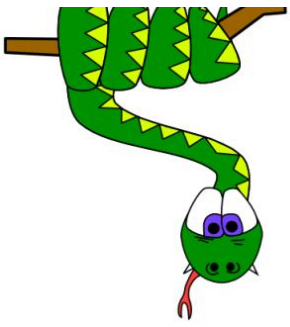




# Introduction To Programming

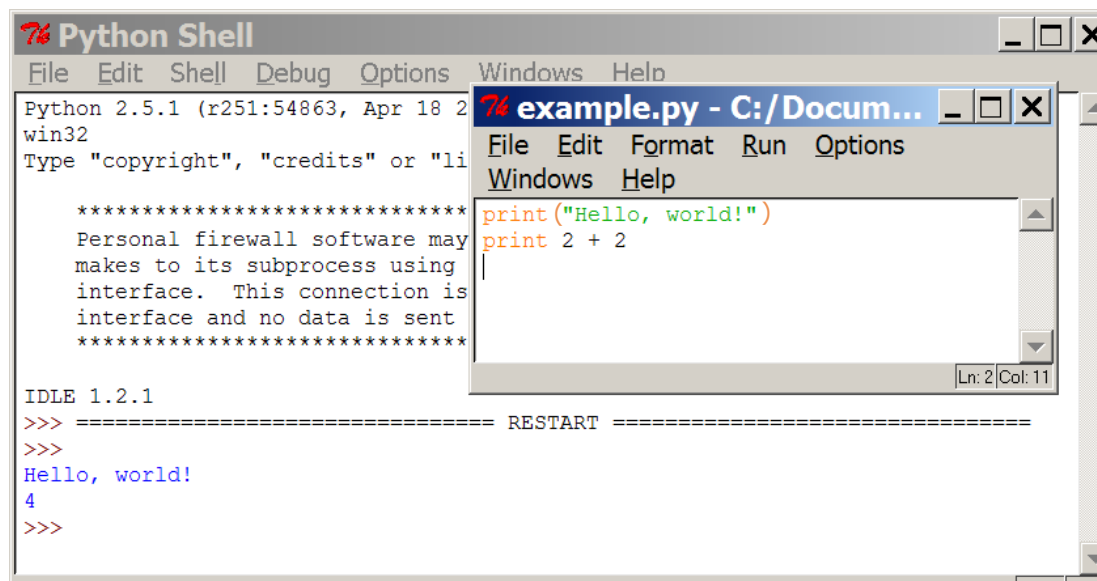
Assist. Prof. Dr. Abdul Hadi Mohammed



# Introduction to Programming with Python

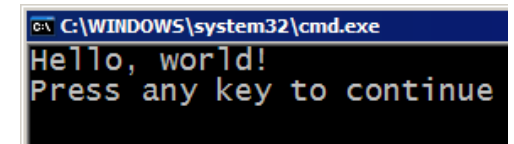
# Programming basics

- **code** or **source code**: The sequence of instructions in a program.
- **syntax**: The set of legal structures and commands that can be used in a particular programming language.
- **output**: The messages printed to the user by a program.
- **console**: The text box onto which output is printed.
  - Some source code editors pop up the console as an external window, and others contain their own console window.



The image shows two overlapping windows from a Python IDE. The background window is titled 'Python Shell' and shows the IDLE 1.2.1 prompt with a 'RESTART' message and the output 'Hello, world!' and '4'. The foreground window is titled 'example.py - C:/Docum...' and shows a code editor with the following code:

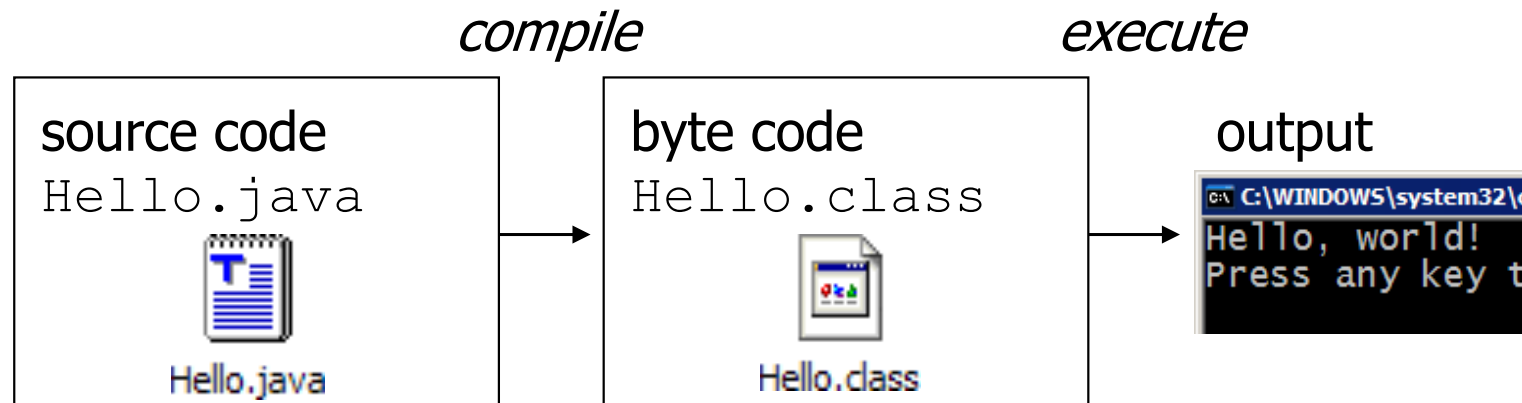
```
print("Hello, world!")
print 2 + 2
```



