ISYS90088 Introduction to Application Development

Department of Computing and Information Systems
University of Melbourne

Semester 2 , 2016

Dr. Thomas Christy

Slides by Antonette Mendozaa, with some slides adapted from Fundamentals of Python by Kenneth A. Lambert and dept. resources

Lecture Agenda

- Introduction
  - Objectives
  - Staff ; Learning & Assessment
- What is a computer, and how do we talk to it?
- Python
- Grok
- Python Basics - print, input, strings

About me (Thomas)

- PhD in Computer Science – Bangor University, Wales
  - Visualization Software Engineer
  - Research Software Engineer
  - AI Software Engineer
  - Android App development

- Email: thomas.christy@unimelb.edu.au
- Office: 5.13, Doug McDonell (Building 168)

About me (Thomas)

- Research focus:
  - Expertise lies in the area of human computer interaction utilizing psychophysiology.
  - Interests extend to visualisation, Virtual and Augmented reality, Boids and AI, video game development.

Setting Expectations

What is Programming?

What is a Software development life cycle?

Is it important?

What do you expect to learn from this subject?

What we will be studying

- Fundamental concepts and models of application development
- Students who have no background in application development or programming.
- Learn about the software development lifecycle, program design, data structures, problem solving, programming logic, implementation considerations, testing and enterprise level applications.

Objectives

- Learn to use primitive data types and data structures
- Understand basic programming concepts
- Write simple applications that relate to a specific domain
- Design, implement, test, and debug a program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and the definition of functions.
- Test applications

Learning and Assessment

- Lectures, Labs
- Assessment
- Textbook & references

Learning and Assessment

Lectures: Wednesday 10am – 12pm in Chemistry-189 (Masson Theatre)

Labs: 7 scheduled labs across the week – attend any one!

Monday	18:15	20:15	Alan Gilbert-111 (Computer based Instructional Environment)
Tuesday	18:15	20:15	Alan Gilbert-111 (Computer based Instructional Environment)
Wednesday x 2	17:15	21:15	Alan Gilbert-111 (Computer based Instructional Environment)
Wednesday	19:15	21:15	Alan Gilbert-111 (Computer based Instructional Environment)
Friday x 2	17:15	21:15	Alan Gilbert-111 (Computer based Instructional Environment)

Tutors: Yang, Ninad, Curtis, Nicholas, Juanna, and Mariam.

Learning and Assessment

Assessment:

- Individual Assignment 1 (10%) due in Week 6
- One mid-semester test (10%) in Week 7
- Individual Assignment 2 (20%) due in Week 12
- Exam (60%) – end of semester (3hr and will be a hurdle)

Learning and Assessment

Text books/references:

- You may use any text book and references.
- Some books/references:
  - Fundamentals of Python – Kenneth A. Lambert
  - There are good online resources (free)
  - Check out: *www.python.org*

LMS

- University's "Learning Management System"
  - http://www.lms.unimelb.edu.au/
  - Use your university email ID and password

A word about the LMS

We will post all code from lectures (other than snippets from the \console) on the LMS after each lecture

- It is a good idea to look back over the code to ensure you fully understand it and play around with it yourself

Academic Honesty

- In accordance with the University's Academic Honest and Plagiarism Policy (which you should familiarize yourself with!):

  https://academichonesty.unimelb.edu.au/

- All examinable work (Grok worksheet answers and all project work) that you submit for this subject must be your own!!

Academic Honesty

- Common causes of breaches in the past have been:
  - ✓ Friends asking to look over your code to "get hints" for their own project
  - ✓ Flatmates accessing your code via a shared desktop computer with saved login details
  - ✓ Study groups where the facilitator has overstepped the line and provided sample code to help people along

Academic Honesty

- Common attempts to escape undetected are:
  - ✓ changing the comments but not the code
  - ✓ changing variable names
  - ✓ rearranging blocks of code (sometimes breaking the logic in the process!)
- It is all too easy to automatically pick up on all of these, and many, many more, approaches using software plagiarism detection software … and we **do** check

So what is appropriate?

