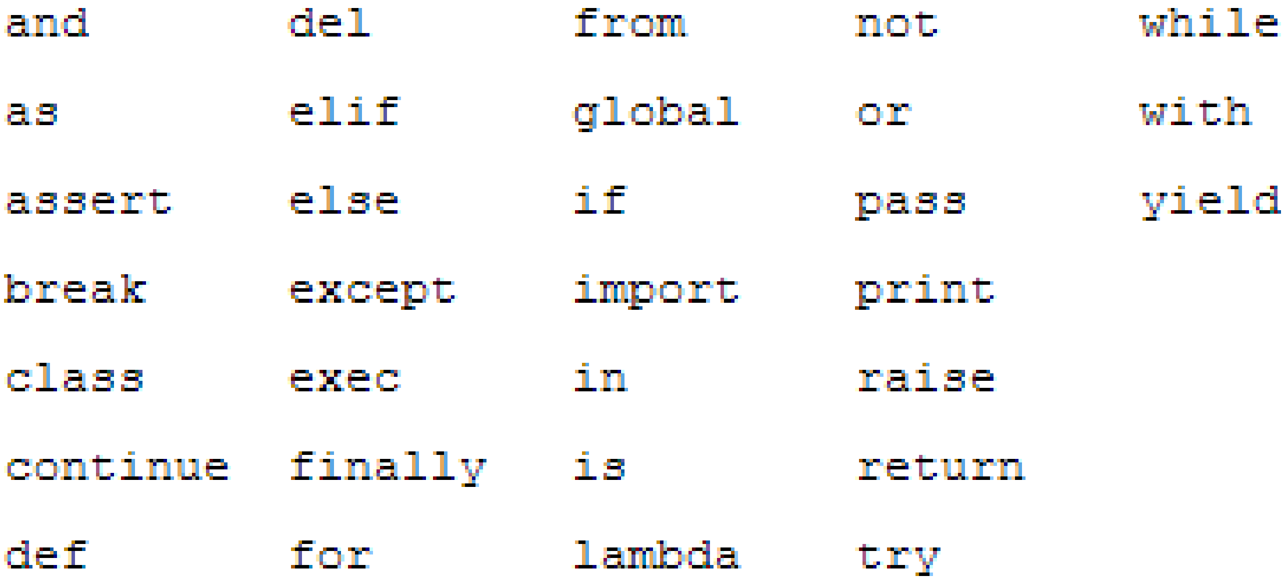
Python Error Message



```
Python 3.2.3 (default, Apr 11 2012, 07:12:16) [MSC v.1500 64 bit (AMD64)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>> 
Traceback (most recent call last):
  File "C:\Users\Haris Ribic\Desktop\CS110Fall2014\01_Chapter_01_Introduction_1\
Chapter_1_Code\TwoNestedFunctions.py", line 12, in <module>
    main()
  File "C:\Users\Haris Ribic\Desktop\CS110Fall2014\01_Chapter_01_Introduction_1\
Chapter_1_Code\TwoNestedFunctions.py", line 10, in main
    printHellog()
NameError: global name 'printHellog' is not defined
>>> 
```

Python Keywords

- Python is a dynamic language. It changes during time. The list of keywords may change in the future.



and
as
assert
break
class
continue
def
del
elif
else
except
exec
finally
for
from
global
if
import
in
is
lambda
not
or
pass
print
raise
return
try
while
with
yield

