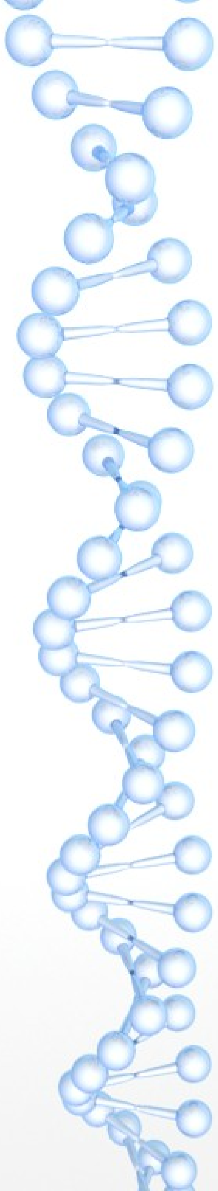


Week 1: Computer Basics

IST 1025: Introduction to Programming

Prof. Audrey Mbogho
Spring 2021
USIU-Africa

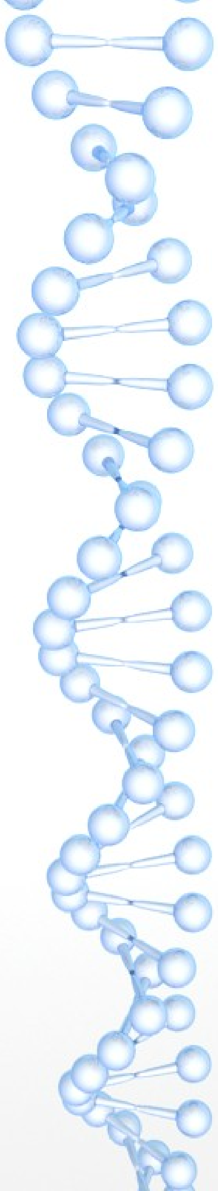
Course Overview

- We will use Python in course
- Out course text is *Fundamentals of Python: First Programs* by Kenneth A. Lambert
- Assessment:

Attendance	10%
Quizzes	20%
Assignments	10%
Midterm	30%
Final	30%

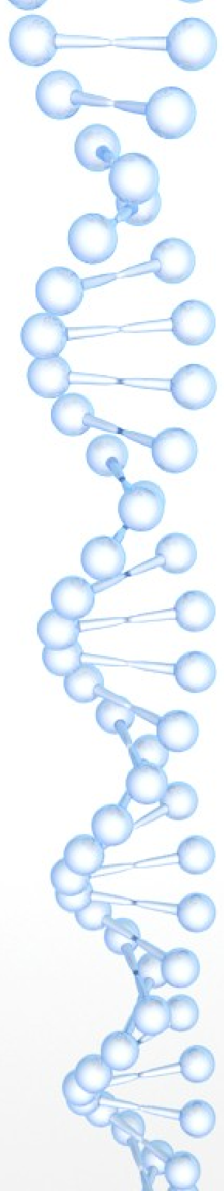
Introduction

- Modern life is heavily dependent on computers.
- How many activities can you think of that use computers?
- Two of the most basic ideas that computer science focuses on are **algorithms** and **information processing**.



Algorithms

- Consider the problem of subtracting one three digit number from another. You might follow the steps below:
  1. Write down the two numbers, with the larger number above the smaller number and their digits aligned in columns from the right.
  2. Start with the rightmost column of digits and work your way left through the rest of the columns.
  3. Write down the difference between the two digits in the current column of digits, borrowing a 1 from the top number’s next column to the left if necessary.
  4. If there is no next column to the left, stop. Otherwise, move to the next column to the left, and go back to Step 3.

