

Python - Input, output and variables

Lecture 22 - COMPSCI111/111G SS 2016

Today's lecture

- ▶ What is Python?
- ▶ Displaying text on screen using `print()`
- ▶ Variables
- ▶ Numbers and basic arithmetic
- ▶ Getting input from keyboard using `input()`

What is a programming language?

- ▶ A formal language that specifies how to perform a computational task
- ▶ Many programming languages exist:
 - ▶ Visual Basic
 - ▶ C and C++
 - ▶ C#
 - ▶ Java
 - ▶ Python
- ▶ Python was created in 1989 by Guido Van Rossum in The Netherlands

Statements

- ▶ A program consists of a series of commands called **statements**
- ▶ They are generally executed (ie. run) in the order they appear
- ▶ The statements must be written correctly otherwise you will get a syntax error
- ▶ Python programs are saved in files with the `' .py'` extension

Translating code

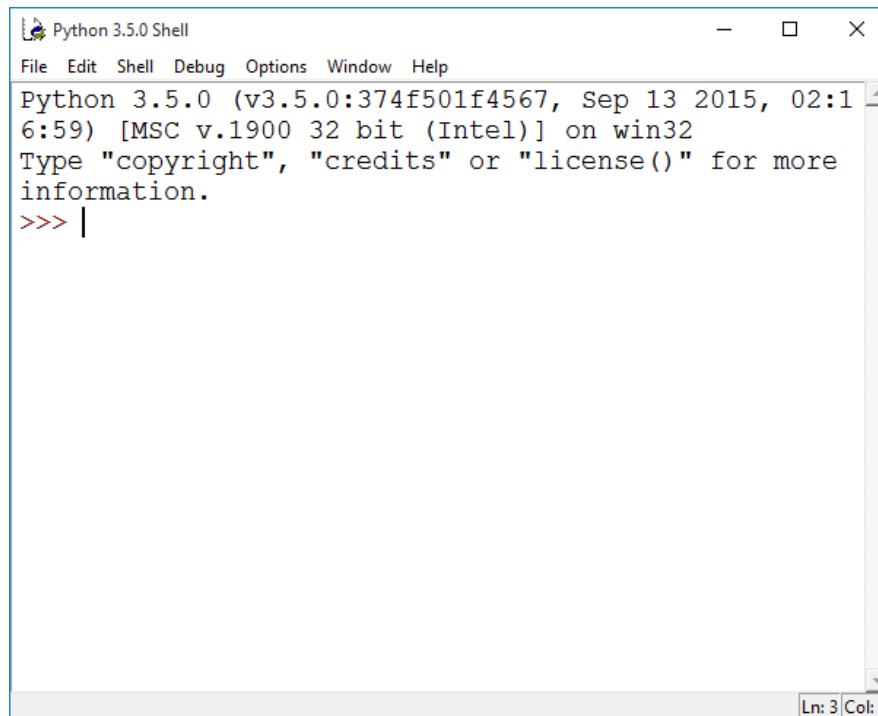
- ▶ The statements in our programs are translated into simpler instructions that the CPU can execute
- ▶ Two ways of doing this:
 - ▶ Compiler: translates the entire program file at once
 - ▶ Interpreter: repeatedly translates one line and runs it
- ▶ Python is an interpretative programming language
 - ▶ There are also compilers available for Python

IDLE Integrated Development Environment (IDE)

- ▶ An IDE is used by programmers to:
 - ▶ Write code
 - ▶ Check for errors
 - ▶ Translate code and run the program
- ▶ We use the IDLE IDE; a popular IDE for Python
- ▶ IDLE has a shell for the Python interpreter
- ▶ You can also create a new file that can be compiled when you've finished writing a program

IDLE IDE

- ▶ The interpreter allows you to type statements, translate them and see them run instantly
- ▶ Very helpful for experimentation and learning

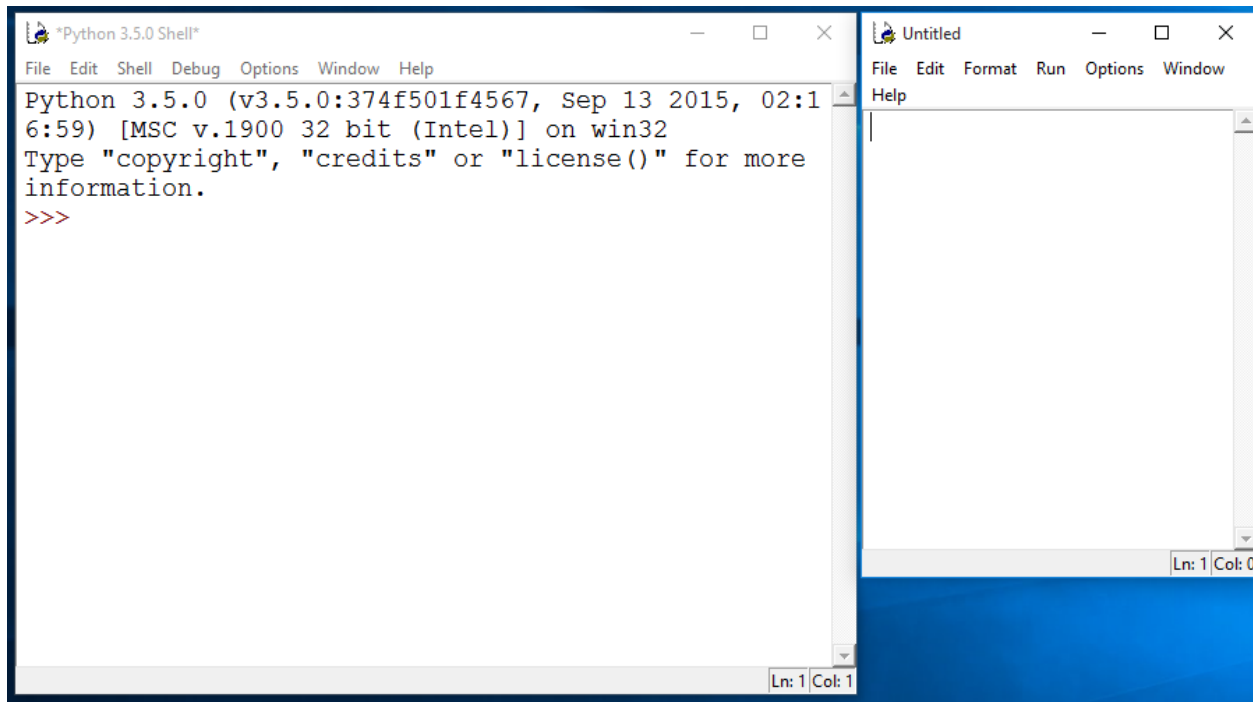


```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more
information.
>>> |
```

