

The background features a light gray gradient with a subtle pattern of thin, dark gray lines and small circles, resembling a circuit board or a network diagram. These lines are more prominent on the left and right sides, framing the central text.

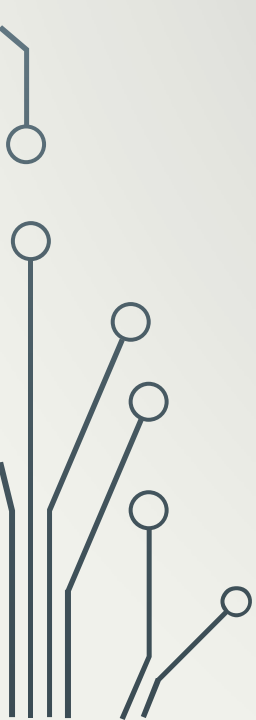

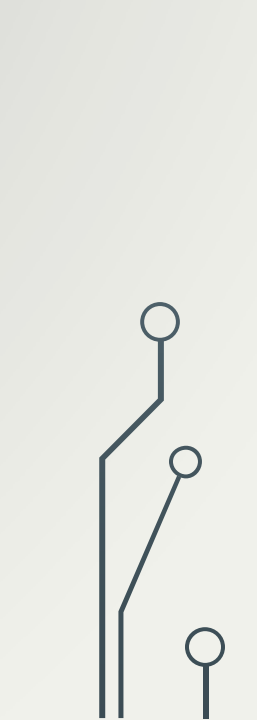
INTRODUCTION TO COMPUTERS, PROGRAMS, AND PYTHON

INTERNATIONAL COLLEGE, KMITL

PRESSESIONAL COURSE



OVERALL

- Computer
 - Programming Languages
 - Python
 - Programming style and documentation
 - Programming errors
 - Graphic programming
- 
- 
- 



COMPUTER



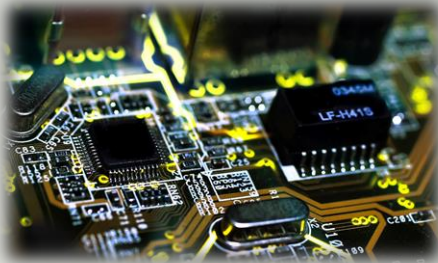
COMPUTER

A **computer** is an electronic device that stores and processes **data**.



COMPUTER

Computer uses a combination of hardware and software



Hard ware is any physical part of the computer, both internal and external components



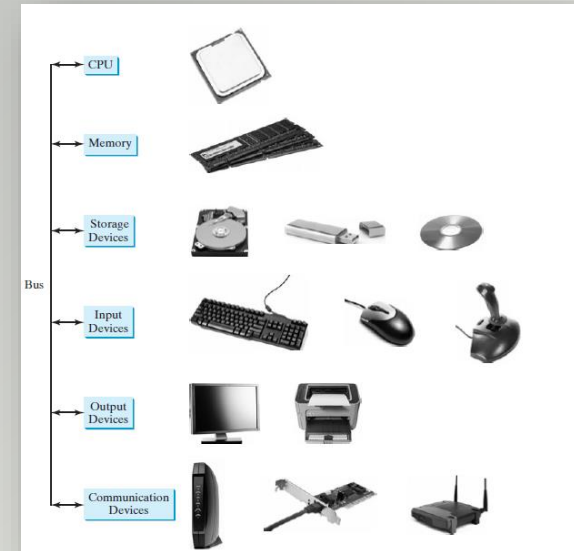
Software is any set of instructions that tells the Hardware what to do

COMPUTER

A computer consists of the following major hardware components:

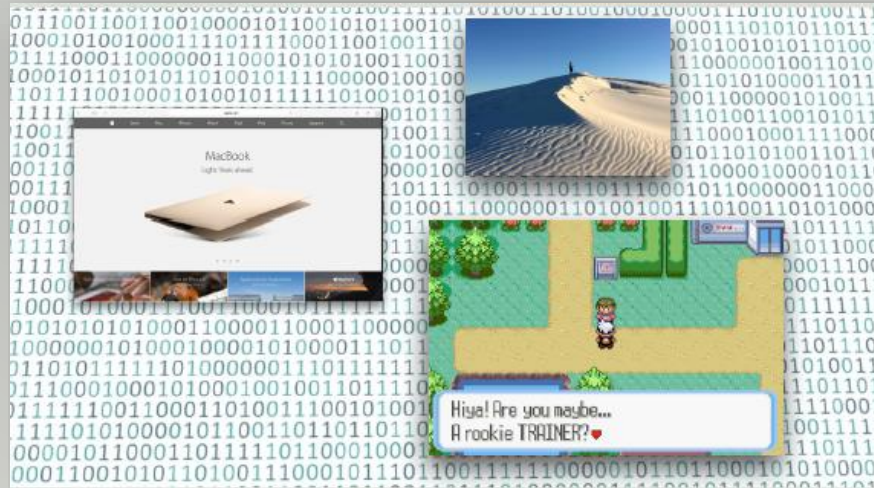
- A central Processing unit (CPU)
- Memory
- Storage devices
- Input devices
- Output devices
- Communication devices

A computer components are interconnected by a subsystem called **bus**



COMPUTER

Computer visualizes data as 0's and 1's because digital devices have two stable electrical states, on and off.