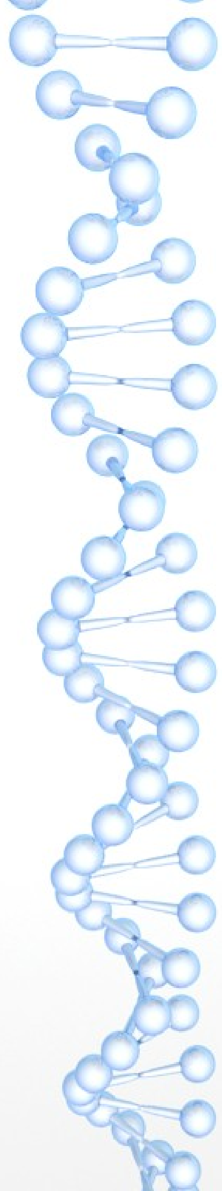


Algorithms

- The above sequence of steps is an example of an algorithm
- An algorithm is like a recipe. It provides a set of instructions that tells us how to do something, such as make change, bake bread, or put together a piece of furniture.
- More precisely, an algorithm describes a process that ends with a solution to a given problem

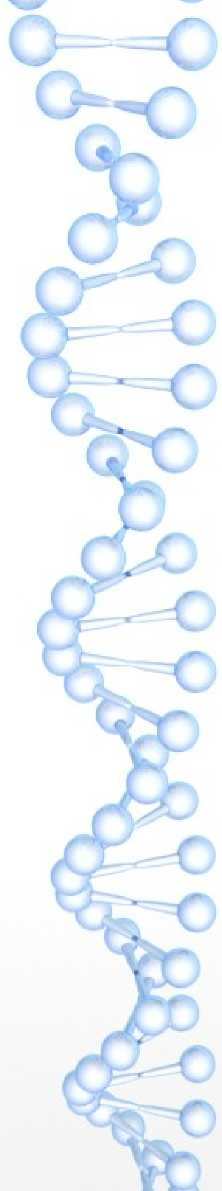
Features of an Algorithm

- An algorithm consists of a finite number of instructions.
- Each individual instruction in an algorithm is well defined.
  - This means that the action described by the instruction can be performed effectively or be executed by a computing agent.
  - For example, any computing agent capable of arithmetic can compute the difference between two digits.
  - So an algorithmic step that says “compute the difference between two digits” would be well defined.
  - On the other hand, a step that says “divide a number by 0” is not well defined, because no computing agent could carry it out.



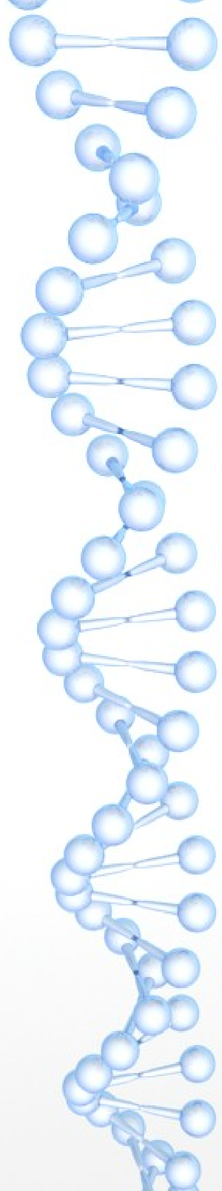
Features of an Algorithm

- An algorithm describes a process that eventually halts after arriving at a solution to a problem.
  - For example, the process of subtraction halts after the computing agent writes down the difference between the two digits in the leftmost column of digits.
- An algorithm solves a general class of problems.
  - For example, an algorithm that describes how to make change should work for any two amounts of money.



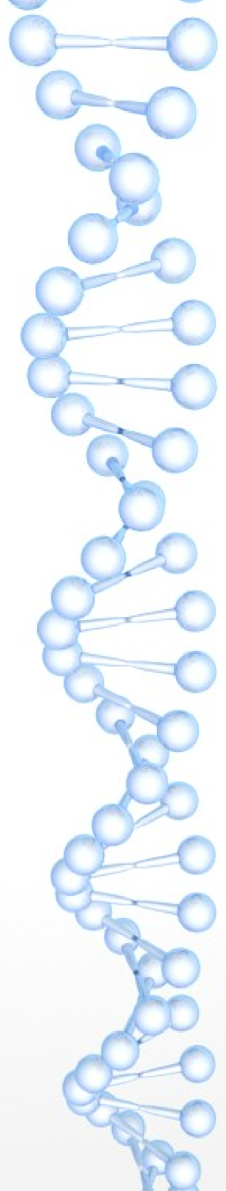
Special- vs. General-Purpose Computers

- Computers can be designed to run a small set of algorithms for performing specialized tasks such as operating a microwave oven.
- But we can also build computers, like the one on your desktop, that are capable of performing a task described by any algorithm.
- These computers are general-purpose problem-solving machines.
- Unlike other tools, computers can be programmed, which is what makes them versatile, general-purpose tools.



