Python – Input, output & variables

Lecture 16 – COMPSCI111/111G SS 2020

Today’s lecture

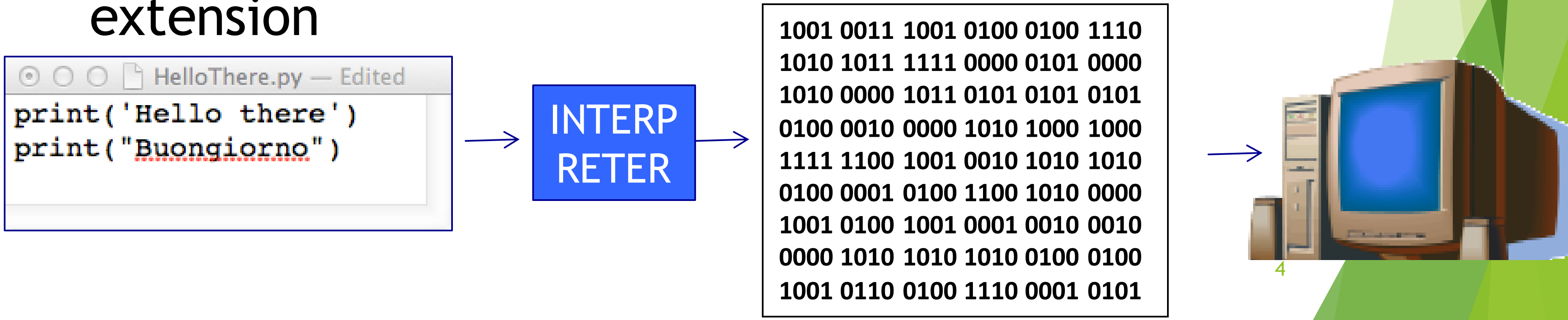
- What is Python?
- Displaying text on screen using `print()`
- Variables
- Numbers and basic arithmetic
- Getting input from keyboard using `input()`

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



```
print('Hello there')
print("Buongiorno")
```

```
1001 0011 1001 0100 0100 1110
1010 1011 1111 0000 0101 0000
1010 0000 1011 0101 0101 0101
0100 0010 0000 1010 1000 1000
1111 1100 1001 0010 1010 1010
0100 0001 0100 1100 1010 0000
1001 0100 1001 0001 0010 0010
0000 1010 1010 1010 0100 0100
1001 0110 0100 1110 0001 0101
```

Translating code

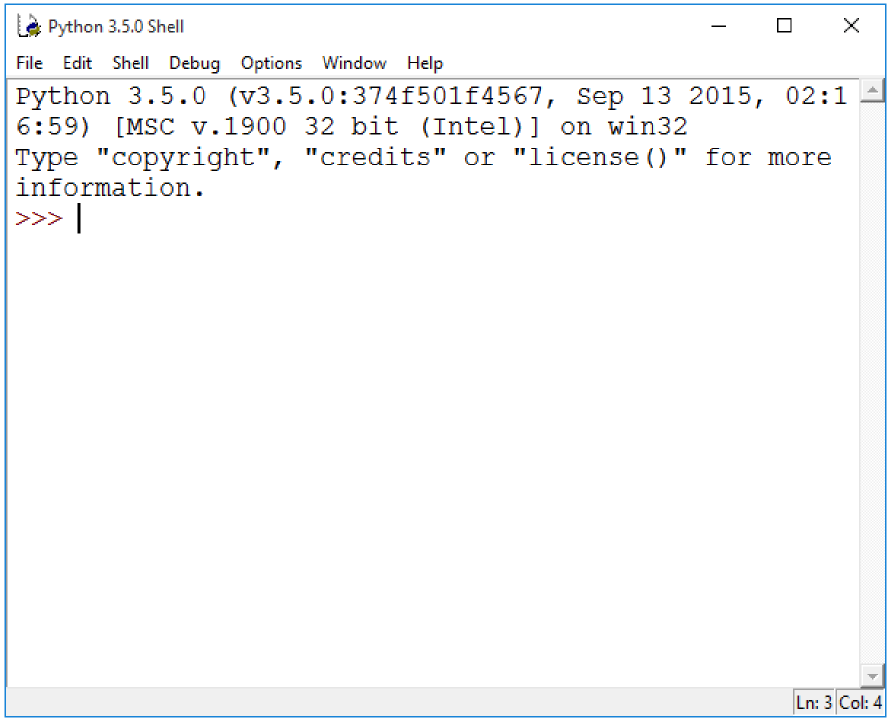
- The statements in our programs are translated into simpler instructions that the CPU can execute
- Two ways of doing this:
  - Compiler: translates the entire program file at once
  - Interpreter: repeatedly translates one line and runs it
- Python is an interpretative programming language
  - There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

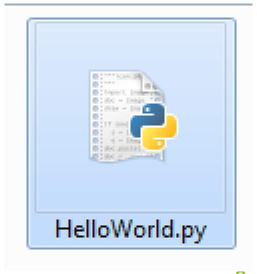
- The interpreter allows you to type statements, translate them and see them run instantly
- Very helpful for experimentation and learning



```
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:1
6:59) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more
information.
>>> 
```