Computer knows how to combine those 0's and 1's to more complex things



COMPUTER

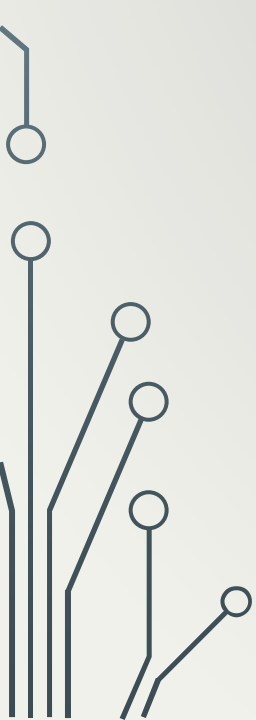
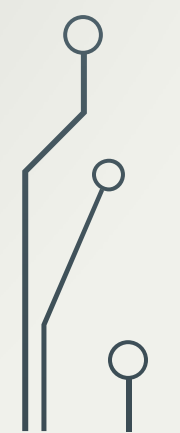
- The 0s and 1s are interpreted as digits in the binary number system and called **bits**
 - A **byte** is composed of 8 bits
 - Storage capacity is measured in bytes and multiples of the byte:
 - A kilobyte (KB) is about 1,000 bytes.
 - A megabyte (MB) is about 1 million bytes.
 - A gigabyte (GB) is about 1 billion bytes.
 - A terabyte (TB) is about 1 trillion bytes
- 
- 



PROGRAMMING LANGUAGES

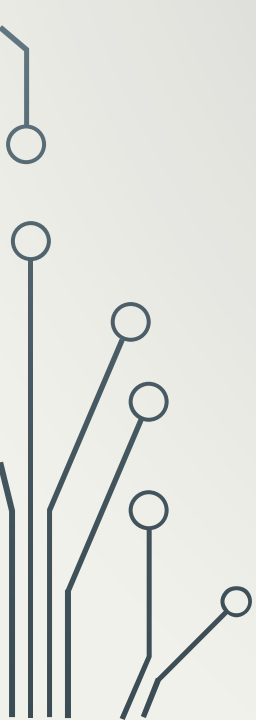
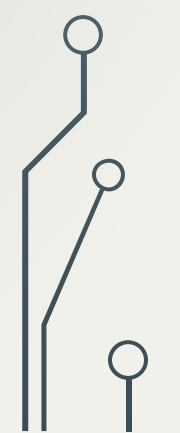


PROGRAMMING LANGUAGES

- **Machine language** is a set of built-in primitive instructions which are in the form of binary code (0's and 1's).
 - **Assembly Language** uses a short descriptive word to represent each of the machine-language instructions.
 - **High-Level Language** are English-like and easy to learn and use.
- 
- 

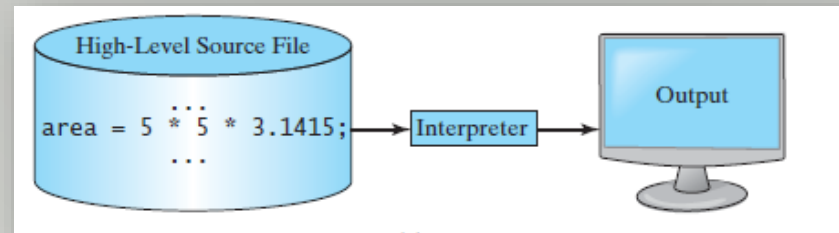


PROGRAMMING LANGUAGES

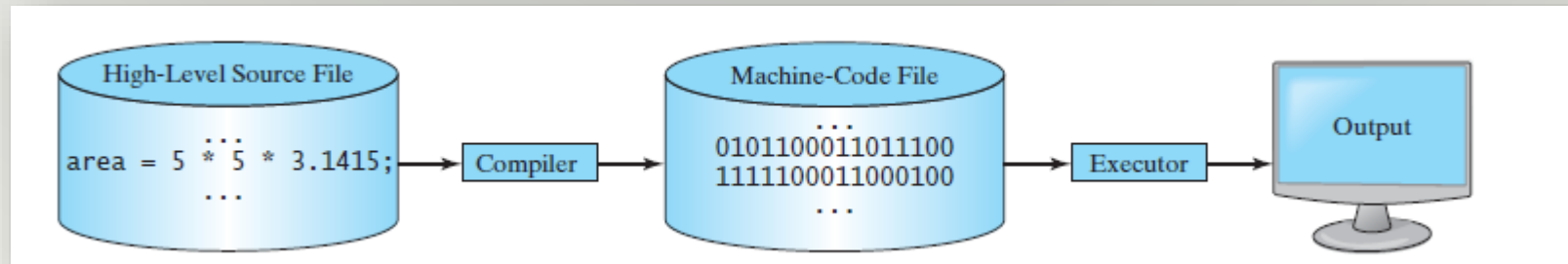
- The instruction in a High-level languages are called **statements**.
 - A program written in a high-level languages is called a **source program** or **source code**.
 - The translation can be done using another programming tool called an **interpreter** or a **compiler**.
- 
- 