Information Processing

- Information processing can be described with algorithms.
- In our earlier example, the subtraction steps involved manipulating symbols used to represent numbers.
- In carrying out the instructions of any algorithm, a computing agent manipulates information.
- The computing agent starts with some given information (known as **input**), transforms this information according to well-defined rules, and produces new information, known as **output**.

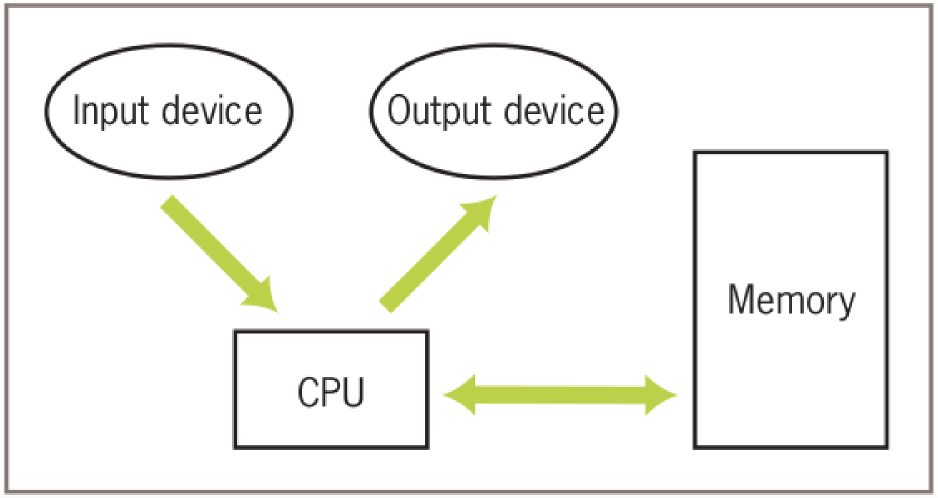


The Structure of a Computer

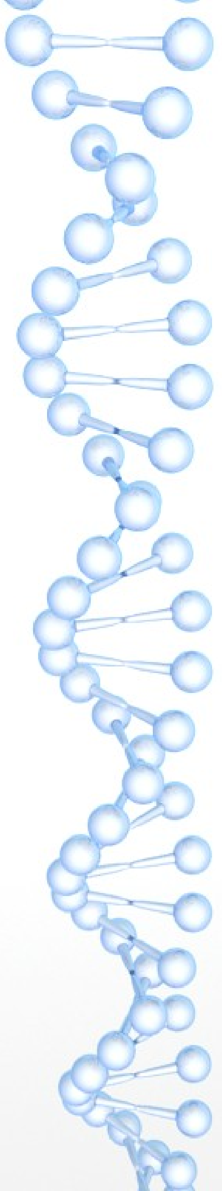
- A modern computer system consists of hardware and software.
- Hardware consists of the physical devices required to execute algorithms.
- Software is the set of these algorithms, represented as programs, in particular programming languages, such as Python, Java, C, C++, etc.
- We will focus on the hardware and software found in a typical desktop computer system, although similar components are also found in other computer systems, such as handheld devices and ATMs

Computer Hardware

- The basic hardware components of a computer are memory, a central processing unit (CPU), and a set of input/output devices, as shown below.



Input and Output Devices



- Human users primarily interact with the input and output devices.
- The input devices include a keyboard, a mouse, a trackpad, a microphone, and a touchscreen.
- Common output devices include a monitor and speakers. Computers can also communicate with the external world through various ports that connect them to networks and to other devices such as smartphones and digital cameras.
- The purpose of most input devices is to convert information that human beings deal with, such as text, images, and sounds, into information for computational processing.
- The purpose of most output devices is to convert the results of this processing back to human-usable form.