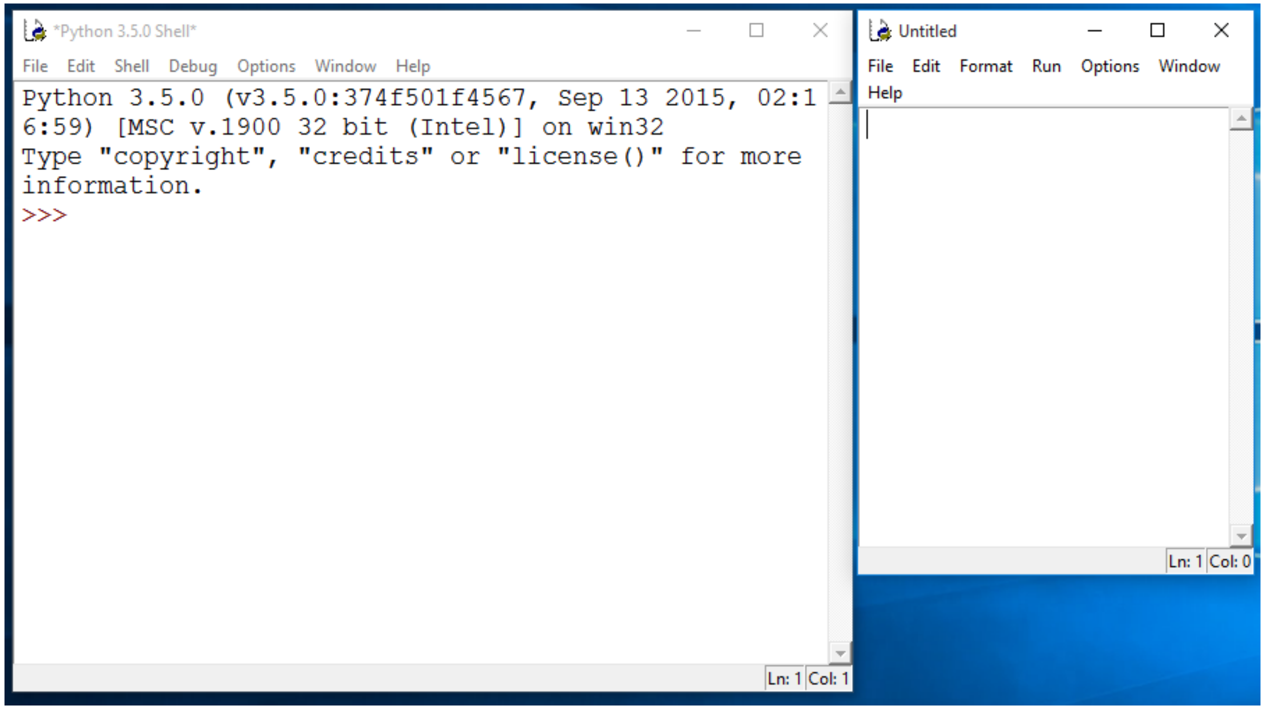
Interactive Interpreter Vs Running a script

- Interactive Interpreter
  - Allows you to type statements directly at the prompt
  - Statement is executed when you hit <Enter>
  - Very useful for experimentation
  - Good for learning
- Running a Script
  - Type a sequence of statements into a file
  - Save the file with the file extension .py
  - Running the program executes each statement in turn



IDLE IDE

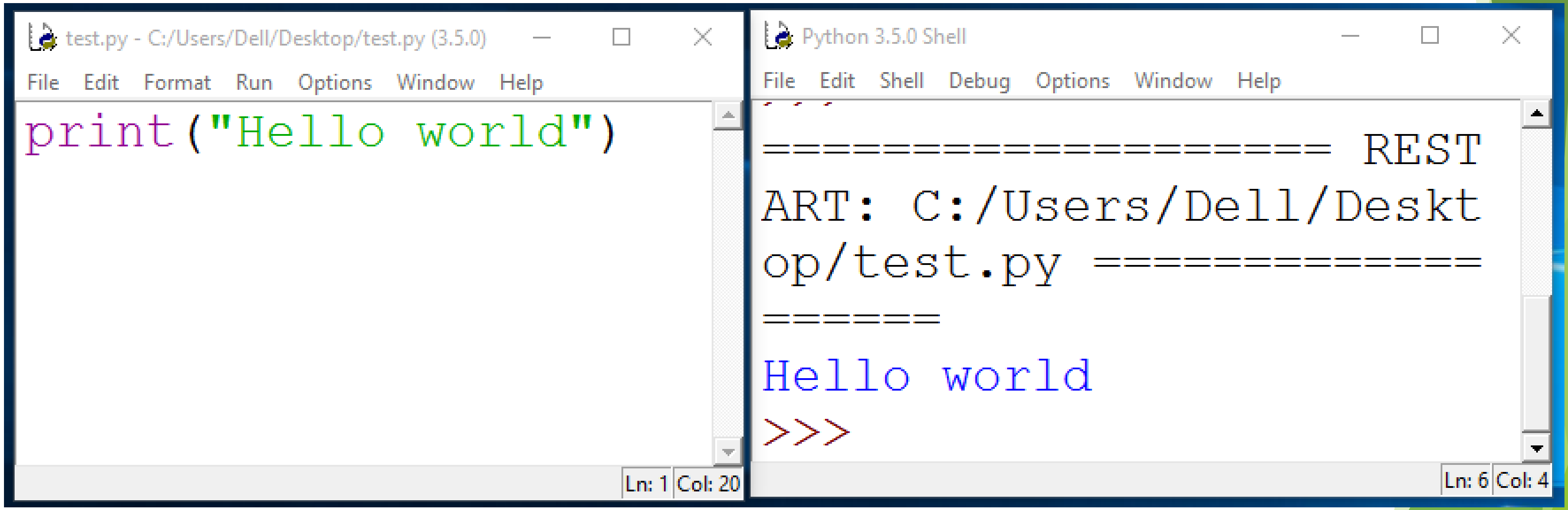
- Create a new program by clicking on File → New File
- Type your statements in the file, then click on Run → Run Module…


Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Ln: 1 Col: 1
Untitled
File Edit Format Run Options Window Help
Ln: 1 Col: 0

```
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.
>>>
```

“Hello world”

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

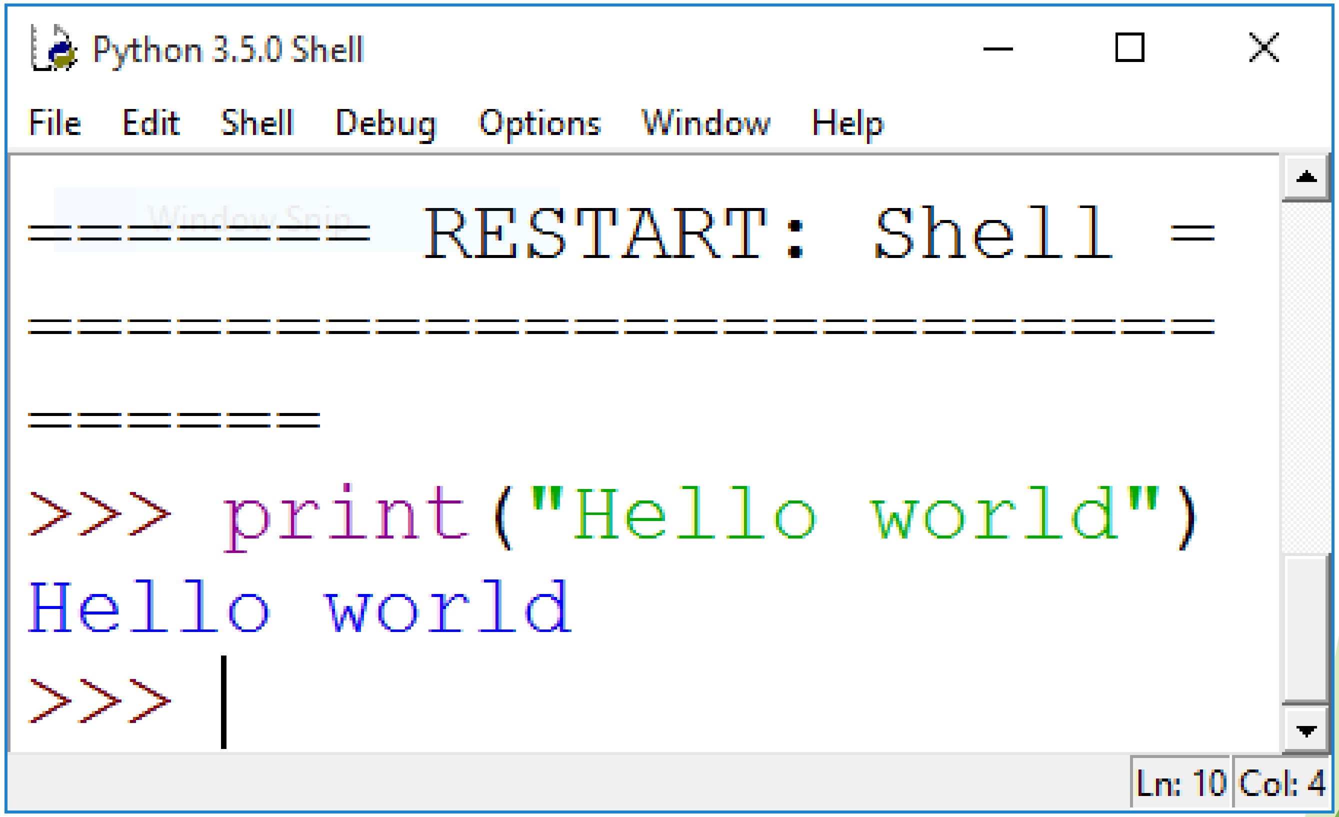

test.py - C:/Users/Dell/Desktop/test.py (3.5.0)
File Edit Format Run Options Window Help
Ln: 1 Col: 20
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Ln: 6 Col: 4

```
print("Hello world")
```

```
==================== REST
ART: C:/Users/Dell/Deskt
op/test.py ==============
======
Hello world
>>>
```

“Hello world”

- Using the Python interpreter:



```
======== RESTART: Shell =
=============================
=======
>>> print("Hello world")
Hello world
>>> 
```

Printing output

- Use the print statement

Code	Output
`print("This is text")`	`This is text`
`print(34.9)`	`34.9`

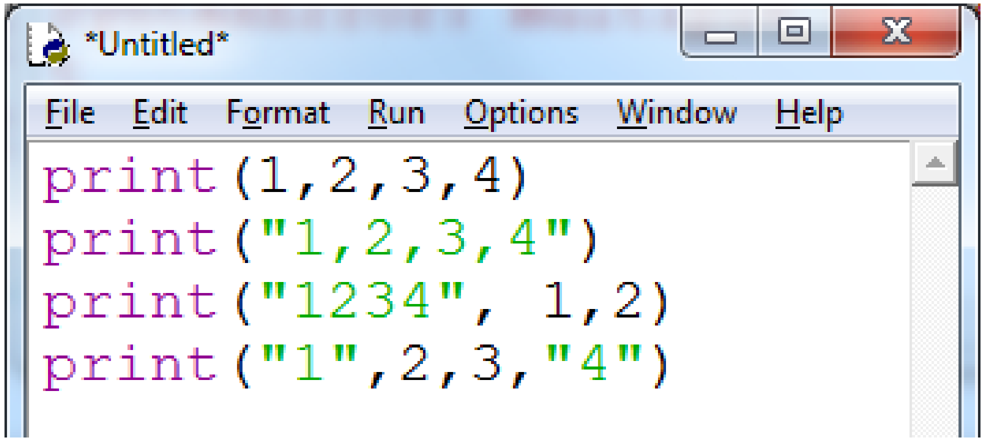
- Printing more than one thing on a single line
  - Separate each thing with a comma
  - Single space used between different things in the output

Code	Output
`print("Hello", "World")`	`Hello World`
`print("The year is", 2017)`	`The year is 2017`

Exercise 1

TRY IT OUT!