- You are encouraged to share/collaborate directly on code for any non-examinable items (notably the worksheets questions) … and you will learn a lot from reading the code of others (including the sample solutions in the worksheets)
- You are very welcome to discuss with fellow classmates your *approach* to worksheet questions and the projects, in conceptual terms, or in terms of key data types or programming constructs used (just **not** with the aid of raw code)

Break!!!!

Lecture Objectives

After completing this lecture, you will be able to:

- Describe the basic features of an algorithm
- Explain how hardware and software collaborate in a computer's architecture
- Compose and run a simple Python program

Algorithms & Information Processing

- Computer science focuses on
  - **Algorithms**
  - **Information processing**

Algorithm – what is it?

- Sequence of steps that describes each of these computational processes is called an **algorithm**
- Features of an algorithm:
  - Consists of a finite number of instructions
  - Each individual instruction is well defined
  - Describes a process that eventually halts after arriving at a solution to a problem
  - Solves a general class of problems

Algorithm – example!

- Steps for making a cup of coffee !!!

- Imagine that you want a robot (or a friend) to make it.
- How will you explain this procedure to the robot (or your friend) so they can make it?

Algorithm – another example!

- Steps for subtracting two numbers:
  - **Step 1:** Write down the numbers, with larger number above smaller one, digits column-aligned from right
  - **Step 2:** Start with rightmost column of digits and work your way left through the various columns
  - **Step 3:** Write down difference between the digits in the current column of digits, borrowing a 1 from the top number's next column to the left if necessary
  - **Step 4:** If there is no next column to the left, stop
    - Otherwise, move to column to the left; go to Step 3
- The **computing agent** is a human being