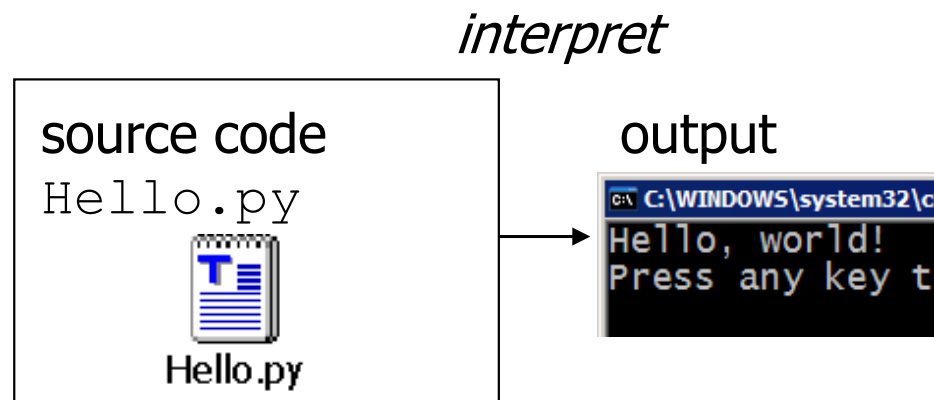
The image shows a Windows command prompt window titled 'C:\WINDOWS\system32\cmd.exe'. It displays the output 'Hello, world!' and the prompt 'Press any key to continue'.

# Compiling and interpreting

- Many languages require you to *compile* (translate) your program into a form that the machine understands.



- Python is instead directly *interpreted* into machine instructions.



# The Python Interpreter

- Python is an interpreted language
- The interpreter provides an interactive environment to play with the language
- Results of expressions are printed on the screen

```
>>> 3 + 7
10
>>> 3 < 15
True
>>> 'print me'
'print me'
>>> print 'print me'
print me
>>>
```

# Expressions

- **expression:** A data value or set of operations to compute a value.

Examples:  $1 + 4 * 3$   
 $42$

- Arithmetic operators we will use:

$+$	$-$	$*$	$/$	addition, subtraction/negation, multiplication, division
$\%$				modulus, a.k.a. remainder
$**$				exponentiation

- **precedence:** Order in which operations are computed.

- $*$   $/$   $\%$   $**$  have a higher precedence than  $+$   $-$

$1 + 3 * 4$  is 13

- Parentheses can be used to force a certain order of evaluation.

$(1 + 3) * 4$  is 16

# Integer division

- When we divide integers with  $/$ , the quotient is also an integer.

$$\begin{array}{r} 3 \\ 4 \overline{) 14} \\ \underline{12} \\ 2 \end{array}$$

$$\begin{array}{r} 52 \\ 27 \overline{) 1425} \\ \underline{135} \\ 75 \\ \underline{54} \\ 21 \end{array}$$

- More examples:

- $35 / 5$  is 7
- $84 / 10$  is 8
- $156 / 100$  is 1

- The  $\%$  operator computes the remainder from a division of integers.

$$\begin{array}{r} 3 \\ 4 \overline{) 14} \\ \underline{12} \\ \mathbf{2} \end{array}$$

$$\begin{array}{r} 43 \\ 5 \overline{) 218} \\ \underline{20} \\ 18 \\ \underline{15} \\ \mathbf{3} \end{array}$$

# Real numbers

- Python can also manipulate real numbers.
  - Examples: `6.022`      `-15.9997`      `42.0`      `2.143e17`
- The operators `+` `-` `*` `/` `%` `**` `()` all work for real numbers.
  - The `/` produces an exact answer: `15.0 / 2.0` is `7.5`
  - The same rules of precedence also apply to real numbers:  
Evaluate `()` before `*` `/` `%` before `+` `-`
- When integers and reals are mixed, the result is a real number.
  - Example: `1 / 2.0` is `0.5`
  - The conversion occurs on a per-operator basis.

$$\begin{array}{rcl} 7 / 3 * 1.2 + 3 / 2 & & \\ \underline{2} * 1.2 + 3 / 2 & & \\ 2.4 + 3 / 2 & & \\ 2.4 + \underline{1} & & \\ 3.4 & & \end{array}$$



# Math commands

- Python has useful commands (or called functions) for performing calculations.