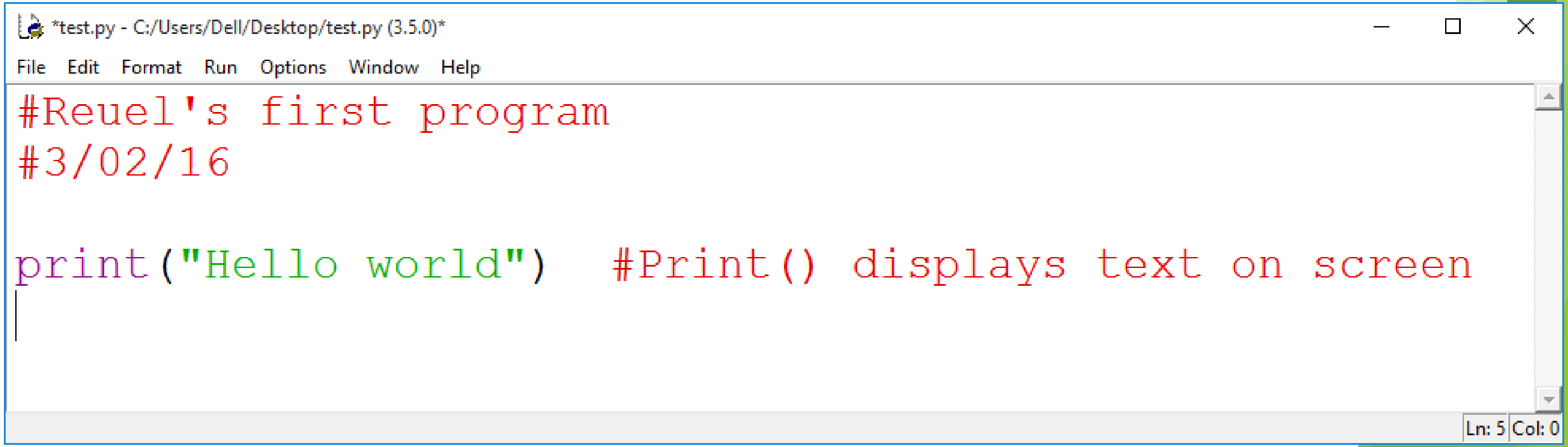
- What is the output produced by the following statements?



```
print(1,2,3,4)
print("1,2,3,4")
print("1234", 1,2)
print("1",2,3,"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 '#'


test.py - C:/Users/Dell/Desktop/test.py (3.5.0)
File Edit Format Run Options Window Help
Ln: 5 Col: 0

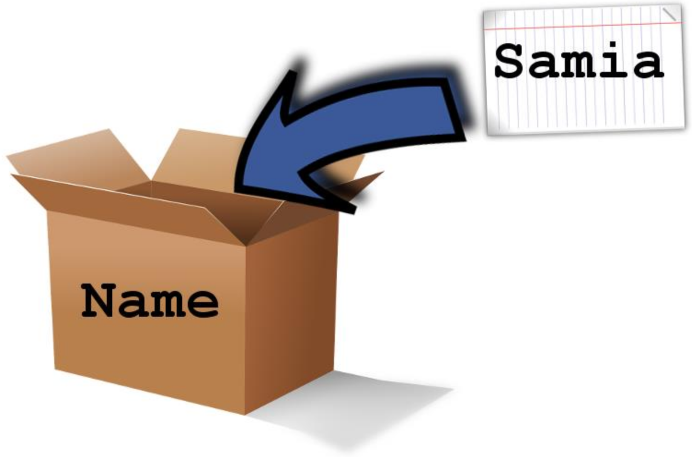
```
#Reuel's first program
#3/02/16