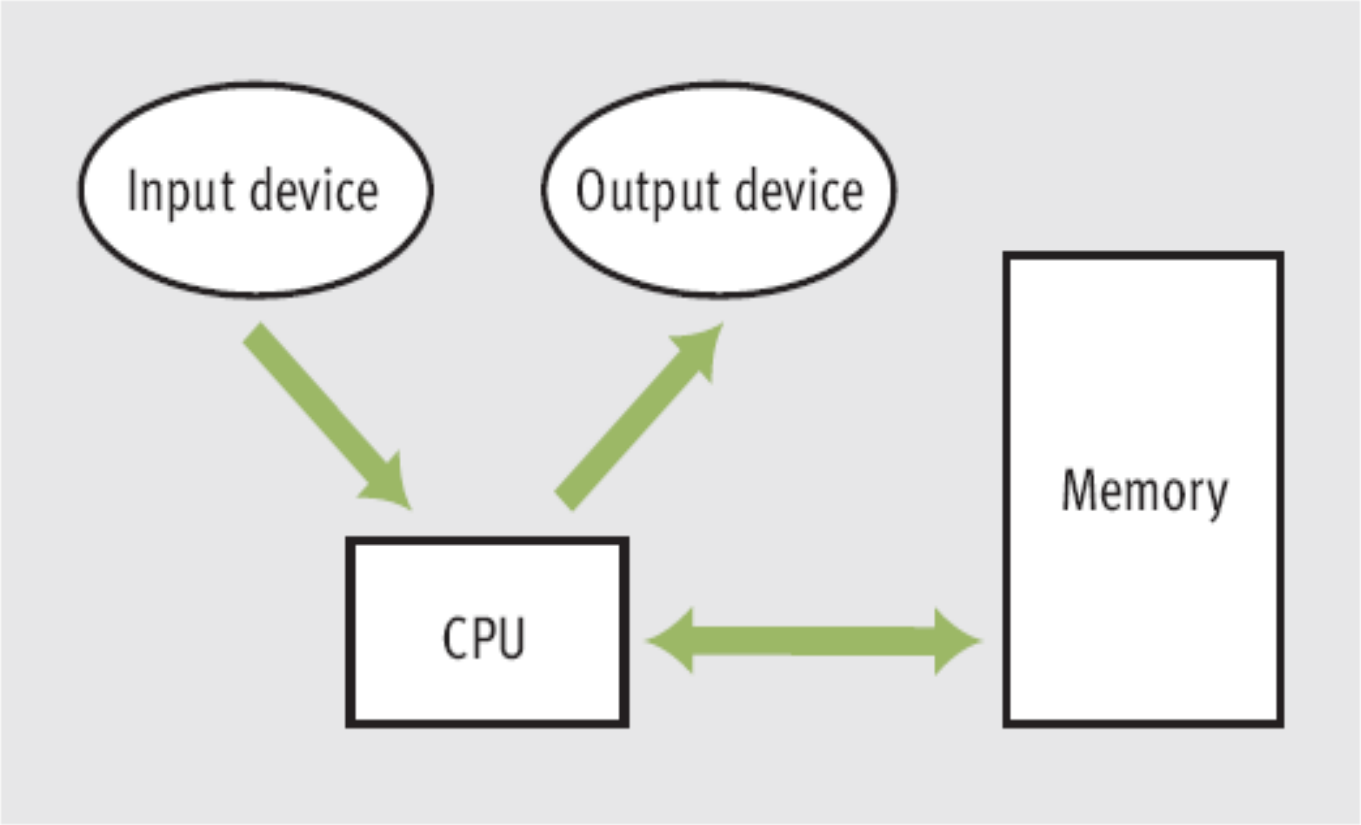
Information Processing

- Information is also commonly referred to as **data**
- Data needs to be organized and stored in some form to be relevant and easily accessed!
- In carrying out the instructions of an algorithm, computing agent manipulates information
  - Start with **input → do something/task →** produce **output**

Structure of a Modern Computer System

- A modern computer system consists of **hardware** and **software**
  - Hardware: physical devices required to execute algorithms
  - Software: set of these algorithms, represented as **programs** in particular **programming languages**

Computer Hardware



Computer Hardware (continued)

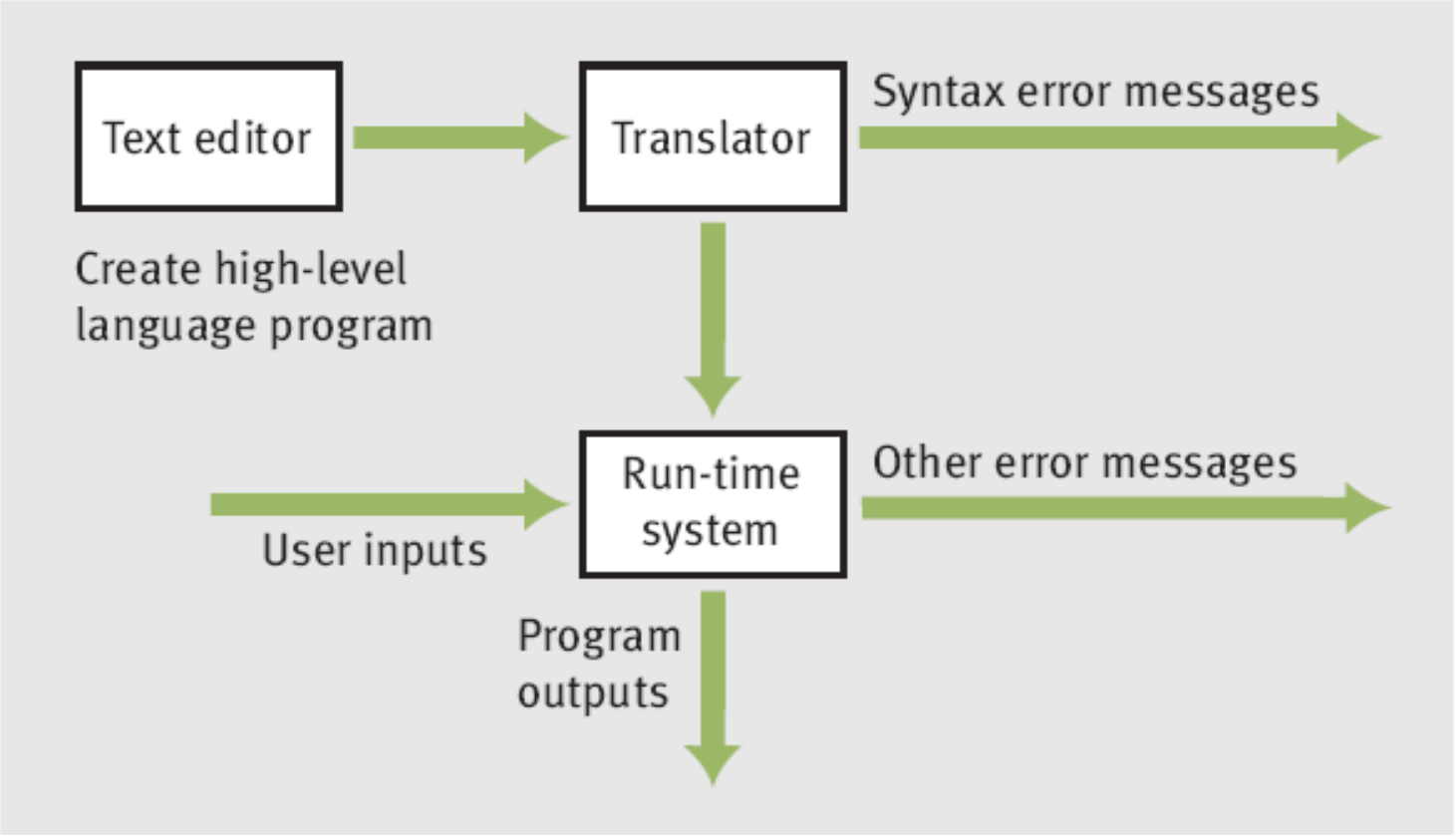
Cell 7	1	1	0	1	1	1	1	0	1	1	1	1	1	1	0	1
Cell 6	1	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1
Cell 5	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1	1
Cell 4	1	0	1	1	1	0	1	1	1	1	1	1	0	1	1	1
Cell 3	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1
Cell 2	0	0	1	1	1	1	0	1	1	1	0	1	1	1	0	1
Cell 1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1
Cell 0	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	0