Ln: 3 Col: 4

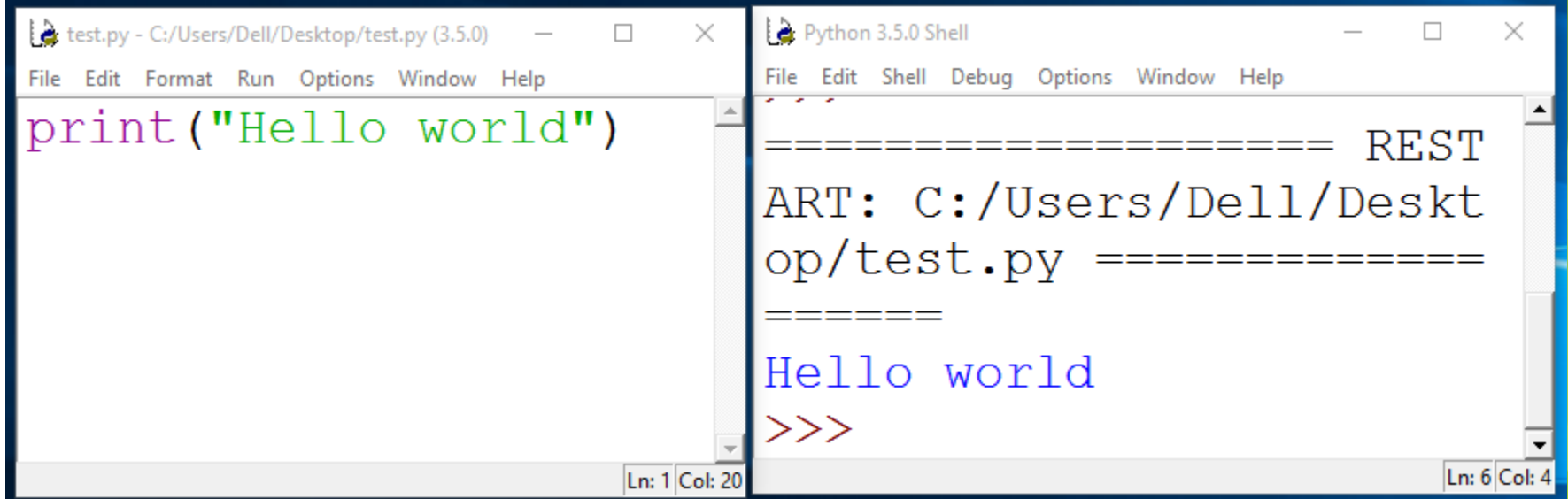
IDLE IDE

- ▶ Create a new program by clicking on File → New File
- ▶ Type your statements in the file, then click on Run → Run Module...



“Hello world”

- ▶ Traditional first program is displaying “Hello World” on screen
- ▶ To display text on screen you use the `print()` function



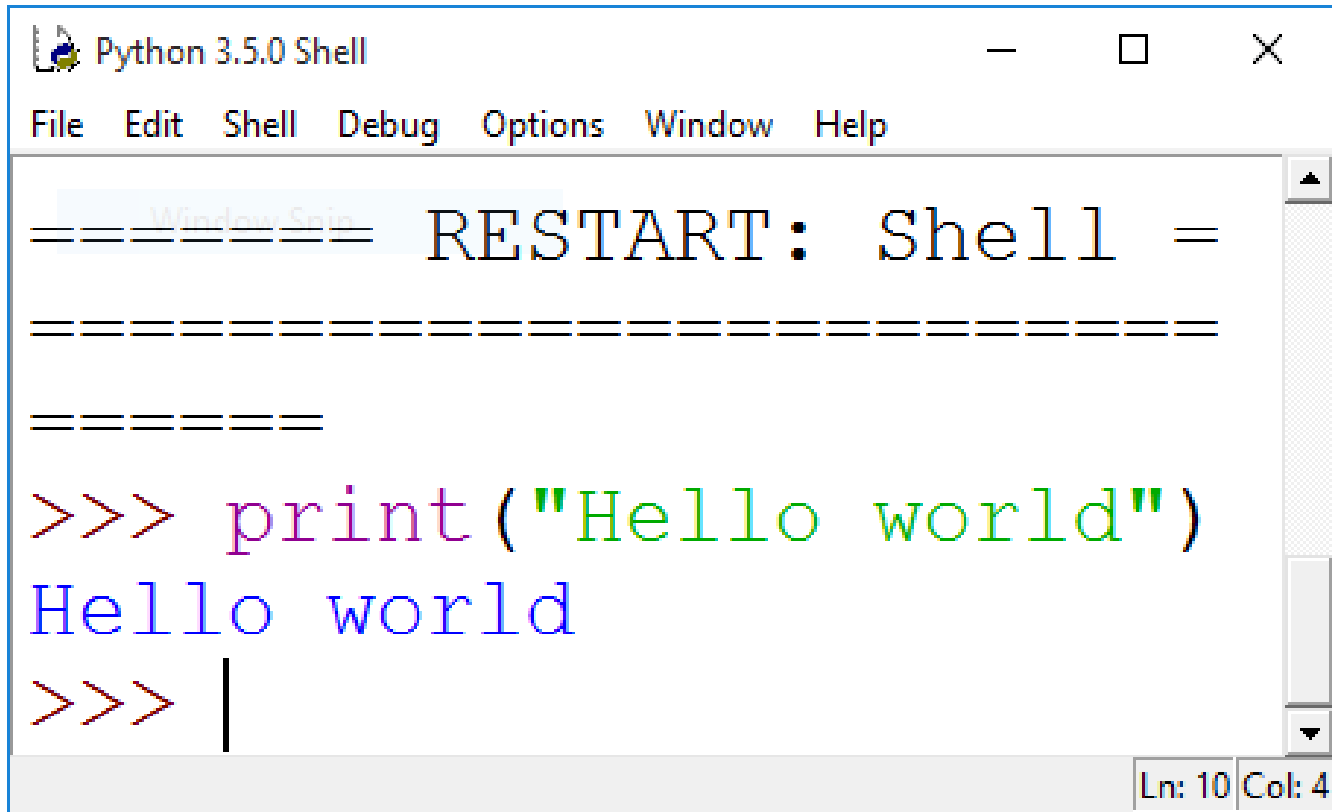
The image shows two side-by-side windows from a Python IDE. The left window, titled 'test.py - C:/Users/Dell/Desktop/test.py (3.5.0)', contains the code `print("Hello world")`. The right window, titled 'Python 3.5.0 Shell', shows the output of the program: a separator line of equals signs, the path 'C:/Users/Dell/Desktop/test.py', another separator line, and the text 'Hello world'. The shell prompt `>>>` is visible at the bottom.

```
test.py - C:/Users/Dell/Desktop/test.py (3.5.0)
File Edit Format Run Options Window Help
print("Hello world")
Ln: 1 Col: 20

Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
===== REST
ART: C:/Users/Dell/Desktop/test.py =====
=====
Hello world
>>>
Ln: 6 Col: 4
```