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

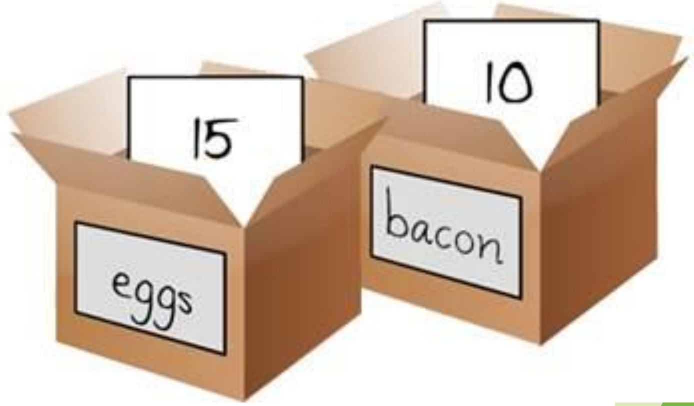
Data types

- 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

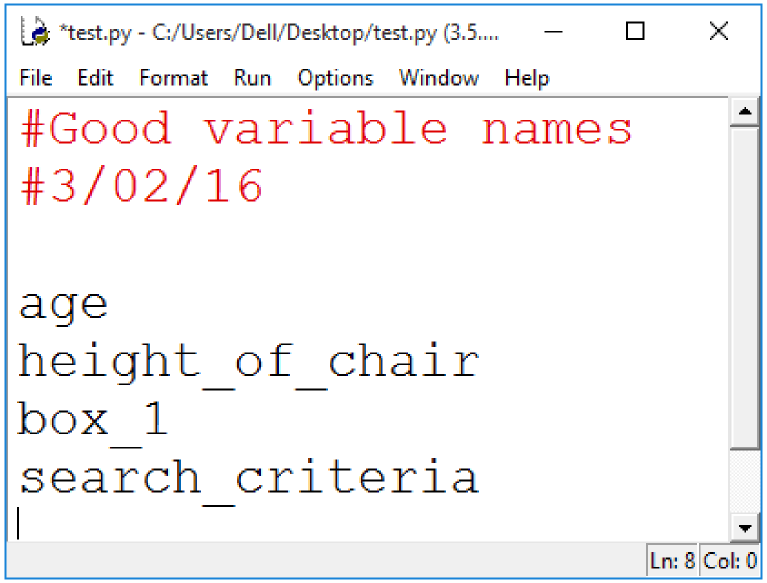
- A ‘container’ in the computer’s memory in which you can store data
- A variable’s value can change when the program runs
- Python variables are loosely-typed; they can hold any data type



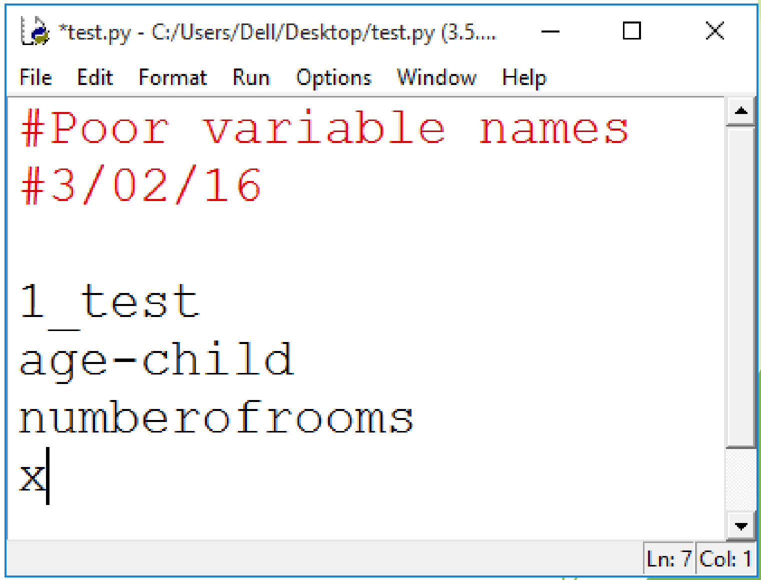


Variables

- 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
Ln: 8 Col: 0

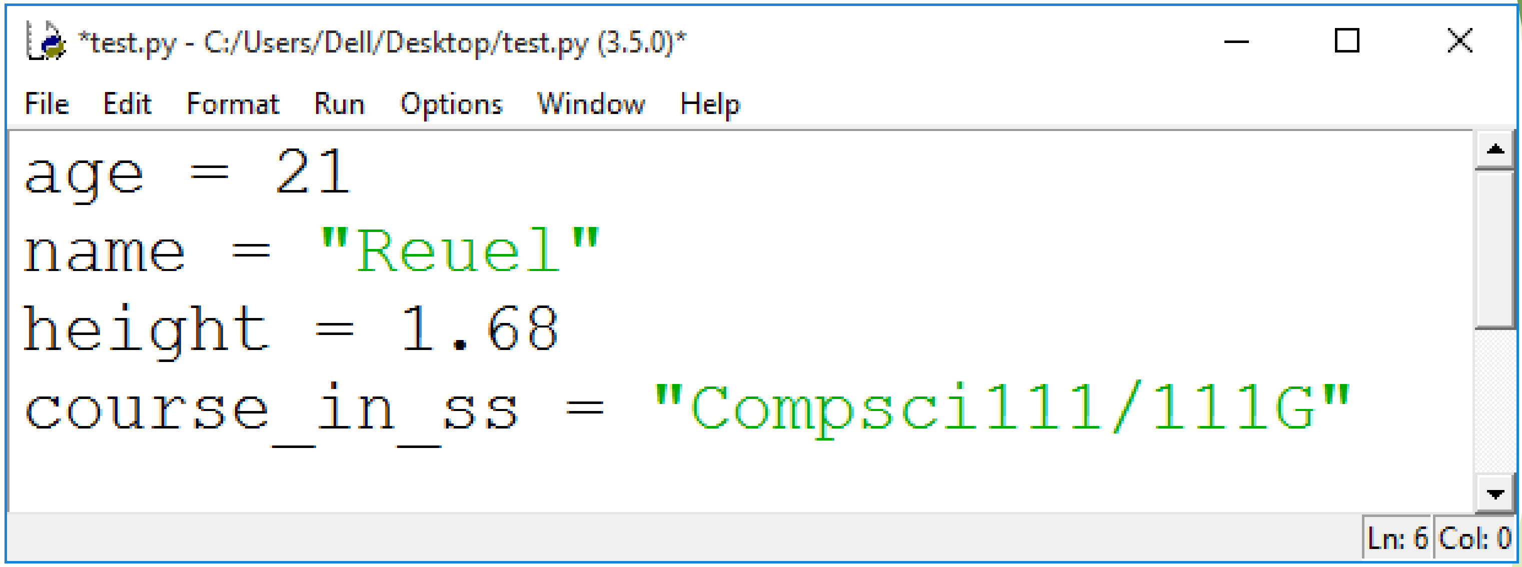
```
#Good variable names
#3/02/16

age
height_of_chair
box_1
search_criteria
```


*test.py - C:/Users/Dell/Desktop/test.py (3.5....
File Edit Format Run Options Window Help
Ln: 7 Col: 1

```
#Poor variable names
#3/02/16

1_test
age-child
numberofrooms
x
```