Writing Python Programs and Running Them in Script Mode

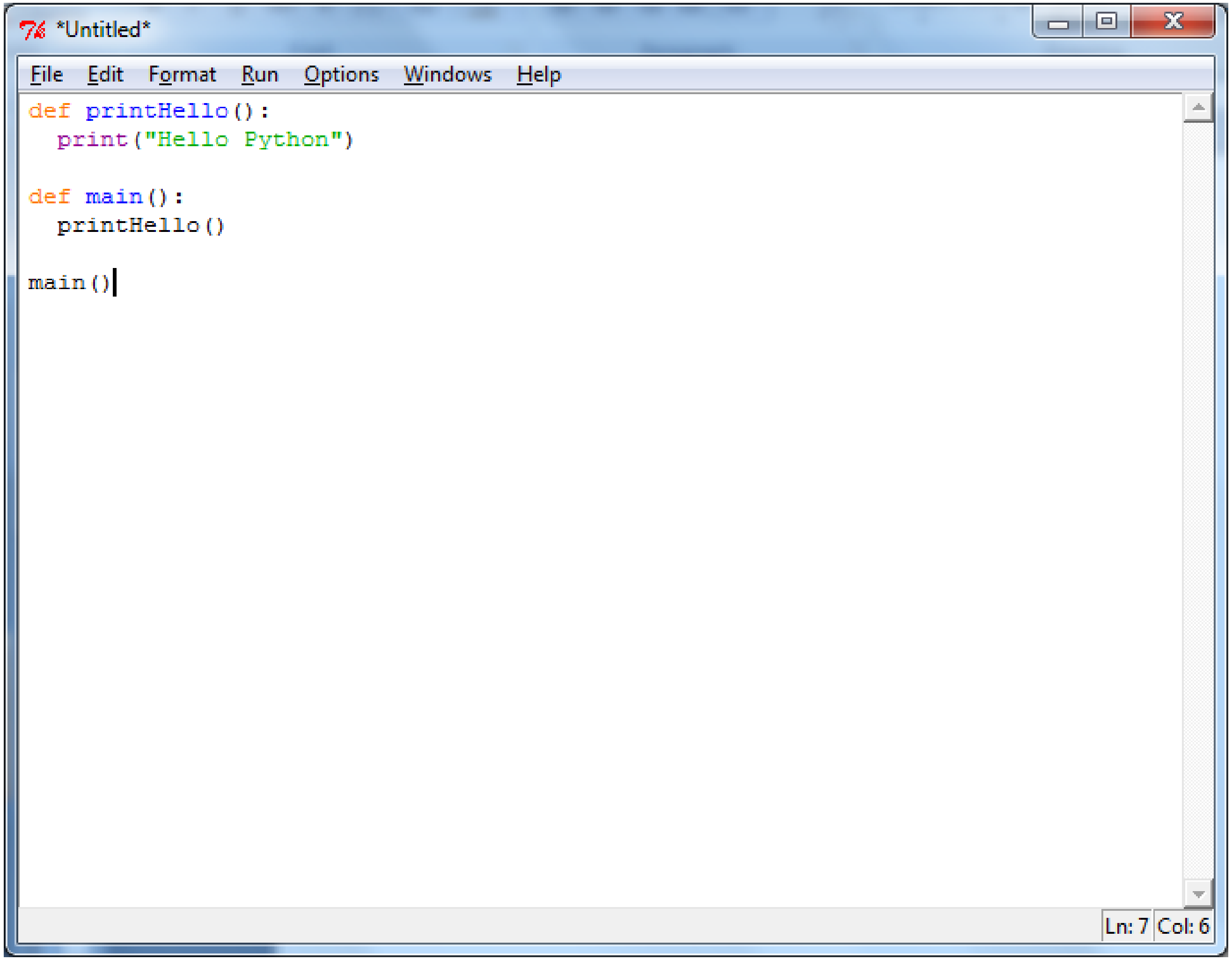
- **Statements entered in interactive mode are not saved as a program**
- **To have a program use script mode**
  - Save a set of Python statements in a file
  - The filename should have the `.py` extension

    `filename.py`

Detecting and Correcting Syntax Errors

- Programmers inevitably make typographical errors when editing programs, called syntax errors
    - The Python interpreter will usually detect these
- Syntax: rules for forming sentences in a language
- When Python encounters a syntax error in a program, it halts execution with an error message

Adding Python Code



```
def printHello():
  print("Hello Python")

def main():
  printHello()

main()
```

Hello Python

```
def printHello():
    print("Hello Python")
```

printHello() function:
Function to print. Called from main() to print.

```
def main():
    printHello()
```