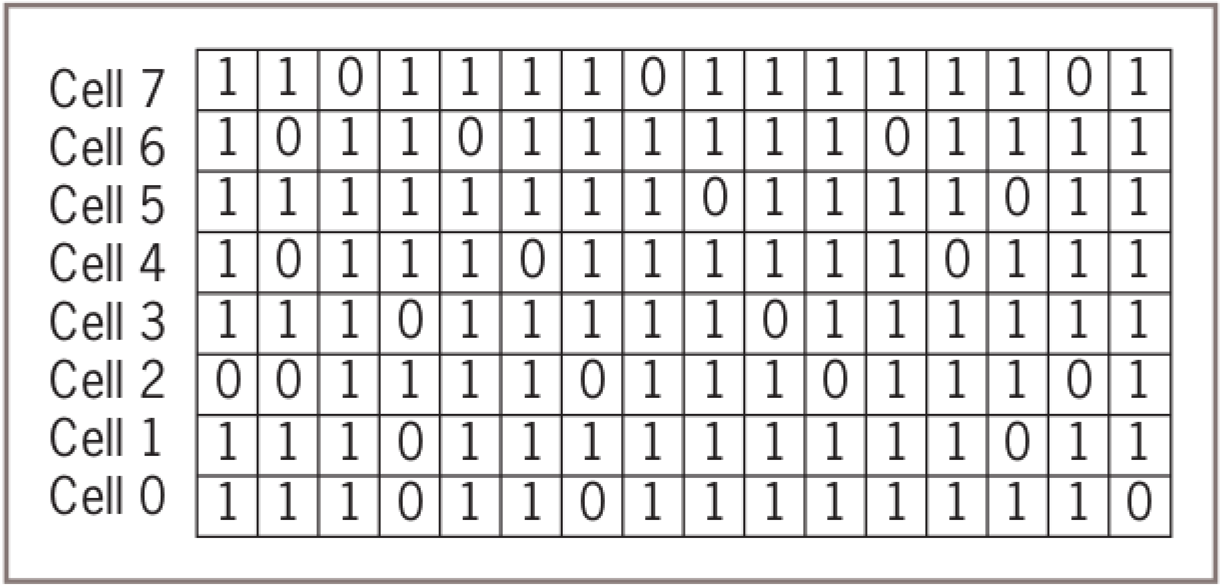
Computer Memory

- Computer memory is set up to represent and store information in electronic form as patterns of binary digits (1s and 0s).
- Consider a basic device such as a light switch, which can only be in one of two states, on or off. Now suppose there is a bank of switches that control 16 small lights in a row. By turning the switches off or on, we can represent any pattern of 16 binary digits (1s and 0s) as patterns of lights that are on or off.
- Computer scientists have discovered how to represent any information, including text, images, and sound, in binary form.

Computer Memory

- Now, suppose there are 8 of these groups of 16 lights. We can select any group of lights and examine or change the state of each light within that collection.
- We have just developed a tiny model of computer memory with 8 cells, each of which can store 16 bits of binary information.
- This memory is also sometimes called **primary** or **internal** or **random access memory (RAM).**
- Information in RAM is meant to be **processed**.

Computer Memory



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

The CPU

- The part of a computer that is responsible for processing data is the **central processing unit (CPU)** or **processor**.
- This device consists of electronic switches arranged to perform simple logical, arithmetic, and control operations.
- The CPU executes an algorithm by fetching its binary instructions from memory, decoding them, and executing them.
- Executing an instruction might involve fetching other binary information—the data—from memory as well.

Secondary Memory

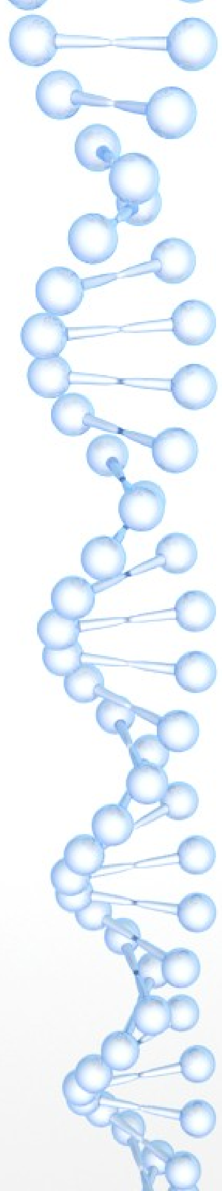
- Information remains in primary memory as long as the computer is on. When the computer is switched off, this information is lost.
- **Secondary** or **external** memory is permanent, and it comes in several forms.
- Magnetic storage media, such as tapes and hard disks, allow bit patterns to be stored as patterns on a magnetic field.
- Semiconductor storage media, such as flash memory sticks, perform much the same function with a different technology, as do optical storage media, such as CDs and DVDs.
- Some of these secondary storage media can hold much larger quantities of information than the internal memory of a computer.