Assignment statement

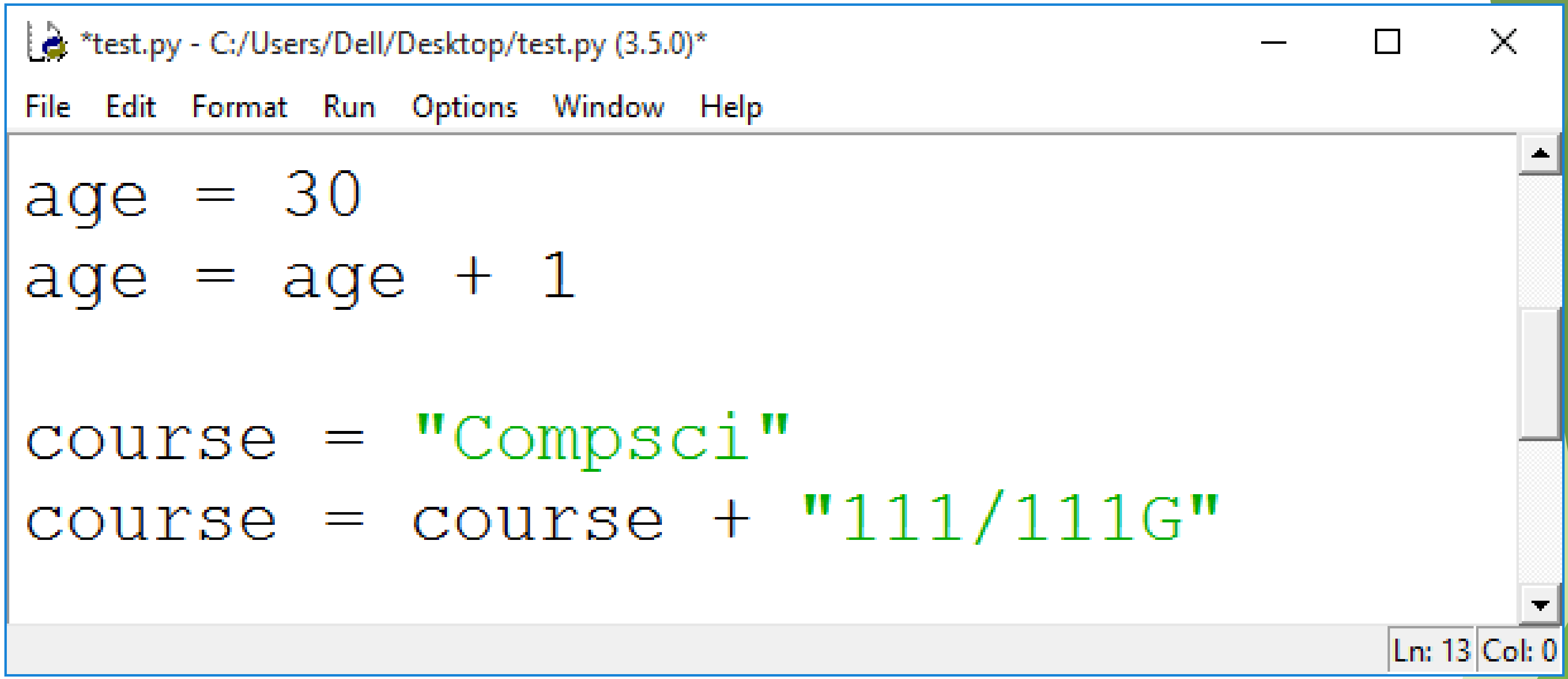
- Assigning a value to a variable:


test.py - C:/Users/Dell/Desktop/test.py (3.5.0)
File Edit Format Run Options Window Help
Ln: 6 Col: 0

```
age = 21
name = "Reuel"
height = 1.68
course_in_ss = "Compsci111/111G"
```

Assignment statement

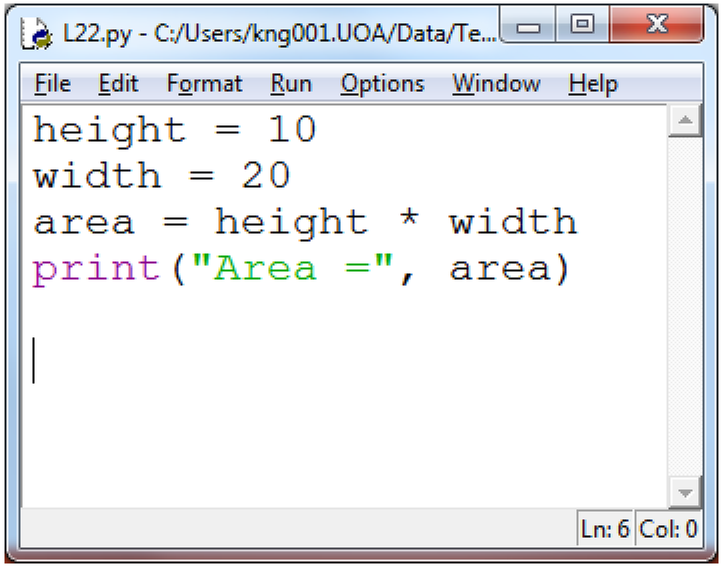
- Changing the value in a variable:


test.py - C:/Users/Dell/Desktop/test.py (3.5.0)
File Edit Format Run Options Window Help
Ln: 13 Col: 0

```
age = 30
age = age + 1

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

Exercise 2

- What is the output produced by the following statements?


L22.py - C:/Users/kng001.UOA/Data/Te...
File Edit Format Run Options Window Help
Ln: 6 Col: 0