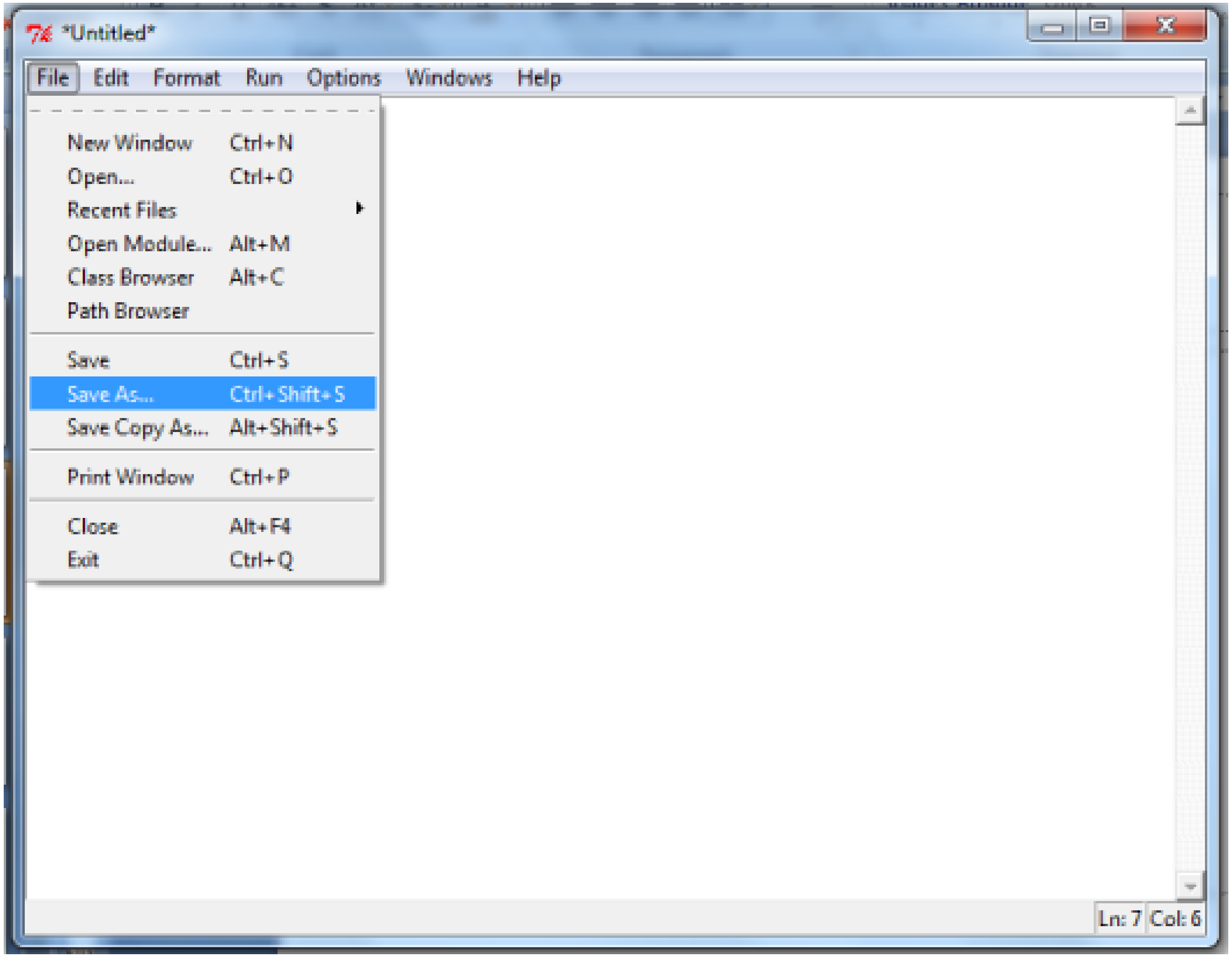
main() function:
Main entry into the program. Transfers control to other functions.

```
main()
```

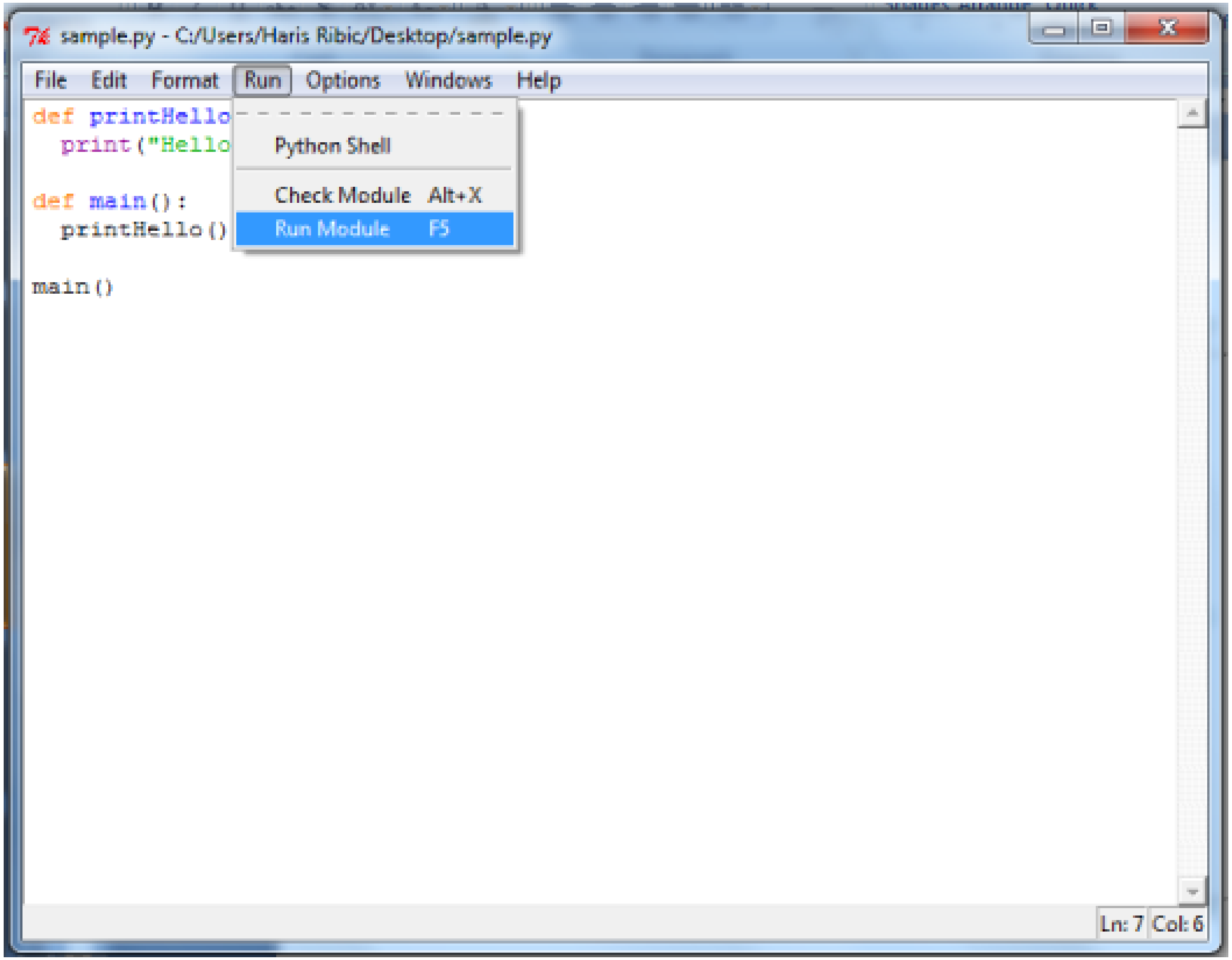
Begin of execution.