“Hello world”

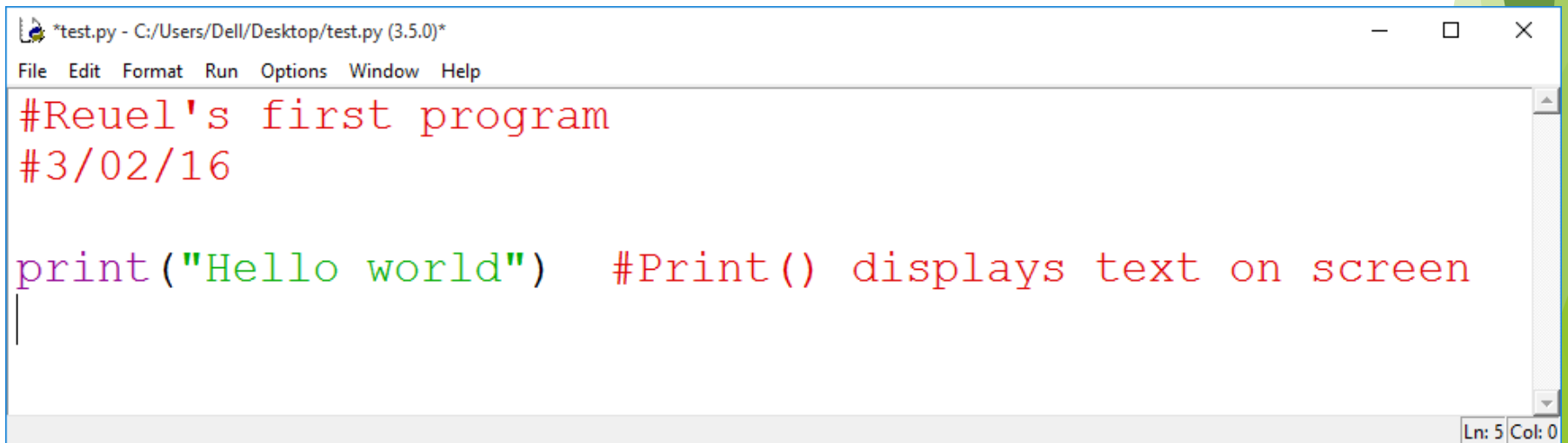
- ▶ Using the Python interpreter:



```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
===== RESTART: Shell =
=====
=====
>>> print("Hello world")
Hello world
>>> |
Ln: 10 Col: 4
```

Comments

- ▶ When writing a program, it is helpful to leave comments in the code
- ▶ You can write a comment in Python by typing a '#' in front of the line
- ▶ The compiler will ignore all text after the '#'



The screenshot shows a window titled '*test.py - C:/Users/Dell/Desktop/test.py (3.5.0)*'. The window contains the following Python code:

```
#Reuel's first program
#3/02/16

print("Hello world") #Print() displays text on screen
```

The status bar at the bottom right of the window indicates 'Ln: 5 | Col: 0'.

Data types

▶ Strings:

- ▶ Sequence of characters
- ▶ Plain text (ASCII or Unicode)
- ▶ Enclosed in quote marks
- ▶ Eg: "Hello", "Goodbye"

▶ Integers:

- ▶ Whole numbers (ie. without a decimal point)
- ▶ Eg. -100, 0, 45

▶ Floating point numbers:

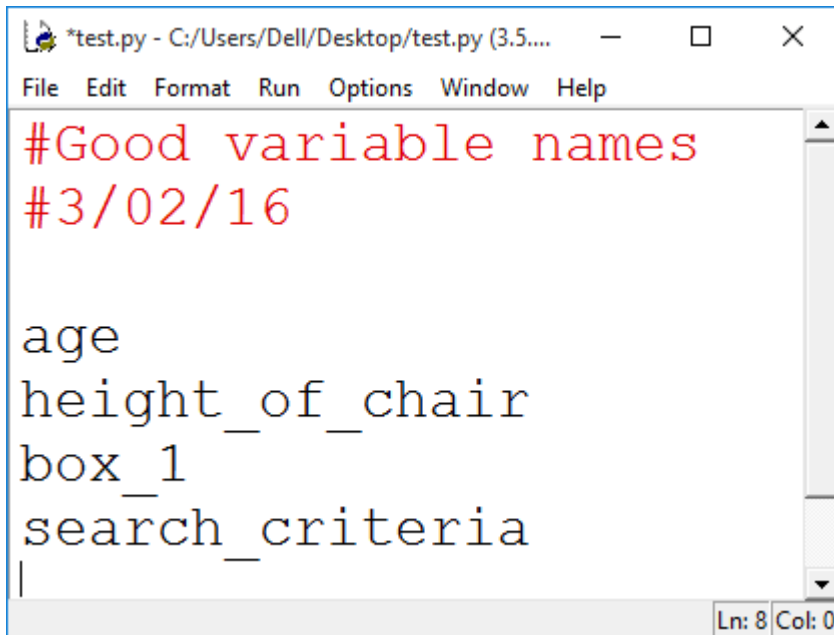
- ▶ Numbers with a decimal point
- ▶ Eg. 5.2, -1.002, 0.0

Variables

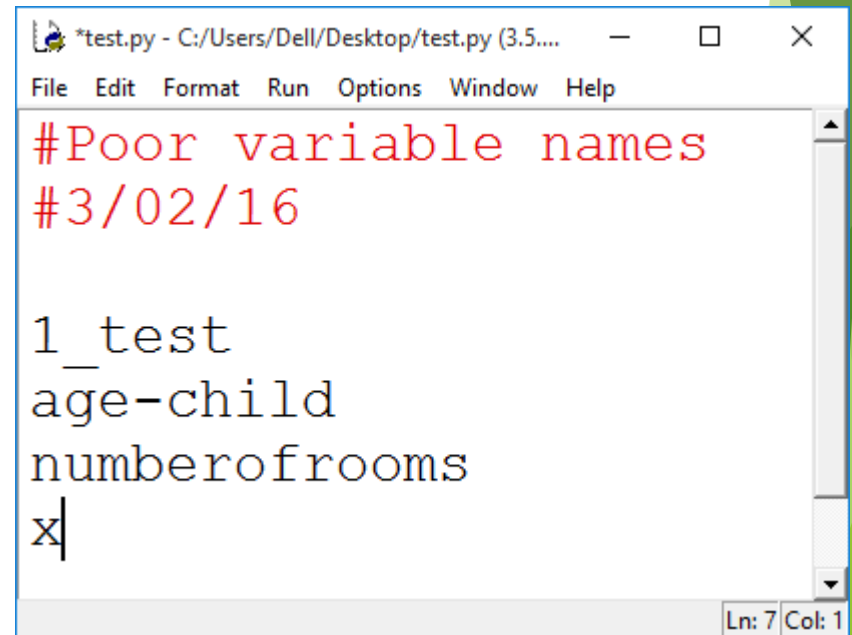
- ▶ A 'container' in the computer's memory in which you can store data
- ▶ A variable's value can change when the program runs
- ▶ Python variables are loosely-typed; they can hold any data type

Variables

- ▶ Rules to follow when naming your variables:
 - ▶ Names should reflect what is stored in the variable
 - ▶ Can begin with a letter or underscore (eg. '_')
 - ▶ Variable names can include numbers
 - ▶ Generally, all words are lowercase and words are separated using an underscore



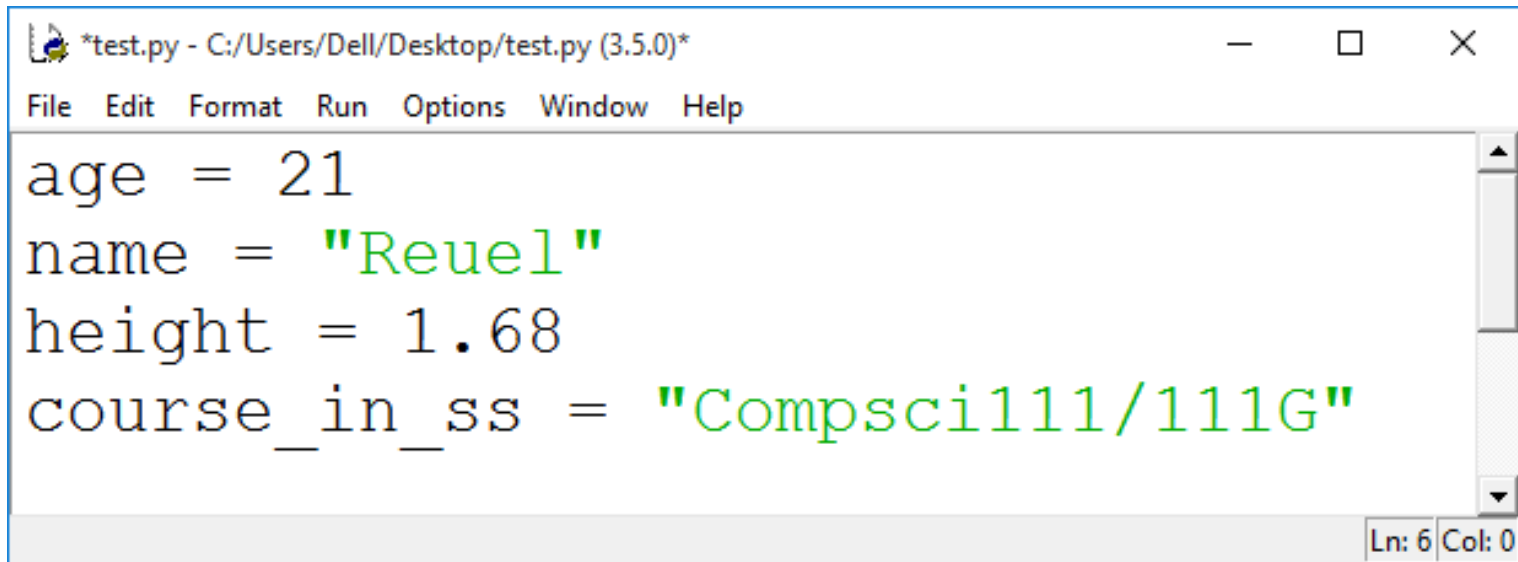
```
*test.py - C:/Users/Dell/Desktop/test.py (3.5...  
File Edit Format Run Options Window Help  
#Good variable names  
#3/02/16  
  
age  
height_of_chair  
box_1  
search_criteria  
|  
Ln: 8 Col: 0
```



```
*test.py - C:/Users/Dell/Desktop/test.py (3.5...  
File Edit Format Run Options Window Help  
#Poor variable names  
#3/02/16  
  
1_test  
age-child  
numberofrooms  
x|  
Ln: 7 Col: 1
```