```
height = 10
width = 20
area = height * width
print("Area =", area)
```

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
`print("This is text")`	`This is text`

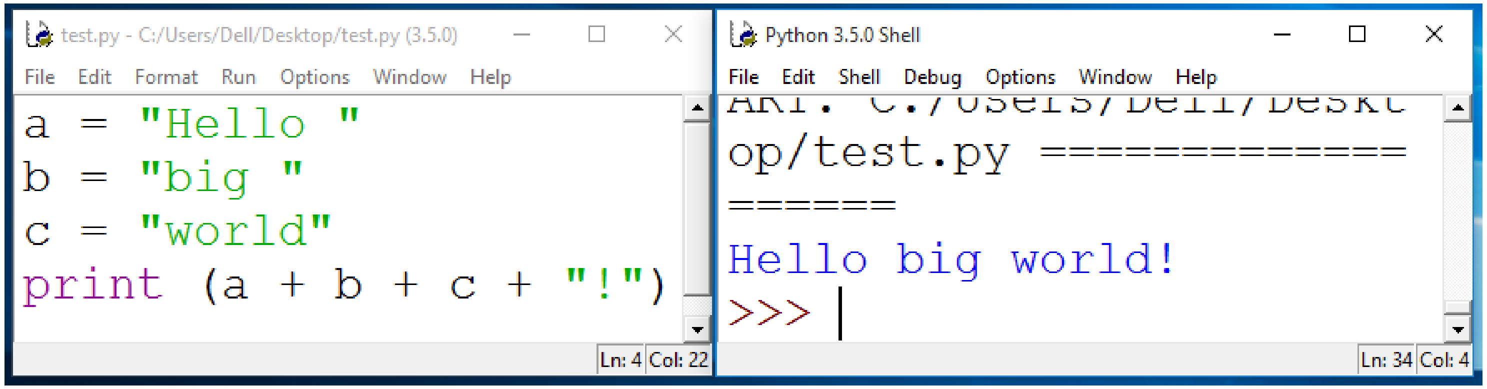
`print(10 / 3)`
`print(2 ** 5)`
	`3.3333333333333335`
	`32`

`age = 21`
`print("You are", age, "years old")`
	`You are 21 years old`

`age = age * 2`
`print("You are actually", age, "!")`
	`You are actually 42 !`

Print() function

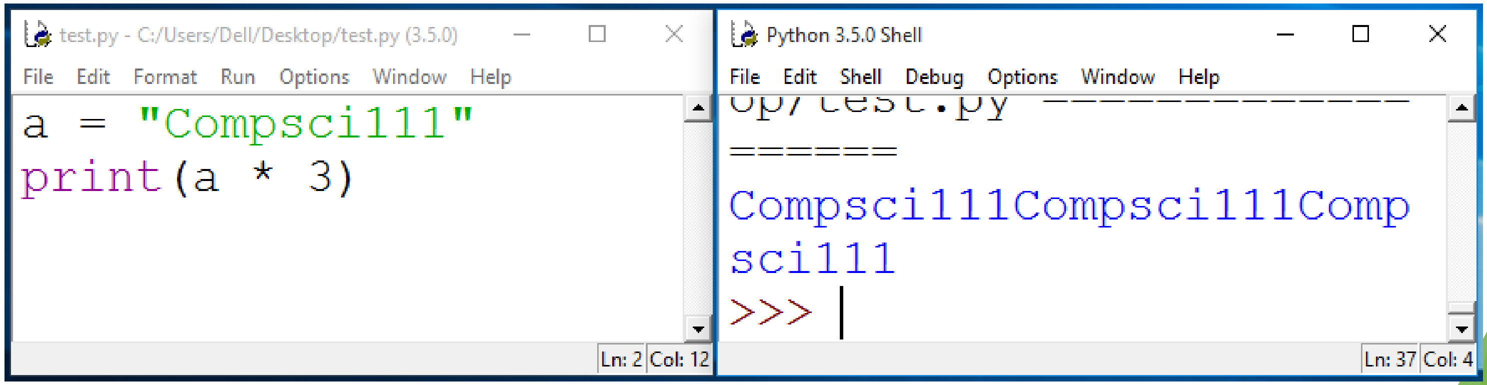
- Concatenation: this involves joining two or more strings together


test.py - C:/Users/Dell/Desktop/test.py (3.5.0)
File Edit Format Run Options Window Help
Ln: 4 Col: 22
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Ln: 34 Col: 4