PROGRAMMING LANGUAGES

- An interpreter translate and executes a program one state at a time.



- A compiler translates the entire source program into a machine-language file for execution





PYTHON

GETTING STARTED



WHAT IS pythonTM USED FOR?



Web Development



Video Game
Development



Desktop GUIs
(Graphic User Interfaces)



Software
Development

Who all are using Python?



YAHOO!

Google

YouTube



reddit

BitTorrent

IBM



Dropbox



redhat

CANONICAL

NETFLIX

Quora

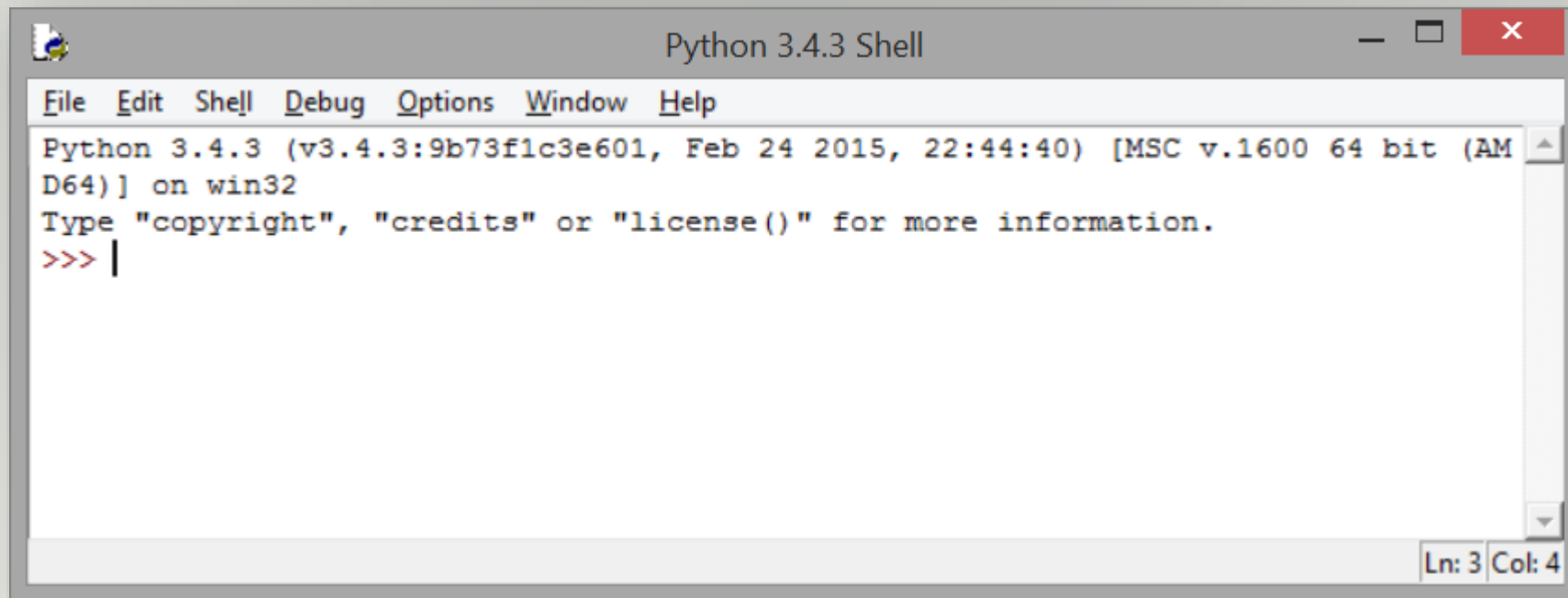


Pinterest

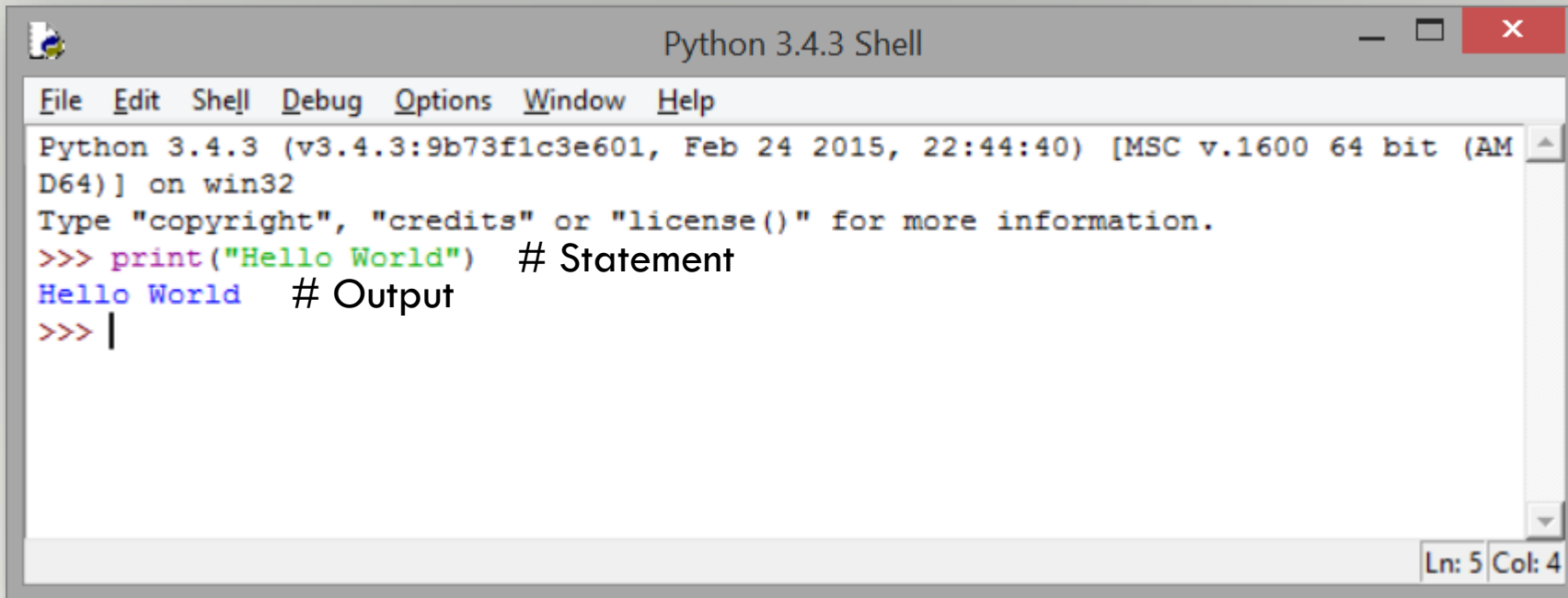