Variables

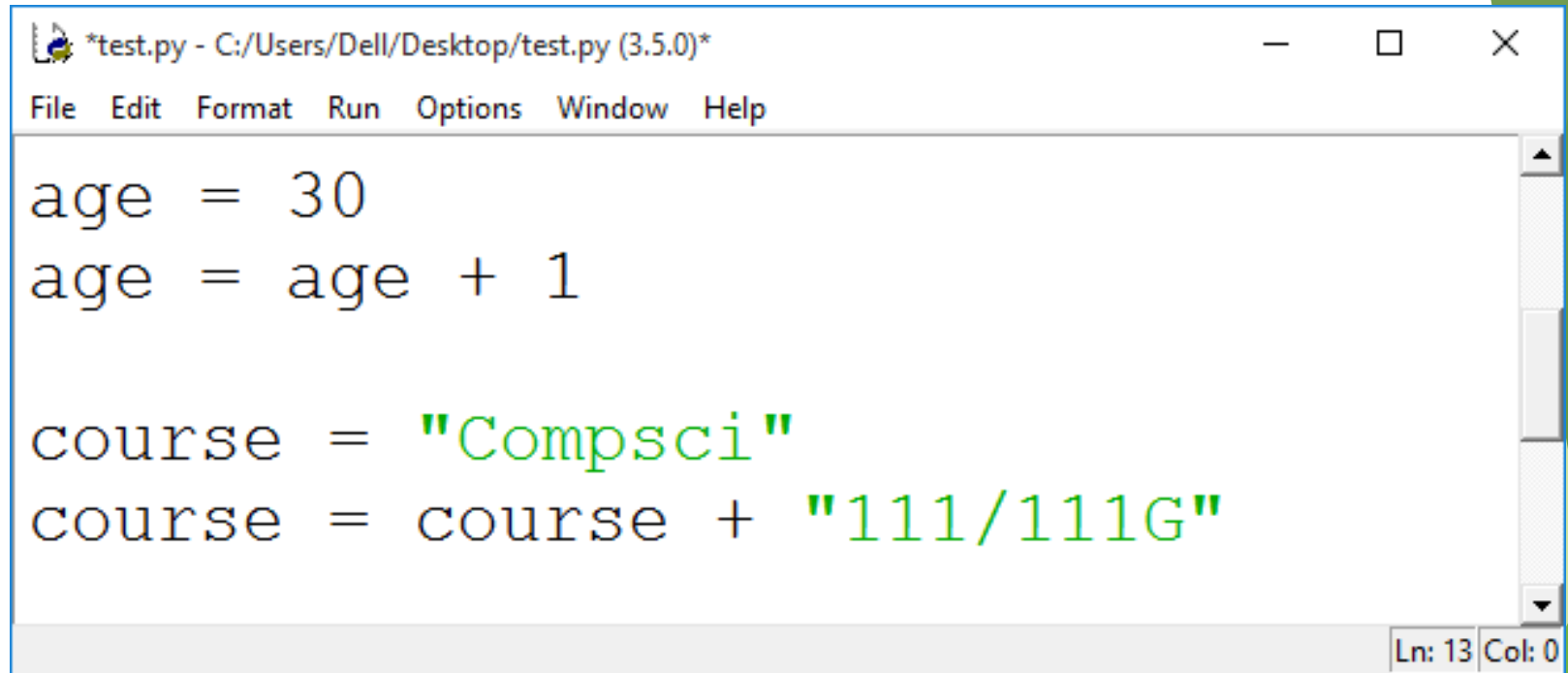
- ▶ Assigning a value to a variable:

A screenshot of a Python IDE window titled '*test.py - C:/Users/Dell/Desktop/test.py (3.5.0)*'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The main text area contains four lines of Python code: 'age = 21', 'name = "Reuel"', 'height = 1.68', and 'course_in_ss = "Compsci111/111G"'. The code is displayed in a monospaced font. The status bar at the bottom right shows 'Ln: 6 Col: 0'.

```
*test.py - C:/Users/Dell/Desktop/test.py (3.5.0)*
File Edit Format Run Options Window Help
age = 21
name = "Reuel"
height = 1.68
course_in_ss = "Compsci111/111G"
Ln: 6 Col: 0
```

Variables

- ▶ Changing the value in a variable:



The screenshot shows a Python IDE window titled "*test.py - C:/Users/Dell/Desktop/test.py (3.5.0)*". The window contains the following Python code:

```
age = 30
age = age + 1

course = "Compsci"
course = course + "111/111G"
```

The status bar at the bottom right of the window indicates "Ln: 13 Col: 0".