Must be the last function of any program.
There is only 1 (one) main() per program.

Save Code

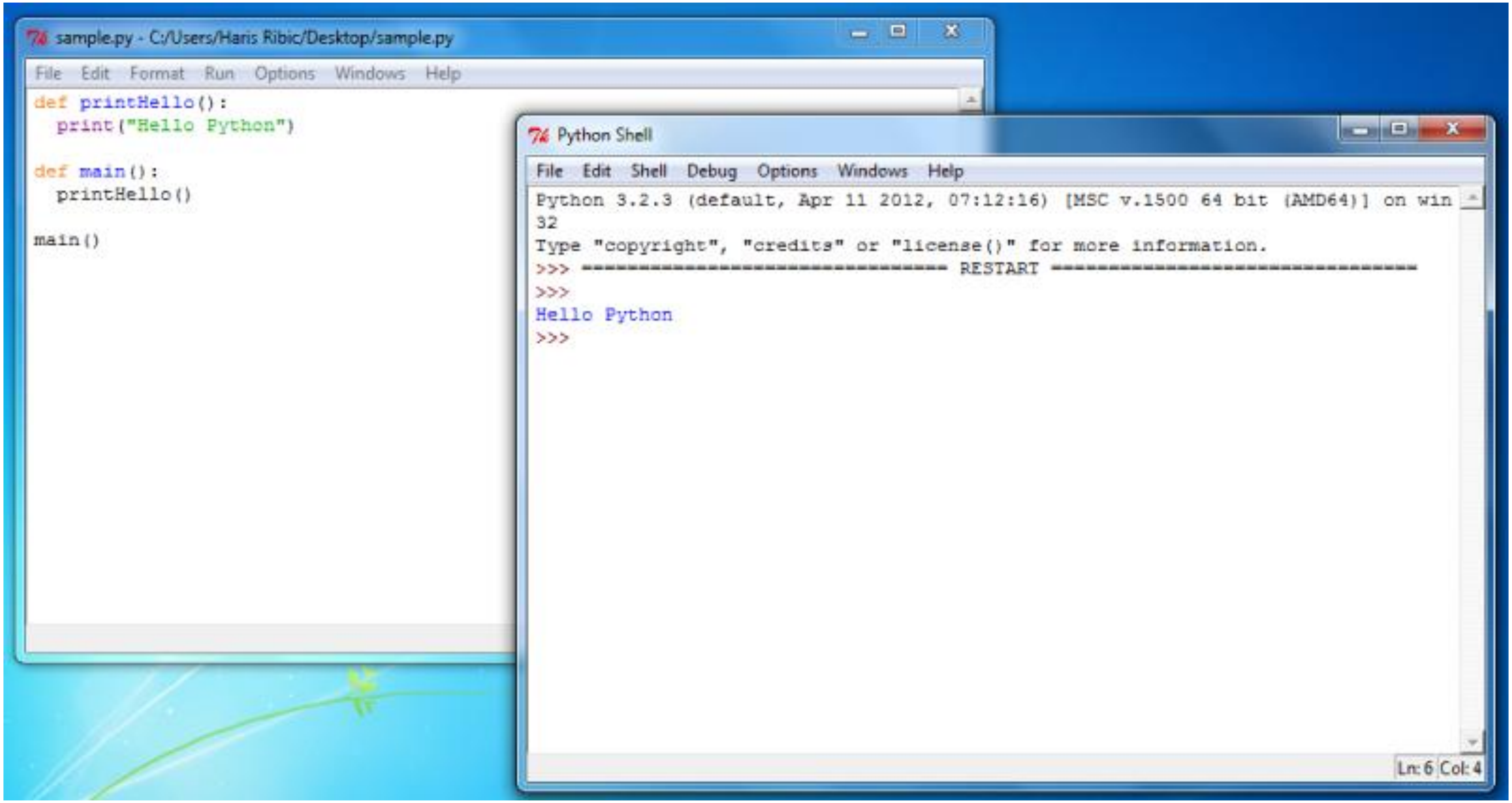

Untitled
File
Edit
Format
Run
Options
Windows
Help
New Window Ctrl+N
Open... Ctrl+O
Recent Files
Open Module... Alt+M
Class Browser Alt+C
Path Browser
Save Ctrl+S
Save As... Ctrl+Shift+S
Save Copy As... Alt+Shift+S
Print Window Ctrl+P
Close Alt+F4
Exit Ctrl+Q
Ln: 7
Col: 6