Computer Software

- You have learned that a computer is a general-purpose problem-solving machine.
- To solve any computable problem, a computer must be capable of executing any algorithm.
- Because it is impossible to anticipate all of the problems for which there are algorithmic solutions, there is no way to “hardwire” all potential algorithms into a computer’s hardware.

Computer Software

- Instead, we build some basic operations into the hardware’s processor and require any algorithm to use them.
- The algorithms are converted to binary form and then loaded, with their data, into the computer’s memory.
- The processor can then execute the algorithms’ instructions by running the hardware’s more basic operations.



Computer Software

- Any programs that are stored in memory so that they can be executed later are called software.
- A program stored in computer memory must be represented in binary digits, which is also known as machine code.
- Loading machine code into computer memory one digit at a time would be a tedious, error-prone task for human beings.
- It would be convenient if we could automate this process to get it right every time.

The Loader

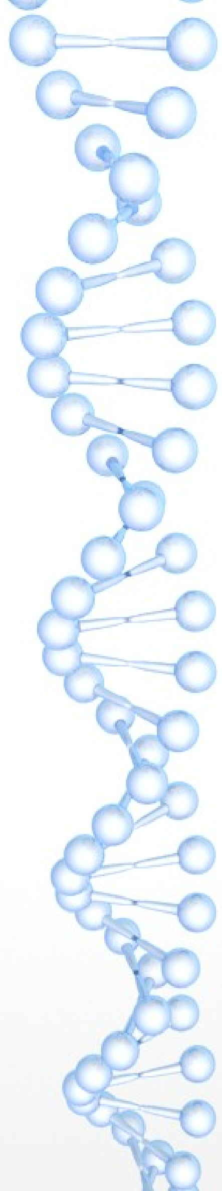
- For this reason, computer scientists have developed another program, called a loader, to perform this task.
- A loader takes a set of machine language instructions as input and loads them into the appropriate memory locations.
- When the loader is finished, the machine language program is ready to execute.
- Obviously, the loader cannot load itself into memory, so this is one of those algorithms that must be hardwired into the computer.

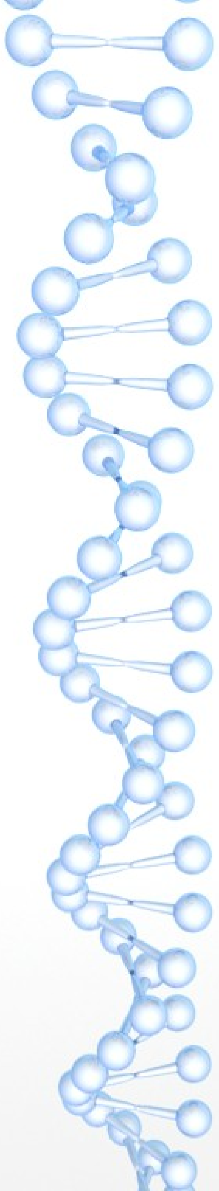
The Operating System

- Now that a loader exists, we can load and execute other programs that make the development, execution, and management of programs easier.
- This type of software is called system. The most important example of system software is a computer’s operating system.
- Familiar examples of operating systems include Linux, Apple’s macOS, and Microsoft’s Windows.

The Operating System

- An operating system is responsible for managing and scheduling several concurrently running programs.
- It also manages the computer’s memory, including the external storage, and manages communications between the CPU, the input/output devices, and other computers on a network.





The Operating Systen

- The operating system provides the following:
- A file system to allow users to organise their data.
- A user interface to allow users to interact with the computer’s software. These include:
  - terminal-based interfaces
  - graphical user interfaces (GUI)
  - touchscreen interfaces.
  - voice user interfaces (VUI).

Application Software

- Another major type of software is called applications software, or simply apps.
- An application is a program that is designed for a specific task, such as editing a document or displaying a Web page.
- Applications include Web browsers, word processors, spreadsheets, database managers, graphic design packages, music production systems, and games, among millions of others.
- As you begin learning to write computer programs, you will focus on writing simple applications.