and the list goes on...

GETTING STARTED

1. Open idle



GETTING STARTED :: PRINT

A screenshot of a Python 3.4.3 Shell window. The window has a title bar with the text "Python 3.4.3 Shell" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with options: File, Edit, Shell, Debug, Options, Window, and Help. The main text area contains the following text:

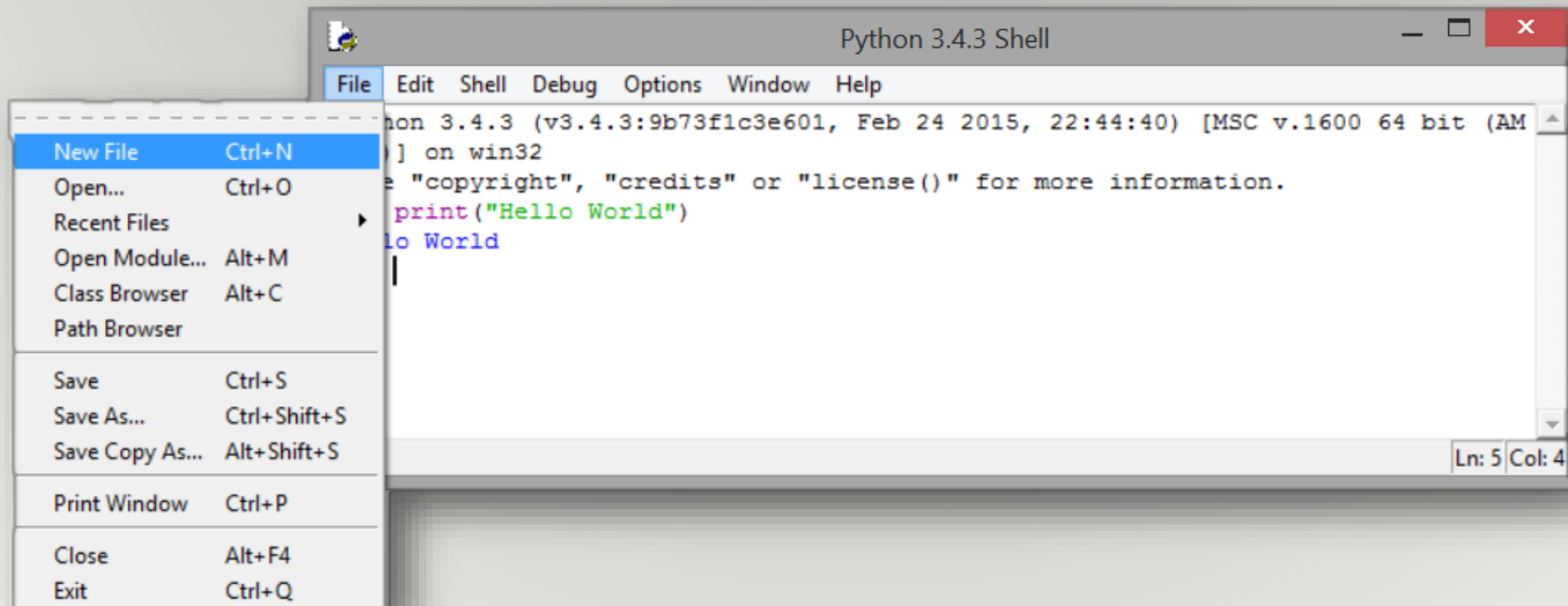
```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:44:40) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Hello World") # Statement
Hello World # Output
>>> |
```