Run Code



```
def printHello
  print("Hello

def main():
  printHello()

main()
```

Run Code



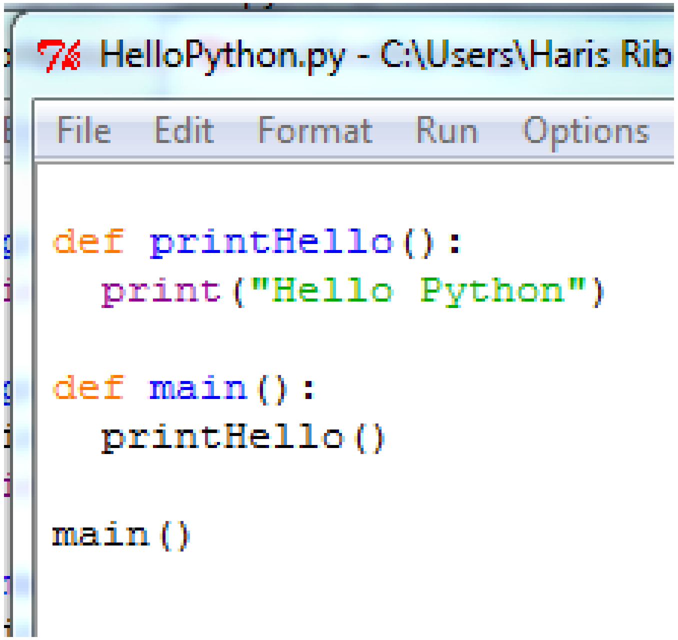
sample.py - C:/Users/Haris Ribic/Desktop/sample.py
File Edit Format Run Options Windows Help

```
def printHello():
  print("Hello Python")

def main():
  printHello()

main()
```

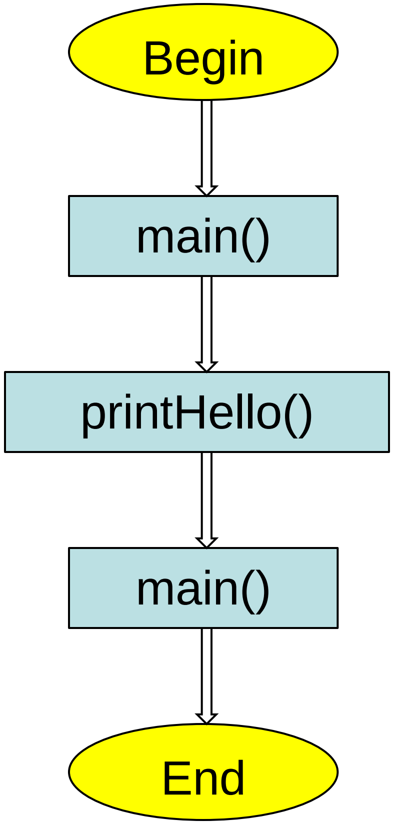
Python Shell
File Edit Shell Debug Options Windows Help

```
Python 3.2.3 (default, Apr 11 2012, 07:12:16) [MSC v.1500 64 bit (AMD64)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Hello Python
>>>
```

Execution Order



```
def printHello():
    print("Hello Python")

def main():
    printHello()

main()
```


Begin

main()

printHello()

main()

End

Once inside main(),
we look for what to do

… now we found printHello() so let’s go do that

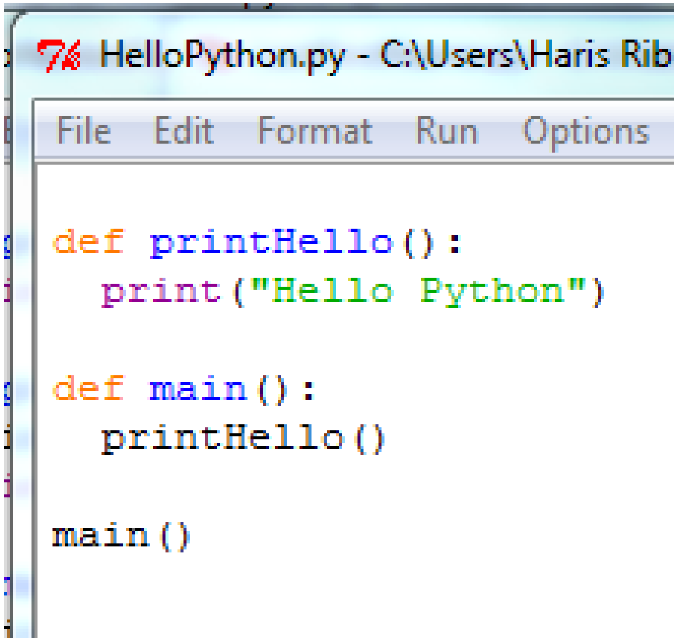
main() transfers control to printHello() and we do what it is in that function

printHello() has finished ad gives back control to main()