- **Random access memory (RAM)** is also called **internal** or primary
- **External** or secondary memory example: cd; floppies magnetic disks; optical disks!!!

Computer Software

- A program stored in computer memory must be represented in binary digits, or **machine code**
- A **loader** takes a set of machine language instructions as input and loads them into the appropriate memory locations
- The most important example of **system software** is a computer's **operating system**
  - Some important parts: **file system**, **user interfaces (terminal-based** or **GUIs)**
- **Applications** include Web browsers, games, etc.

Computer Software (continued)



Getting Started with Python Programming

- Early 1990s: Guido van Rossum
  - invented the Python programming language

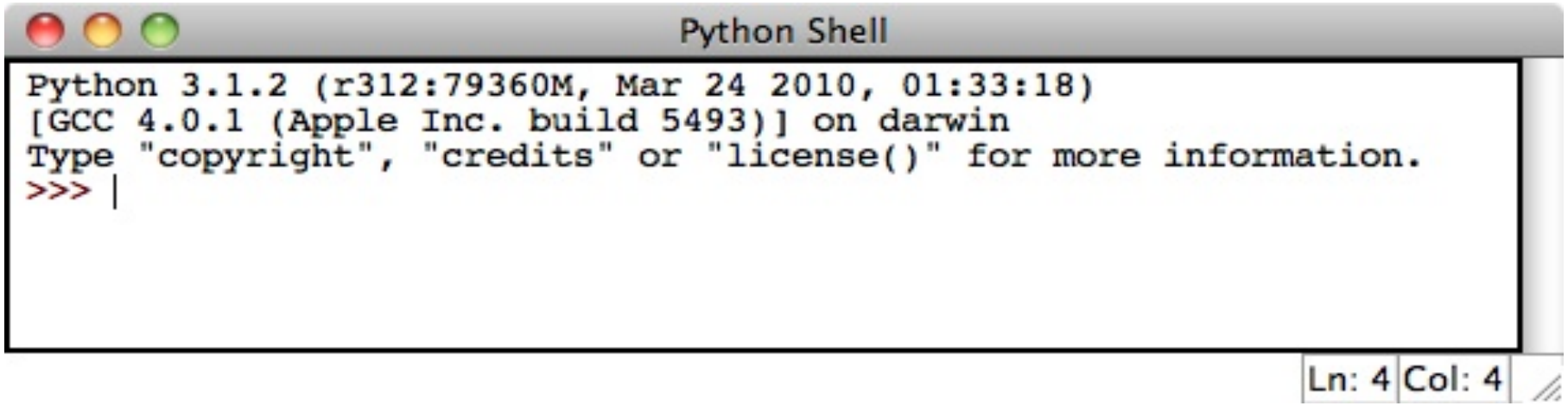
- **Python** is a high-level, general-purpose programming language for solving problems on modern computer systems