Arithmetic operations

Operation	Symbol	Example
Exponent	**	$2 ** 3 = 8$
Multiply	*	$2 * 2 = 4$
Divide	/	$10 / 3 = 3.333$
Divide (integer)	//	$10 // 3 = 3$
Remainder	%	$10 \% 3 = 1$
Add	+	$8 + 9 = 17$
Subtract	-	$9 - 7 = 2$

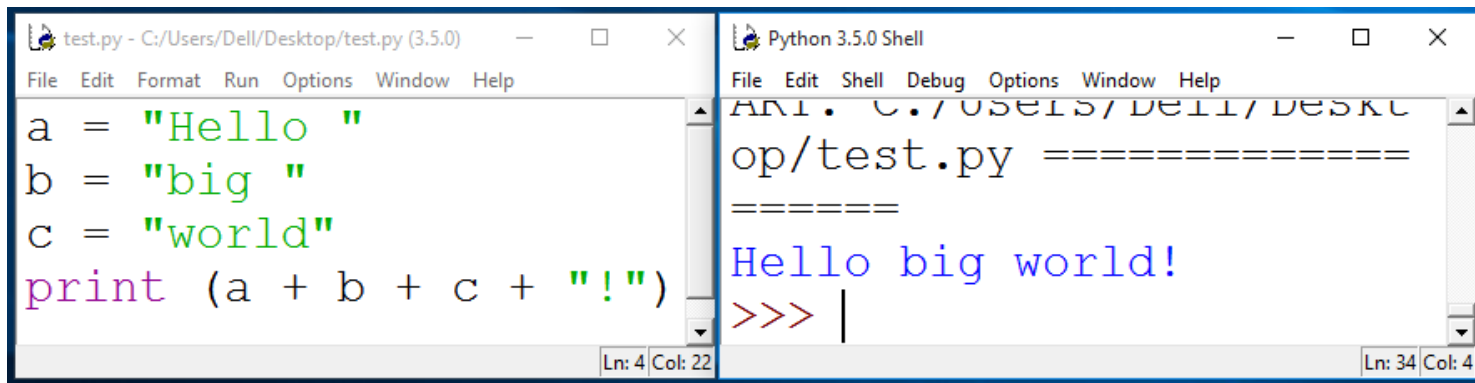
Print() function

- ▶ Used to display information on the screen

Code	Output
<pre>print("This is text")</pre>	<pre>This is text</pre>
<pre>print(10 / 3) print(2 ** 5)</pre>	<pre>3.3333333333333335 32</pre>
<pre>age = 21 print("You are", age, "years old")</pre>	<pre>You are 21 years old</pre>
<pre>age = age * 2 print("You are actually", age, "!")</pre>	<pre><u>You are actually 42 !</u></pre>

Print() function

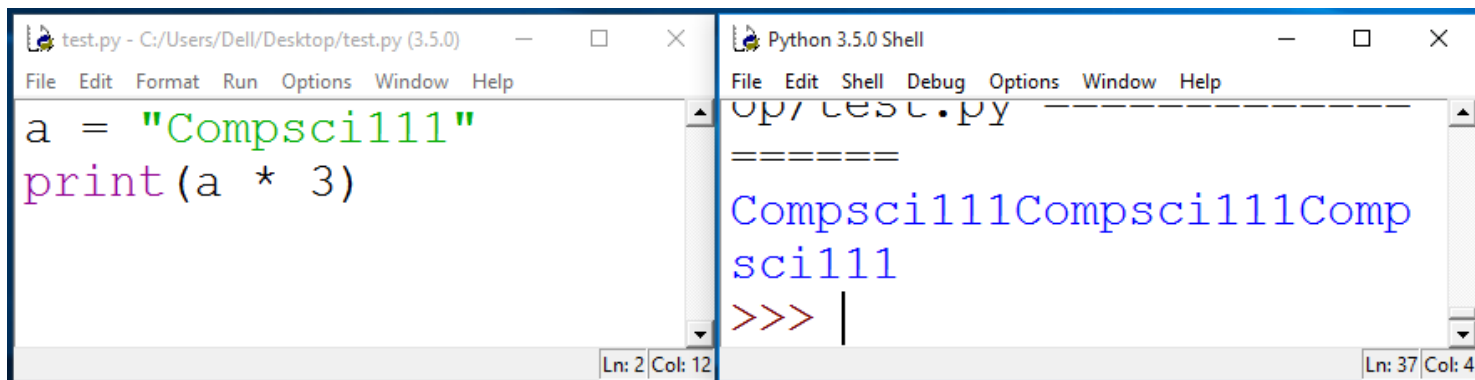
- ▶ Concatenation: this involves joining two or more strings together



```
test.py - C:/Users/Dell/Desktop/test.py (3.5.0)
File Edit Format Run Options Window Help
a = "Hello "
b = "big "
c = "world"
print (a + b + c + "!")
Ln: 4 Col: 22

Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
C:/Users/Dell/Desktop/test.py =====
=====
Hello big world!
>>> |
Ln: 34 Col: 4
```