… main() has nothing else to do so the program ends

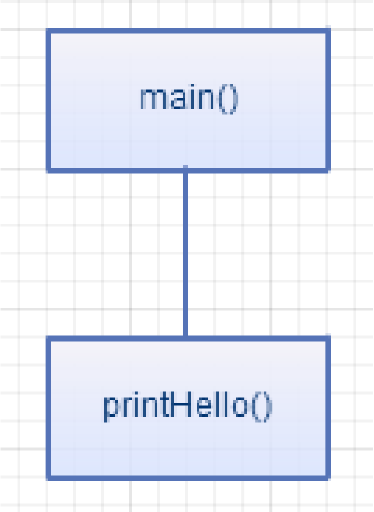
Execution order only shows function() calls

Hierarchy Chart



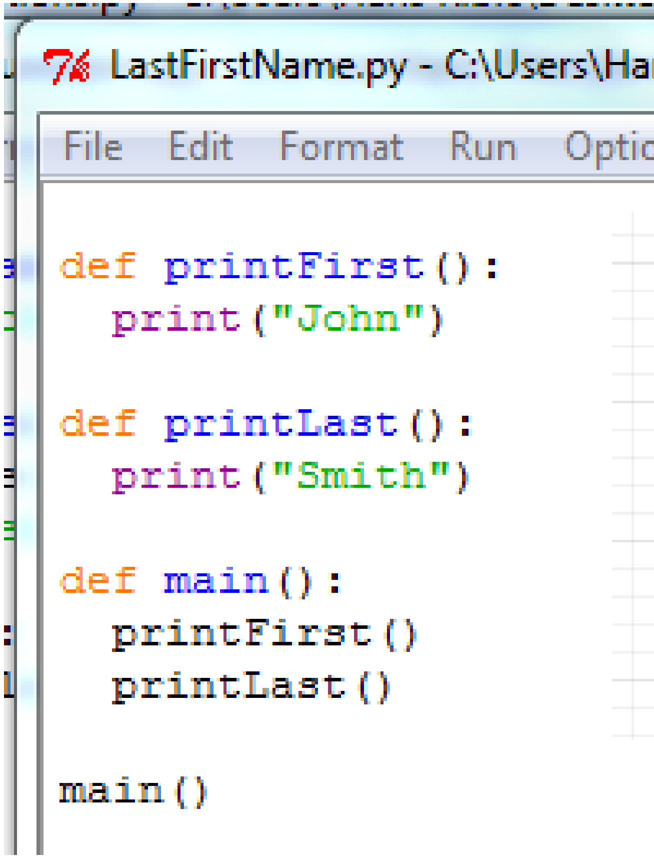
HelloPython.py - C:\Users\Haris Rib
File Edit Format Run Options

```
def printHello():
  print("Hello Python")

def main():
  printHello()

main()
```