Function	Description	Example	Module Required
<code>abs(value)</code>	Returns the absolute value of value.	<code>abs(-5)</code> # Output: 5	No
<code>math.ceil(value)</code>	Rounds up to the nearest whole number.	<code>math.ceil(4.2)</code> # Output: 5	math
<code>math.cos(value)</code>	Returns the cosine of value (in radians).	<code>math.cos(math.pi)</code> # Output: -1.0	math
<code>math.floor(value)</code>	Rounds down to the nearest whole number.	<code>math.floor(4.8)</code> # Output: 4	math
<code>math.log(value)</code>	Returns the natural logarithm (base e).	<code>math.log(10)</code> # Output: 2.302585092994046	math
<code>math.log10(value)</code>	Returns the logarithm of value to base 10.	<code>math.log10(100)</code> # Output: 2.0	math
<code>max(value1, value2, ...)</code>	Returns the largest value.	<code>max(3, 7, 2)</code> # Output: 7	No
<code>min(value1, value2, ...)</code>	Returns the smallest value.	<code>min(3, 7, 2)</code> # Output: 2	No
<code>round(value)</code>	Rounds value to the nearest whole number.	<code>round(4.5)</code> # Output: 4	No
<code>math.sin(value)</code>	Returns the sine of value (in radians).	<code>math.sin(math.pi / 2)</code> # Output: 1.0	math
<code>math.sqrt(value)</code>	Returns the square root of value.	<code>math.sqrt(16)</code> # Output: 4.0	math

- To use many of these commands, you must write the following at the top of your Python program: `import math`

# Numbers: Floating Point

- `int(x)` converts `x` to an integer
- `float(x)` converts `x` to a floating point
- The interpreter shows a lot of digits

```
>>> 1.23232
1.232320000000000001
>>> print 1.23232
1.23232
>>> 1.3E7
13000000.0
>>> int(2.0)
2
>>> float(2)
2.0
```

# Variables

- **variable:** A named piece of memory that can store a value.

- Usage:

- Compute an expression's result,
- store that result into a variable,
- and use that variable later in the program.



- **assignment statement:** Stores a value into a variable.

- Syntax:

***name = value***

- Examples:

`x = 5`

`gpa = 3.14`

x 

5
---

gpa 

3.14
------

- A variable that has been given a value can be used in expressions.

`x + 4` is 9

- **Exercise:** Evaluate the quadratic equation for a given  $a$ ,  $b$ , and  $c$ .

# Example

```
>>> x = 7
>>> x
7
>>> x+7
14
>>> x = 'hello'
>>> x
'hello'
>>>
```

# print

- `print` : Produces text output on the console.

- Syntax:

```
print "Message"
```

```
print Expression
```

- Prints the given text message or expression value on the console, and moves the cursor down to the next line.

```
print Item1, Item2, ..., ItemN
```

- Prints several messages and/or expressions on the same line.

- Examples:

```
print ("Hello, world!")
```

```
age = 45
```

```
print ("You have", 65 - age, "years until retirement")
```

## Output:

```
Hello, world!
```

```
You have 20 years until retirement
```

# Example: print Statement

- Elements separated by commas print with a space between them
- A comma at the end of the statement (`print 'hello',`) will not print a newline character

```
>>> print 'hello'
```

```
hello
```

```
>>> print 'hello', 'there'
```

```
hello there
```

# input

- `input` : Reads a number from user input.
  - You can assign (store) the result of `input` into a variable.
  - Example:

```
age = input("How old are you? ")
print "Your age is", age
print "You have", 65 - age, "years until retirement"
```

Output:

```
How old are you? 53
Your age is 53
You have 12 years until retirement
```

- **Exercise:** Write a Python program that prompts the user for his/her amount of money, then reports how many Nintendo Wiis the person can afford, and how much more money he/she will need to afford an additional Wii.

# Input: Example

```
print("What's your name?")
name = input("> ")
print("What year were you born?")
birthyear = int(input("> "))
current_year = 2016
age = current_year - birthyear
print(f"Hi {name}! You are {age} years old.")
```

```
% python input.py
What's your name?
> Michael
What year were you born?
> 1980
Hi Michael! You are 31
```