The text is color-coded: "print" is purple, "Hello World" is green, and the prompt ">>>" is red. The output "Hello World" is blue. At the bottom right of the window, a status bar shows "Ln: 5 Col: 4".

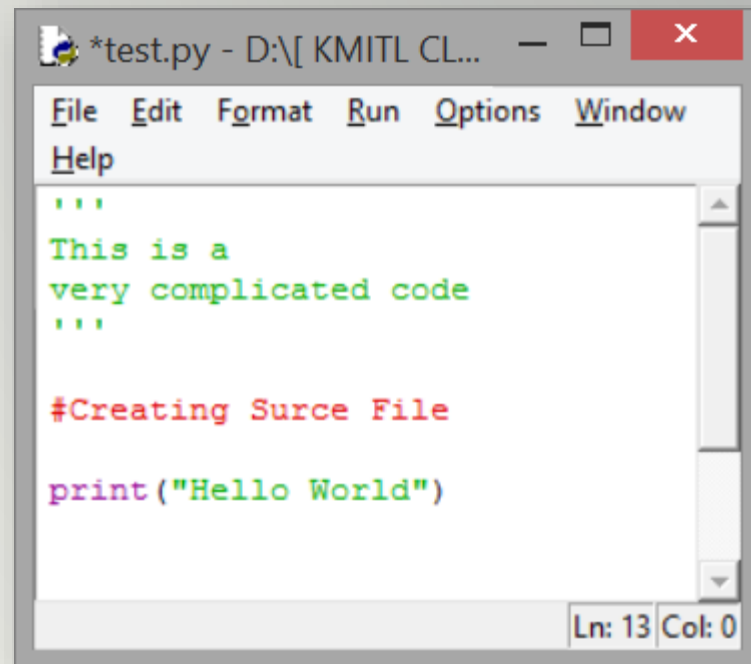
```
Python 3.4.3 Shell
File Edit Shell Debug Options Window Help
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:44:40) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Hello World") # Statement
Hello World # Output
>>> |
Ln: 5 Col: 4
```

GETTING STARTED :: CREATING NEW FILE

- Python files are named with extension .py



GETTING STARTED :: SAMPLE RUN

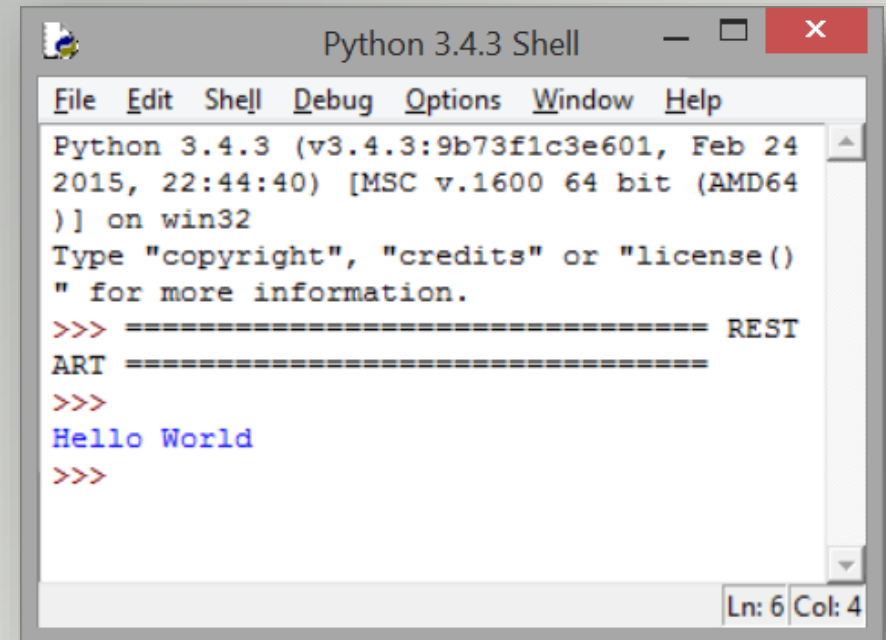


A screenshot of a Python IDE window titled '*test.py - D:\[KMITL CL...'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code editor contains the following text:

```
'''  
This is a  
very complicated code  
'''  
  
#Creating Surce File  
  
print("Hello World")
```

The status bar at the bottom right shows 'Ln: 13 Col: 0'.