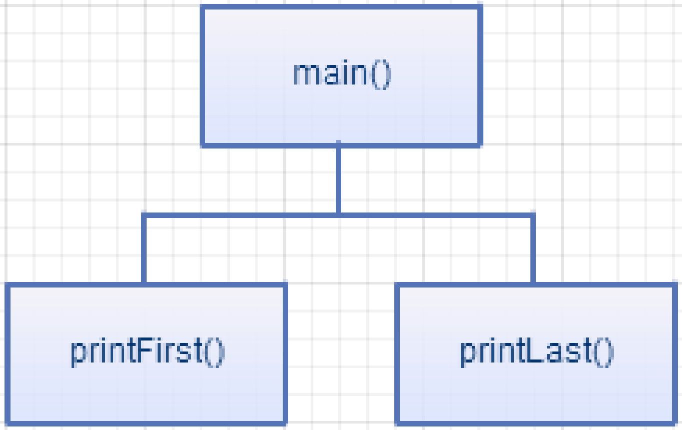


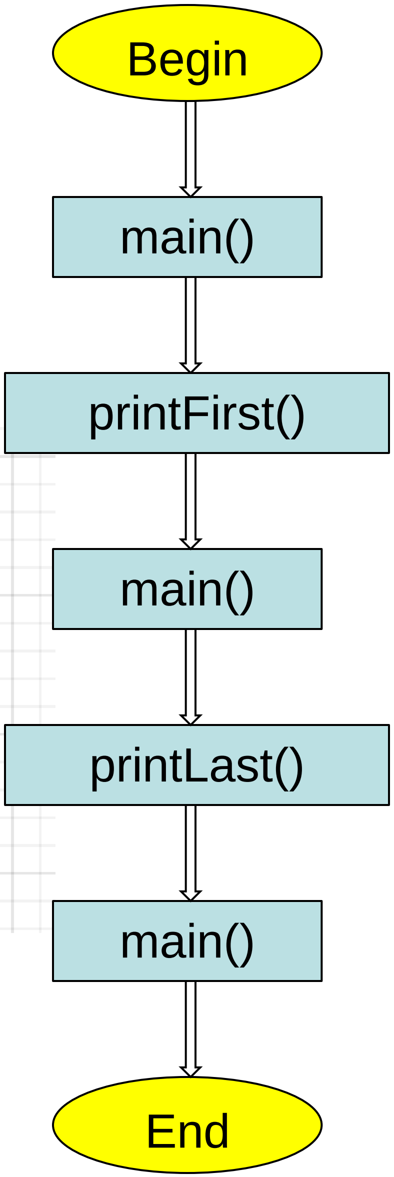
Shows nested function() calls.

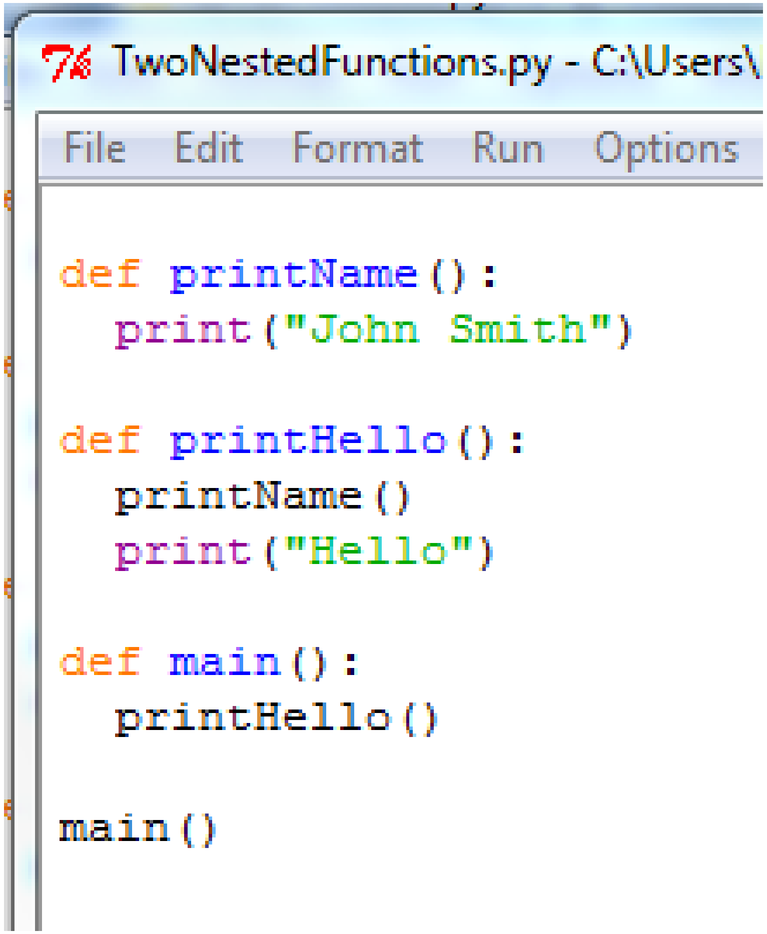
Shows relationships among functions()



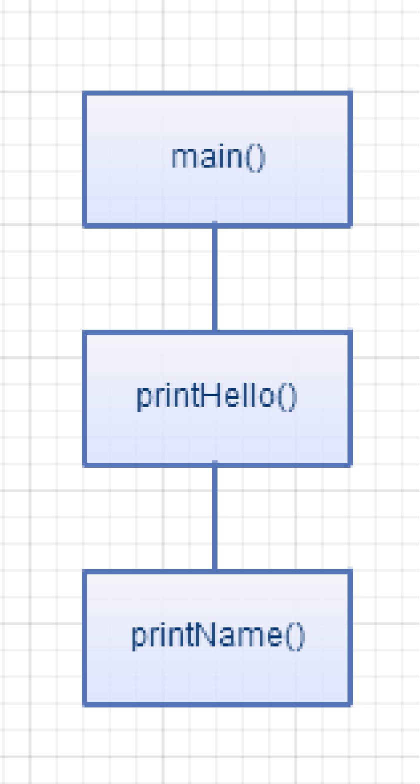
```
def printFirst():
    print("John")

def printLast():
    print("Smith")

def main():
    printFirst()
    printLast()

main()
```