- Useful resources at *www.python.org*

Running Code in the Interactive Shell

- Python is an **interpreted** language
- Simple Python expressions and statements can be run in the **shell**
  - Easiest way to open a Python shell is to launch the IDLE (**I**ntegrated **D**eve**L**opment **E**nvironment)
  - To quit, select the window's close box
  - Shell or command line is useful for:
    - Experimenting with short expressions or statements
    - Consulting the documentation

Running Code in the Interactive Shell (continued)



Input, Print, String Processing & Output

- Programs usually accept inputs from a source, process them, and output results to a destination
  - In terminal-based interactive programs, these are the keyboard and terminal display

The **print** statement

A **print** statement evaluates the expression and displays them, separated by one space, in the console window

- **Syntax:**

  print (<expression>, …, <expression>)

  Example:

  >>>print (3)

  3

- If you want to display a text (in python called a string), then the string must be enclosed within quotations – single or double

strings in python

- The word **string** is used to describe a piece of text inside quotes.
- You've seen how Python can be used to manipulate numbers, but it is also useful to be able to manipulate text.
- The things you type into a Python program are expected to be Python commands.

  Example:

  >>>print (“hi there”)

  >>>print (‘hi there’)

  >>>print (“what’s she doing”)

Lets run this and see what happens!!!!!

Strings in python

- Suppose you want to represent the words **Hello There** as text inside a Python program. If you type them as is into the program directly, Python will try to treat those words as names in the Python language:

```
>>> print (hello there)
```

What happens? Lets check it out!

We get an error message!!!!!

Errors and python translator

- Sometimes, you will write something which the Python interpreter does not understand. The most basic kind of mistake you can make is a **syntax error**.
- A syntax error occurs when you type something which is not properly formed, according to the rules of the programming language.
- It is the same as a grammatical error in English (or any natural language for that matter).
- So, in this example it did not know what hello there is (not an identifier) until you tell it to be a string within quotes.

```
>>> print ('hello there')
hello there
```

print - examples

```
print(<expression>)
```

```
>>> print('Hi there')
Hi there
```

```
print(<expression>, … , <expression>)
```

- You can type in expressions and it will calculate the answer for you (noting that the asterisk, or *, signifies multiplication):

```
>>>print (3 * 2)
6
```

The **input** statement

- Python has a built-in function called **input** that can be used to get keyboard input from the user as a string.
- The **input** function can be given a message to display, usually prompting the user with what kind of information the program wants.
- A **variable** is a name for a value. When a variable receives a value, that value can be accessed using the variable name when required.

The Input statement - example

```
>>> name = input("Enter your name: ")
Enter your name: Ken Lambert
>>> name
'Ken Lambert'
>>> print(name)
Ken Lambert
>>>
```

```
<variable identifier> = input(<a string prompt>)
```

```
>>> name
'Ken Lambert'
```

```
>>> first = int(input("Enter the first number: "))
Enter the first number: 23
>>> second = int(input("Enter the second number: "))
Enter the second number: 44
>>> print("The sum is", first + second)
The sum is 67
>>>
```