Programming Languages

- As you have learned, computer hardware can execute only instructions that are written in binary form—that is, in machine language.
- Writing a machine language program, however, would be an extremely tedious, error-prone task.
- To ease the process of writing computer programs, computer scientists have developed high-level programming languages for expressing algorithms.
- These languages resemble English and allow the author to express algorithms in a form that other people can understand.

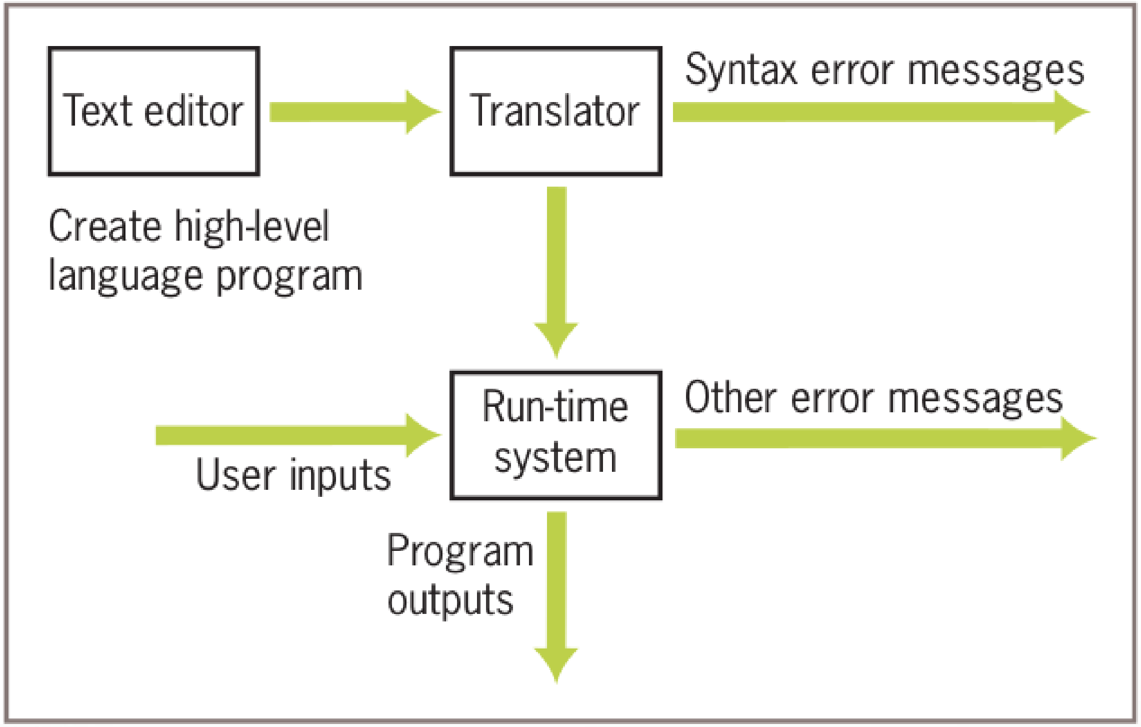
Programming

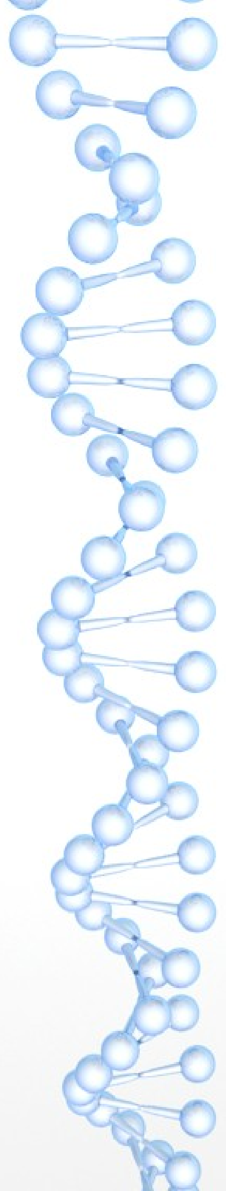
- A programmer typically starts by writing high-level language statements in a text editor.
- The programmer then runs another program called a translator to convert the high-level program code into executable code.
- Because it is possible for a programmer to make grammatical mistakes even when writing high-level code, the translator checks for syntax errors before it completes the translation process.
- If it detects any of these errors, the translator alerts the programmer via error messages.