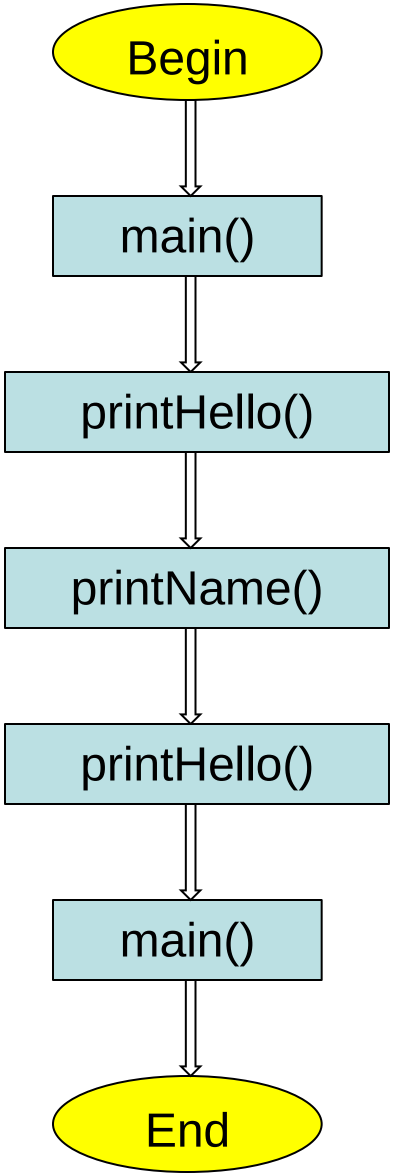
```
def printName():
  print("John Smith")

def printHello():
  printName()
  print("Hello")

def main():
  printHello()

main()
```



main()

printHello()

printName()



Begin

main()

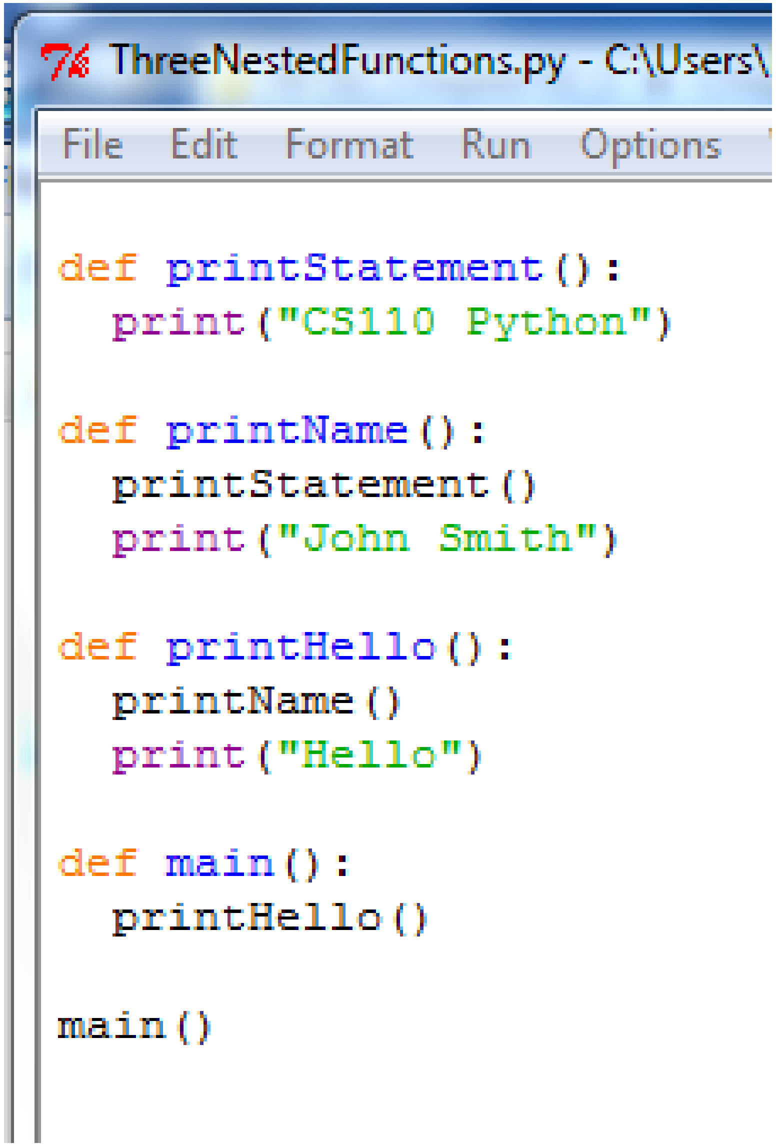
printHello()

printName()

printHello()

main()

End



```
def printStatement():
   print("CS110 Python")

def printName():
   printStatement()
   print("John Smith")

def printHello():
   printName()
   print("Hello")

def main():
   printHello()

main()
```