F5
(run)



A screenshot of a Python 3.4.3 Shell window titled 'Python 3.4.3 Shell'. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The output text is as follows:

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24  
2015, 22:44:40) [MSC v.1600 64 bit (AMD64  
)] on win32  
Type "copyright", "credits" or "license()  
" for more information.  
>>> ===== REST  
ART =====  
>>>  
Hello World  
>>>
```

The status bar at the bottom right shows 'Ln: 6 Col: 4'.

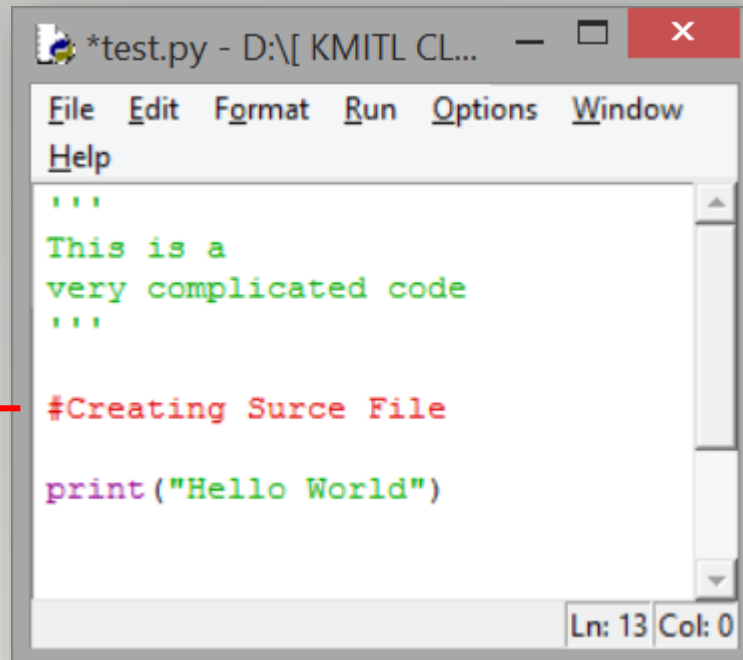
GETTING STARTED :: COMMENT

Paragraph comment

'''

Line comment

#



The screenshot shows a window titled '*test.py - D:\[KMITL CL...'. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the following text:

```
'''  
This is a  
very complicated code  
'''  
  
#Creating Surce File  
  
print("Hello World")
```

The status bar at the bottom right indicates 'Ln: 13 Col: 0'.

GETTING STARTED :: SPECIAL CHARACTER

TABLE 1.2 Special Characters

<i>Character</i>	<i>Name</i>	<i>Description</i>
<code>()</code>	Opening and closing parentheses	Used with functions.
<code>#</code>	Pound sign	Precedes a comment line.
<code>" "</code>	Opening and closing quotation marks	Encloses a string (i.e., sequence of characters).
<code>''' '''</code>	Paragraph comments	Encloses a paragraph comment.

GETTING STARTED :: MATHEMATICAL OPERATOR

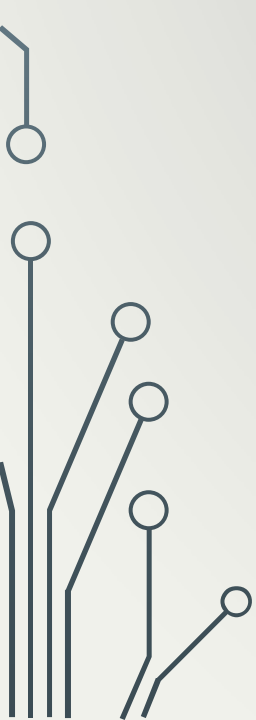
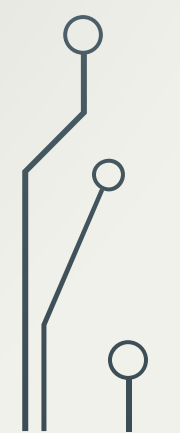
<i>Name</i>	<i>Meaning</i>	<i>Example</i>	<i>Result</i>
+	Addition	34 + 1	35
-	Subtraction	34.0 - 0.1	33.9
*	Multiplication	300 * 30	9000
/	Float Division	1 / 2	0.5
//	Integer Division	1 // 2	0
**	Exponentiation	4 ** 0.5	2.0
%	Remainder	20 % 3	2

The image features a light gray background with decorative circuit-like lines in the corners. These lines are composed of straight segments and small circles, resembling a stylized electronic circuit or a network diagram. They are positioned in the top-left, top-right, bottom-left, and bottom-right corners, framing the central text.

PROGRAMMING STYLE & DOCUMENTATION



APPROPRIATE COMMENTS AND COMMENT STYLES

- Explain what the program does, its key features.
 - Introduce each major step
 - Explain anything that is difficult to read
 - Be concise
- 
- 

PROPER SPACING