Programming

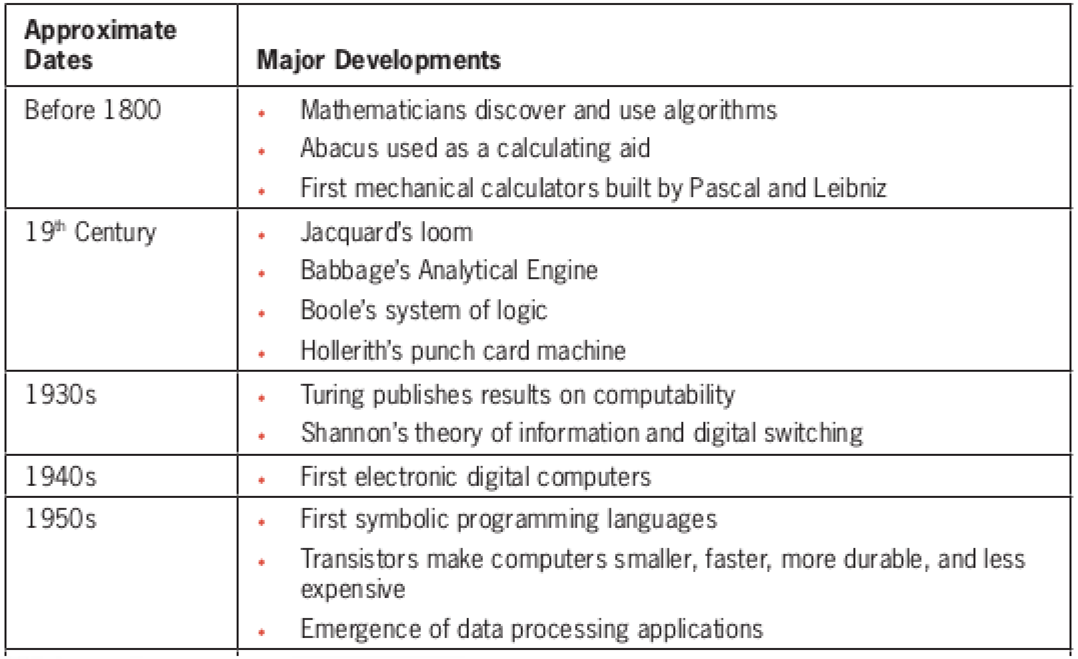
- The programmer then has to revise the program.
- If the translation process succeeds without a syntax error, the program can be executed by the run-time system.
- The run-time system might execute the program directly on the hardware or run yet another program called an interpreter or virtual machine to execute the program.