```
a = "Hello "
b = "big "
c = "world"
print (a + b + c + "!")
```

```
op/test.py ==============
=======
Hello big world!
>>> 
```

- Repetition: lets you print a string multiple times

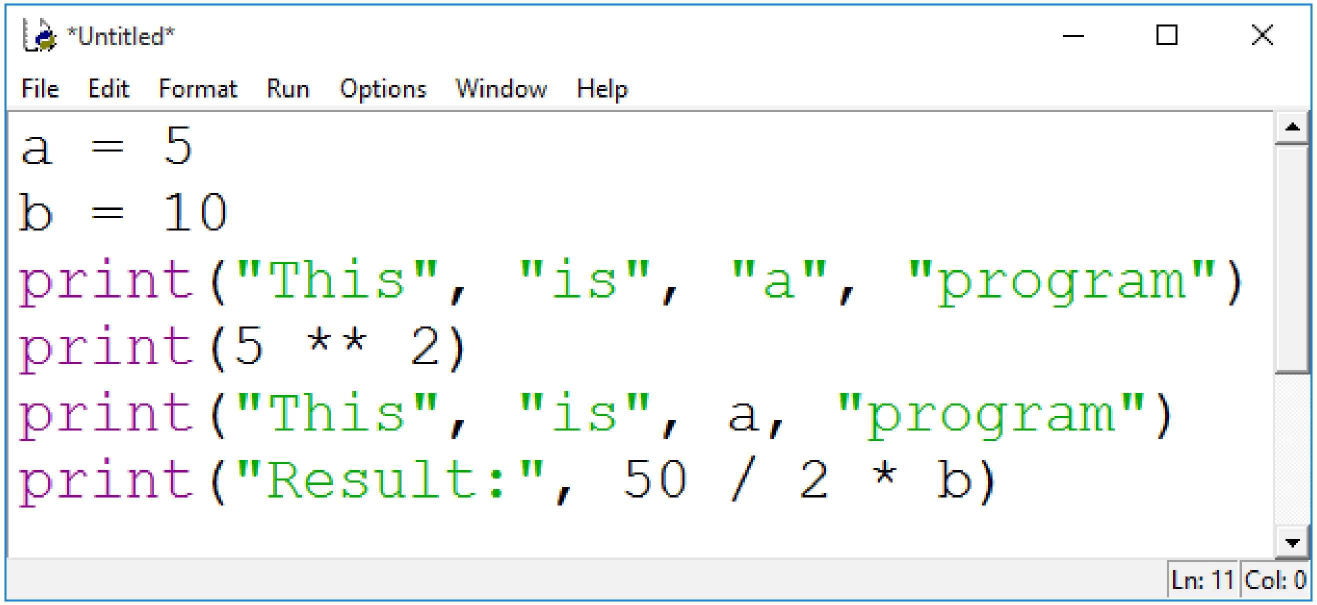

test.py - C:/Users/Dell/Desktop/test.py (3.5.0)
File Edit Format Run Options Window Help
Ln: 2 Col: 12
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Ln: 37 Col: 4

```
a = "Compsci111"
print(a * 3)
```

```
=======
Compsci111Compsci111Comp
sci111
>>> 
```

Exercise 3

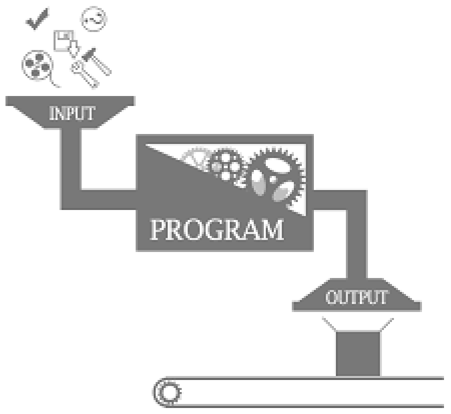
- What is the output for the following code?

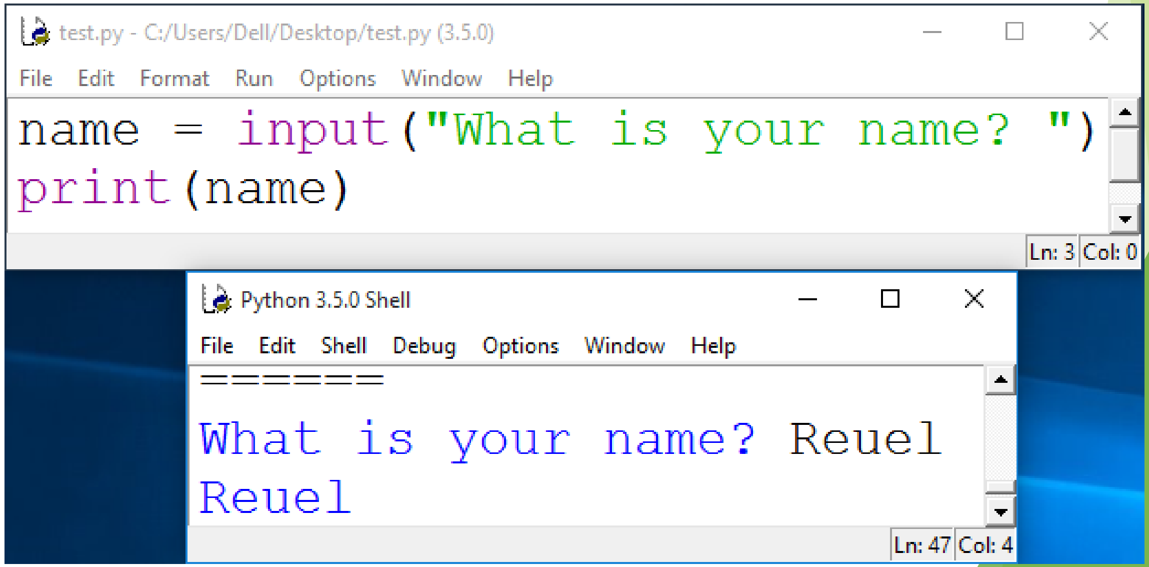


```
a = 5
b = 10
print("This", "is", "a", "program")
print(5 ** 2)
print("This", "is", a, "program")
print("Result:", 50 / 2 * b)
```

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




test.py - C:/Users/Dell/Desktop/test.py (3.5.0)
File Edit Format Run Options Window Help
Ln: 3 Col: 0
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Ln: 47 Col: 4

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

```
=======
What is your name? Reuel
Reuel
```

Getting input

- Converting 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 4

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

  1 foot = 0.3048 metres

- Your program's output should look like this:

```
Enter feet: 34
34 feet = 10.3632 metres.
```

- You will need to use:
  - Variables
  - Arithmetic operator
  - `input()` and `print()`
- You can try this online at:
  https://trinket.io/features/python3

Algorithm

Prompt for the value

Create a variable and set the value (`feet_to_metres =` 0.3048)

Calculate the corresponding value

print the result

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 conventions
  - `input()` and `int()`, `float()`