```
print(3+4*4)
```

← Bad style

```
print(3 + 4 * 4)
```

← Good style



PROGRAMMING ERRORS



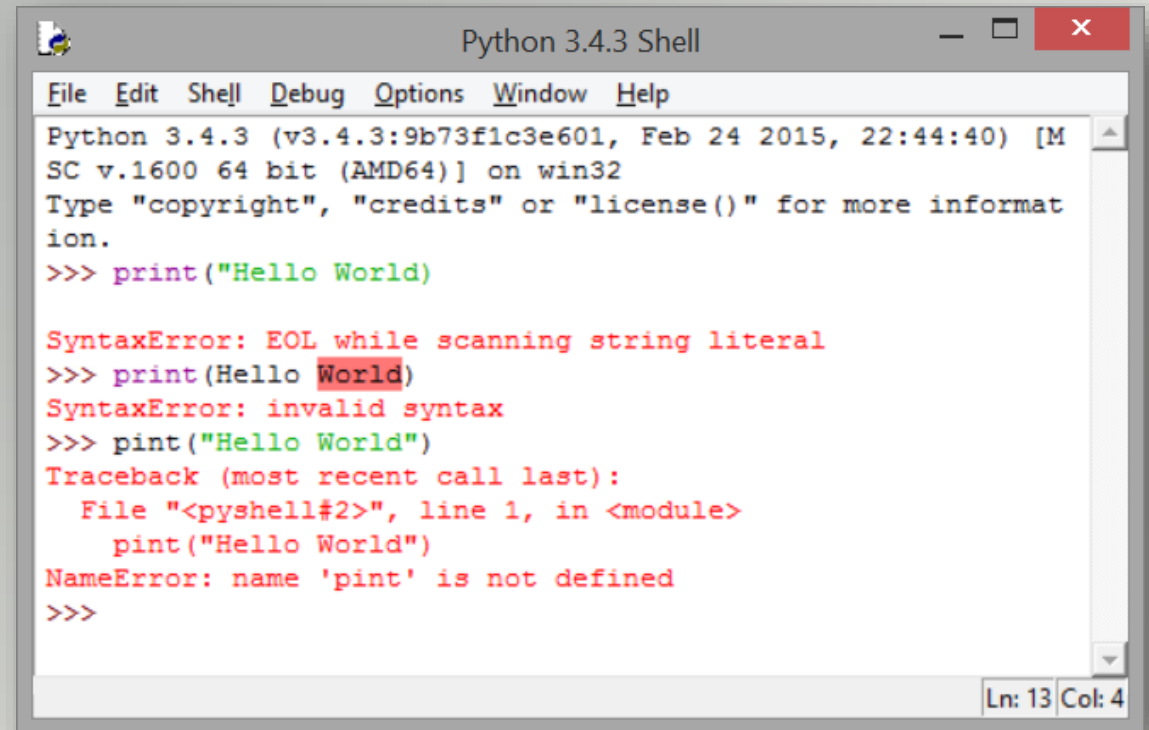
PROGRAMMING ERRORS

- Syntax Error
 - Run-time Error
 - Logic Error
- Cause program to terminate
- Lead to unexpected output



SYNTAX ERRORS

- Mistyping a statement
- Incorrect indentation
- Omitting some necessary punctuation
- Incomplete parenthesis format

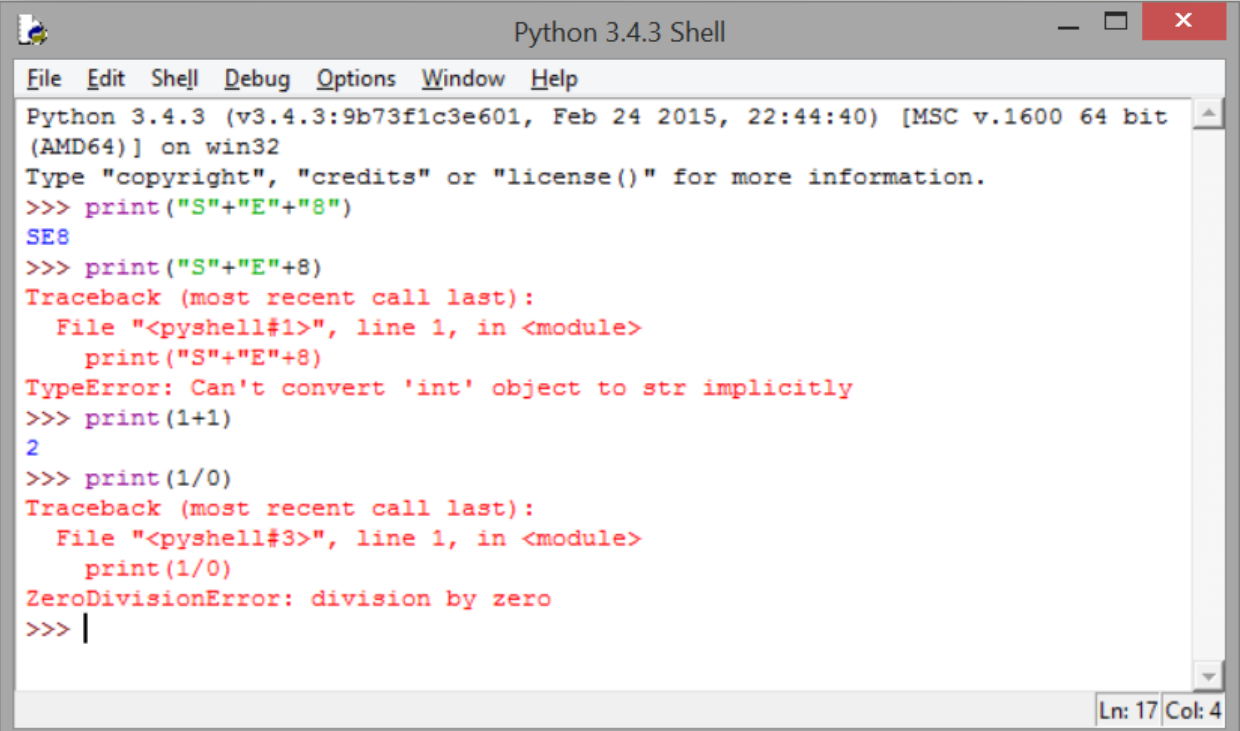
A screenshot of a Python 3.4.3 Shell window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following content:

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:44:40) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Hello World)
SyntaxError: EOL while scanning string literal
>>> print(Hello World)
SyntaxError: invalid syntax
>>> pint("Hello World")
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    pint("Hello World")
NameError: name 'pint' is not defined
>>>
```