Editing, Saving, and Running a Script

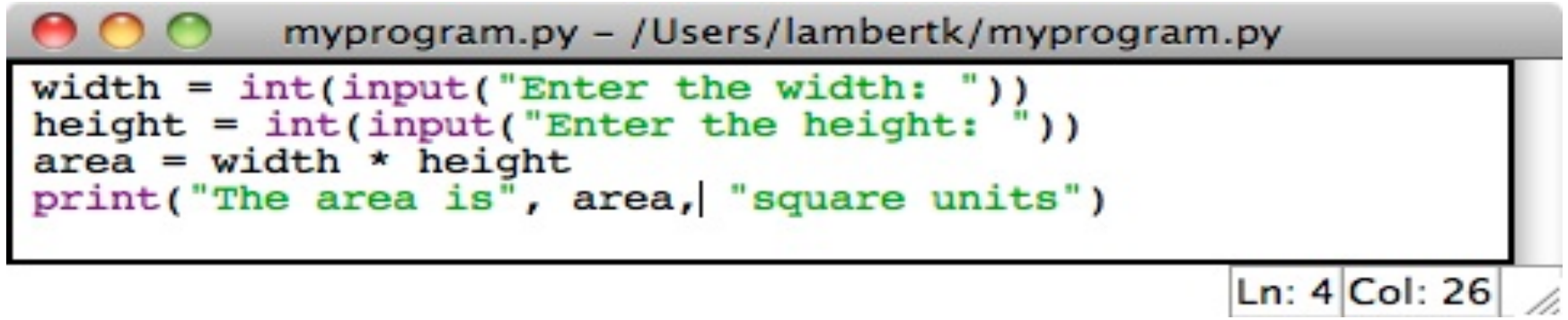
- We can then run Python program files or **scripts** within IDLE or from the OS's command prompt
- Python program files use .py extension
- Running a script from IDLE allows you to construct some complex programs, test them, and save them in **program libraries** to reuse or share with others

Editing, Saving, and Running a Script

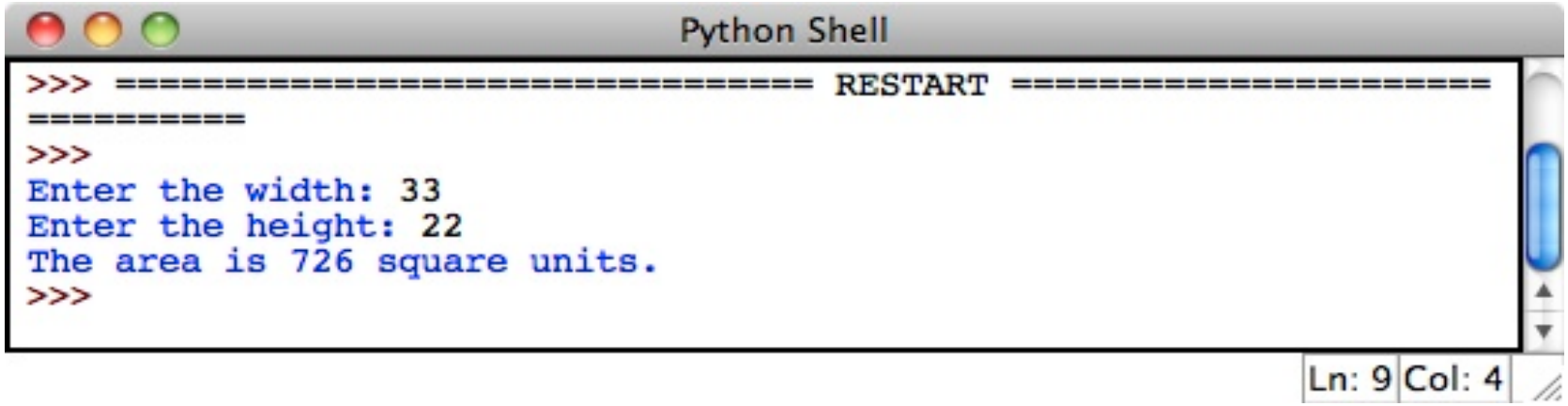
- Select New File from the File menu
- enter the python code/statements
- File/Save as (give a sensible name to your file. The extension will be.py)
- To run the file or code as a python script, select Run module from the Run Menu