- ▶ Repetition: lets you print a string multiple times

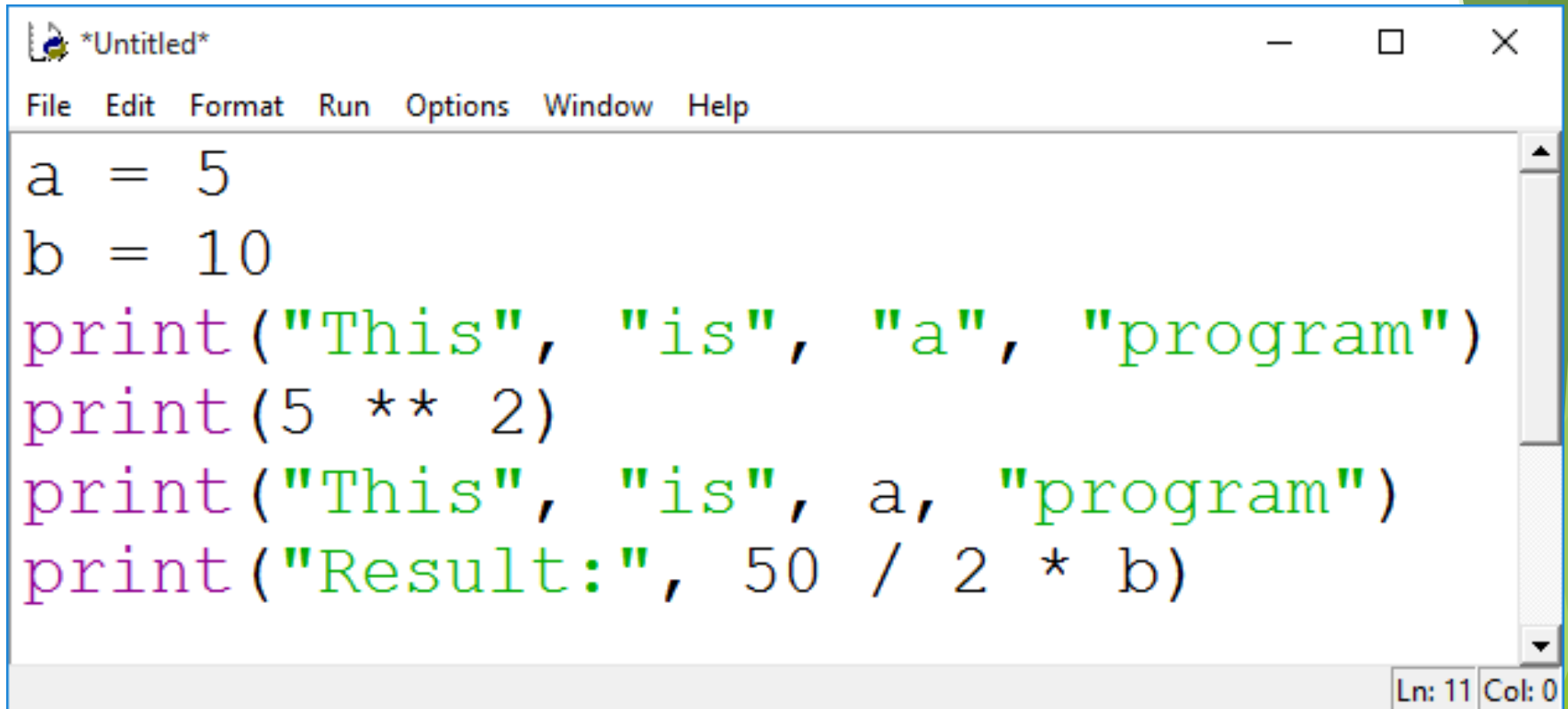


```
test.py - C:/Users/Dell/Desktop/test.py (3.5.0)
File Edit Format Run Options Window Help
a = "Compsci111"
print(a * 3)
Ln: 2 Col: 12

Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
C:/Users/Dell/Desktop/test.py -----
-----
Compsci111Compsci111Comp
sci111
>>> |
Ln: 37 Col: 4
```

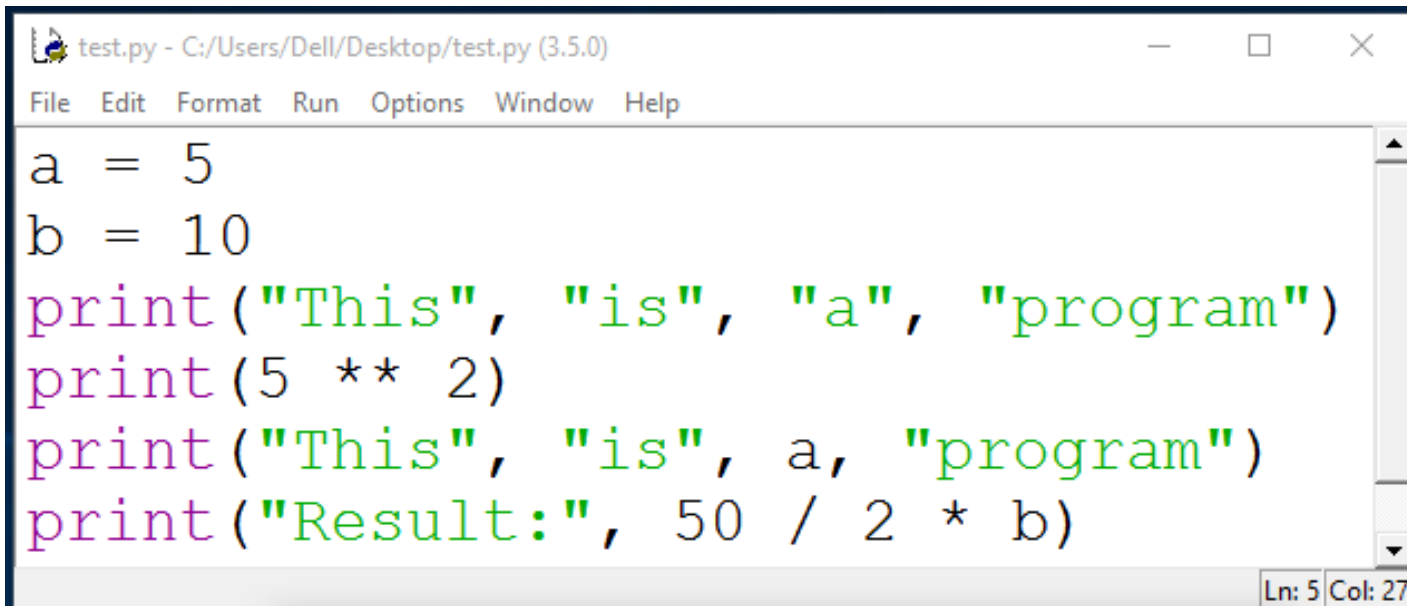
Exercise

- ▶ What is the output for the following `print()` statements:

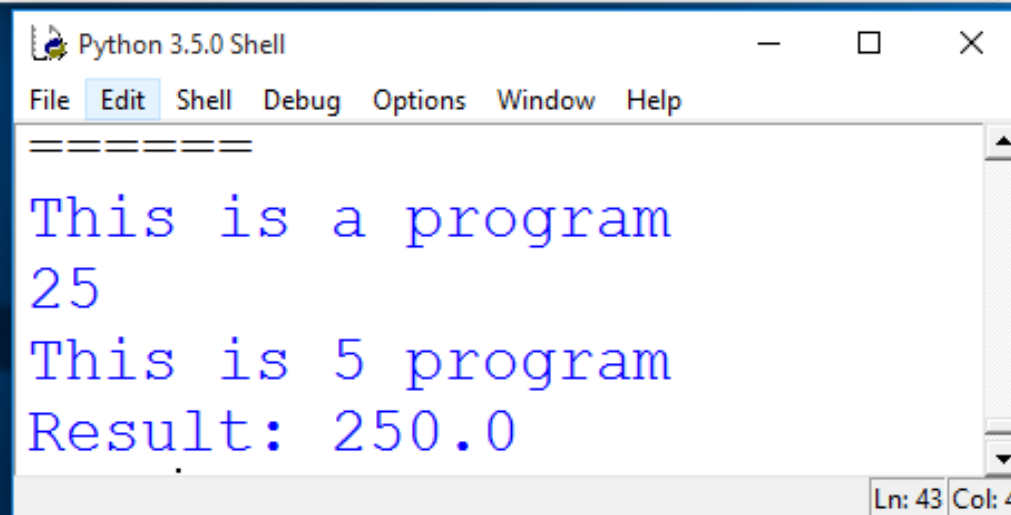


```
*Untitled*
File Edit Format Run Options Window Help
a = 5
b = 10
print("This", "is", "a", "program")
print(5 ** 2)
print("This", "is", a, "program")
print("Result:", 50 / 2 * b)
Ln: 11 Col: 0
```

Exercises



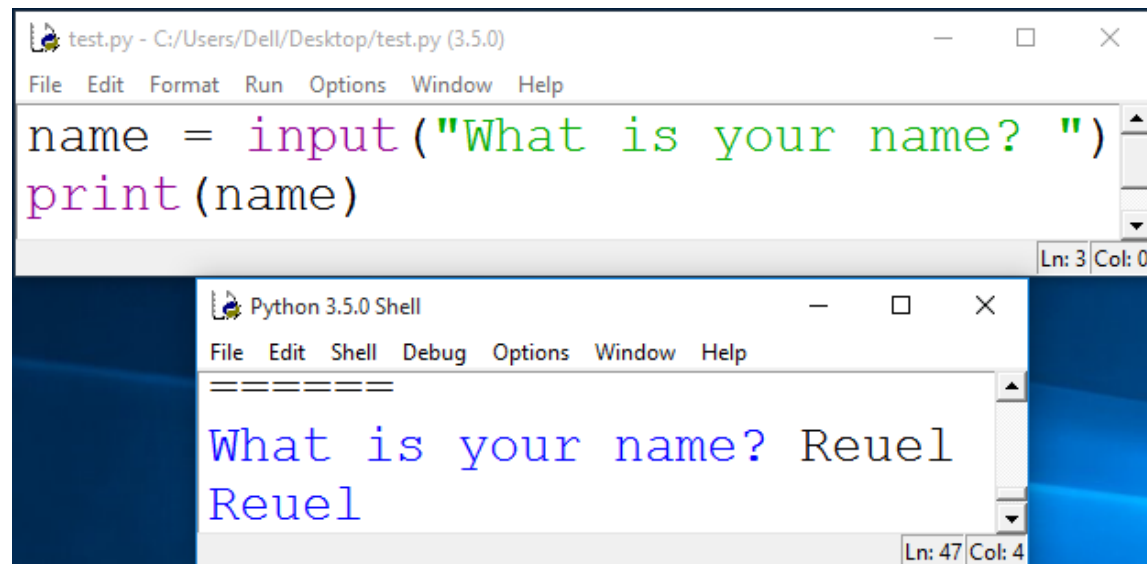
```
test.py - C:/Users/Dell/Desktop/test.py (3.5.0)
File Edit Format Run Options Window Help
a = 5
b = 10
print("This", "is", "a", "program")
print(5 ** 2)
print("This", "is", a, "program")
print("Result:", 50 / 2 * b)
Ln: 5 Col: 27
```



```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
=====
This is a program
25
This is 5 program
Result: 250.0
Ln: 43 Col: 4
```

Getting input

- ▶ Primary source of input for our programs will be the keyboard
- ▶ The `input()` function:
 - ▶ Prints a prompt for the user to read
 - ▶ Captures the user's keystrokes
 - ▶ When the user presses 'Enter', stores the string in a variable



The image shows two overlapping windows from a Python IDE. The top window, titled 'test.py - C:/Users/Dell/Desktop/test.py (3.5.0)', contains the following code:

```
name = input("What is your name? ")  
print(name)
```

The bottom window, titled 'Python 3.5.0 Shell', shows the execution of the script. It displays the prompt 'What is your name?' followed by the user's input 'Reuel' and the output 'Reuel'.

Getting input

- ▶ You convert the string value returned by `input()` to an integer or floating point value
 - ▶ You need to do this when you want the actual numerical value the user is entering
- ▶ `age = int(input("Enter your age: "))`
- ▶ `height = float(input("Enter your height: "))`
- ▶ `height = height + 1.5`

Exercise

- ▶ Write a Python program that converts feet to metres. The conversion formula is:

$$1 \text{ foot} = 0.3048 \text{ meters}$$

- ▶ Your program's output should look like this:

```
Enter feet: 34
```

```
34 feet is equal to 10.3632 metres
```

- ▶ You will need to use:
 - ▶ Variables
 - ▶ Arithmetic operator
 - ▶ `input()` and `print()`

Exercise

```
feet = int(input("Enter feet: "))  
  
feet_to_metres = 0.3048  
  
metres = feet * feet_to_metres  
  
print(feet, "feet is equal to", metres,  
      "metres.")
```

Summary

- ▶ Python programs consist of statements that are translated by an interpreter or compiler into instructions that the CPU can execute
- ▶ We've discussed the Python programming language and its features:
 - ▶ `print()`
 - ▶ Data types: `string`, `int`, `float`
 - ▶ Arithmetic operators
 - ▶ Variables and variable naming convention
 - ▶ `input()` and `int()`, `float()`