The status bar at the bottom right shows 'Ln: 13 Col: 4'. The window title is 'Python 3.4.3 Shell'.

RUN-TIME ERROR

- Input Error
- Mathematic Error
- Variable's type conflict



The screenshot shows a Python 3.4.3 Shell window with the following content:

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:44:40) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("S"+"E"+"8")
SE8
>>> print("S"+"E"+8)
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    print("S"+"E"+8)
TypeError: Can't convert 'int' object to str implicitly
>>> print(1+1)
2
>>> print(1/0)
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    print(1/0)
ZeroDivisionError: division by zero
>>> |
```

The status bar at the bottom right indicates "Ln: 17 Col: 4".

LOGIC ERROR

- Difficult to located
- Provided unexpected outcome
- Program still compiled

```
>>> PI = 3.14
>>> r = 2
>>> area = (PI * r) ** 2
>>> area
39.4384
```

Wrong Answer

```
>>> PI = 3.14
>>> r = 2
>>> area = PI * (r ** 2)
>>> area
12.56
```

Correct Answer



GRAPHIC PROGRAMMING

AKA. TURTLE



TURTLE :: IMPORT MODULE

```
>>>  
>>> import turtle          >>> turtle.forward(100)  
>>>  
>>> import turtle as t     >>> t.forward(100)  
>>>  
>>> from turtle import *   >>> forward(100)  
>>>
```

TURTLE :: METHODS

MOVE AND DRAW

Method	Description	Parameter
forward()	Move turtle forward	1
backward()	Move turtle backward	1
right()	Turn right	1
left()	Turn left	1
goto()	Move to coordinate	2
home()	Move to point(0,0)	None
circle()	Draw a circle	...

TURTLE :: METHODS

PEN CONTROL

Method	Description	Parameter
pendown()	Pull the pen down	None
penup()	Pull the pen up	None
pensize()	Set the line thickness	1
color()	Set color to be filled	1
pencolor()	Set pen color	1
begin_fill()	Called before filling color of shape	None
end_fill()	Used to end begin_fill()	None
clear()	Delete all turtle's drawings	None
write()	Write text	...

TURTLE :: METHODS

TURTLE CONTROL

Method	Description	Parameter
speed()	Specify turtle's speed	1
showturtle()	Display turtle (arrow)	None
hideturtle()	Hide turtle (arrow)	None
done()	Causes program to pause for displaying the graphic	None

TURTLE :: EXAMPLE