Examples using IDLE – shown in lecture

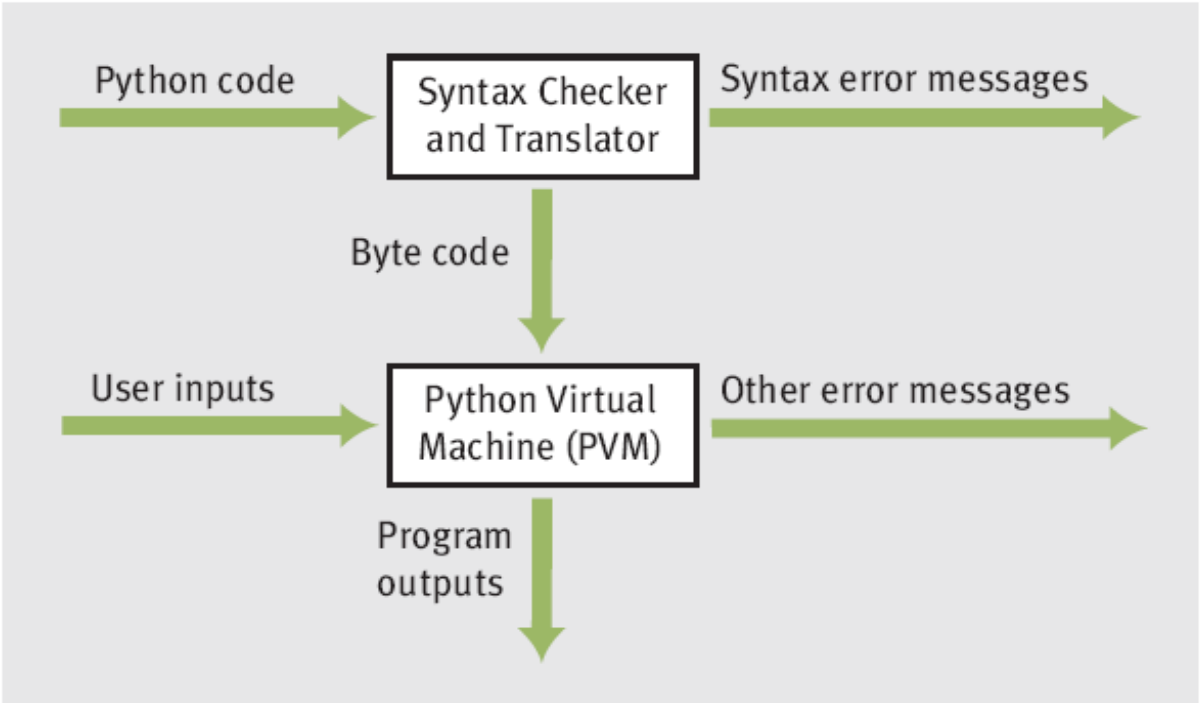
Editing, Saving, and Running a Script (continued)



Editing, Saving, and Running a Script (continued)



Behind the Scenes: How Python Works



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

Detecting and Correcting Syntax Errors (continued)

```
>>> length = int(input("Enter the length: "))
Enter the length: 44

>>> print(lenth)
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
NameError: name 'lenth' is not defined
```

```
>>>  print length
  File "<pyshell#1>", line 1
    print length
    ^
SyntaxError: unexpected indent
```

```
>>> 3 +
    3 +
SyntaxError: invalid syntax
```

Install Python

- We will use Python v3.4 via GROK (on my machine 3.5.1)
- You just write it like a text and the python interpreter turns it into machine code for you
- Get a copy of python for your own machine at home - there are free versions for Windows, MacOS and Linux

  http://www.python.org/download/
- Portable version (USB) http://portablepython.com/
- Advanced Python Distribution (for scientific experimentation)

  http://www.enthought.com/products/edudownload.php

Grok Learning environment

- Grok Learning is the web-based programming environment we will be using for the duration of this subject in your labs:

  https://groklearning.com/course/unimelb-isys90088-2016-s2/
- All you need to access the system is a browser, an internet connection and your Grok account
- Different modes of working in Grok: code, run, mark, terminal

Summary

- Fundamental ideas of computer science
  - The algorithm
  - Information processing
- Real computing agents can be constructed out of hardware devices
  - CPU, memory, and input and output devices

Summary (continued)

- Software provides the means whereby different algorithms can be run on a general-purpose hardware device
  - Written in programming languages
- Languages such as Python are high-level
- Interpreter translates a Python program to a lower-level form that can be executed on a real computer
- Python shell provides a command prompt for evaluating and viewing the results of Python expressions and statements