The Coding Process



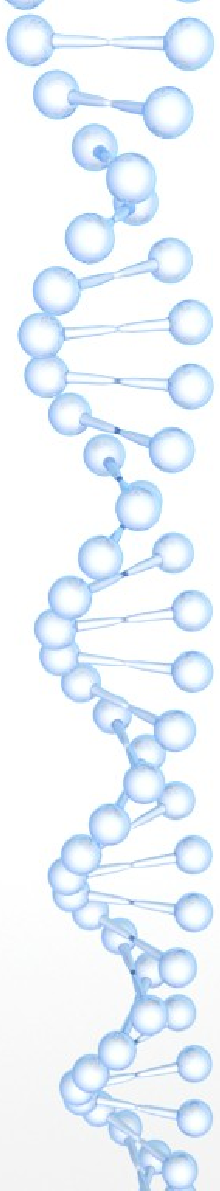


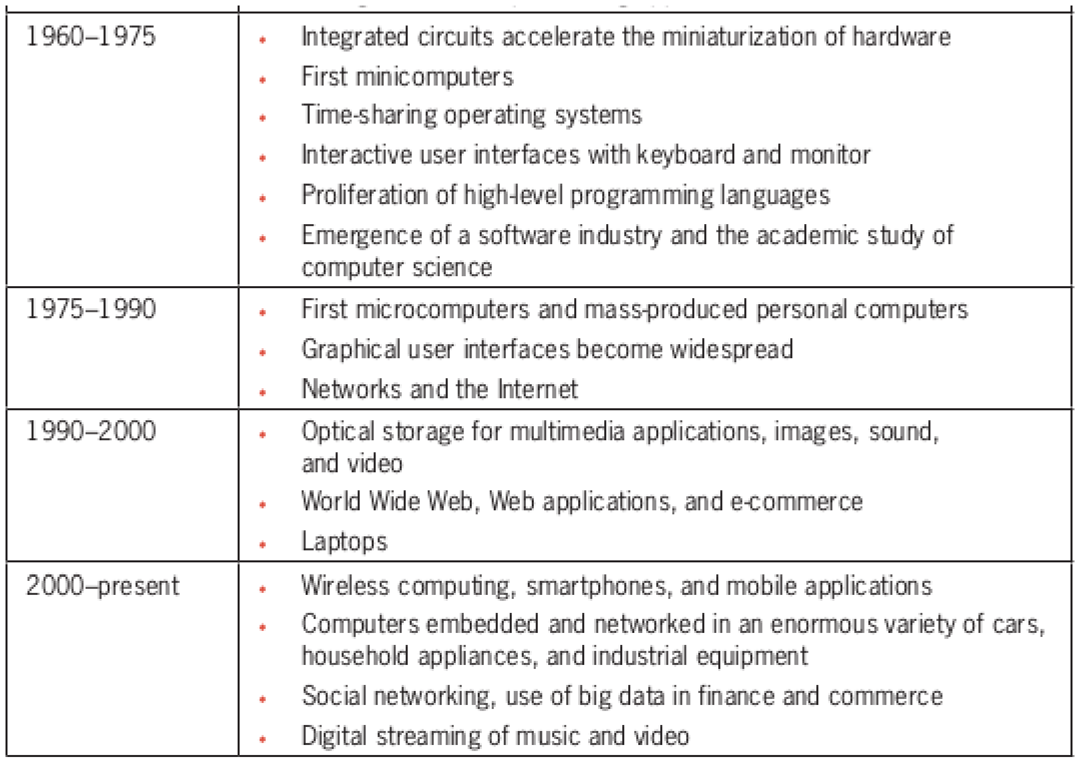
A History of the Computer



Approximate Dates	Major Developments
Before 1800	• Mathematicians discover and use algorithms • Abacus used as a calculating aid • First mechanical calculators built by Pascal and Leibniz
19th Century	• Jacquard's loom • Babbage's Analytical Engine • Boole's system of logic • Hollerith's punch card machine
1930s	• Turing publishes results on computability • Shannon's theory of information and digital switching
1940s	• First electronic digital computers
1950s	• First symbolic programming languages • Transistors make computers smaller, faster, more durable, and less expensive • Emergence of data processing applications

A History of the Computer

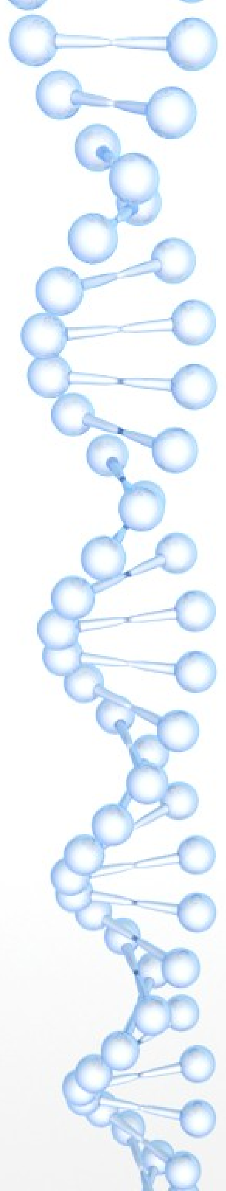




1960–1975	• Integrated circuits accelerate the miniaturization of hardware • First minicomputers • Time-sharing operating systems • Interactive user interfaces with keyboard and monitor • Proliferation of high-level programming languages • Emergence of a software industry and the academic study of computer science
1975–1990	• First microcomputers and mass-produced personal computers • Graphical user interfaces become widespread • Networks and the Internet
1990–2000	• Optical storage for multimedia applications, images, sound, and video • World Wide Web, Web applications, and e-commerce • Laptops
2000–present	• Wireless computing, smartphones, and mobile applications • Computers embedded and networked in an enormous variety of cars, household appliances, and industrial equipment • Social networking, use of big data in finance and commerce • Digital streaming of music and video

Getting Started with Python

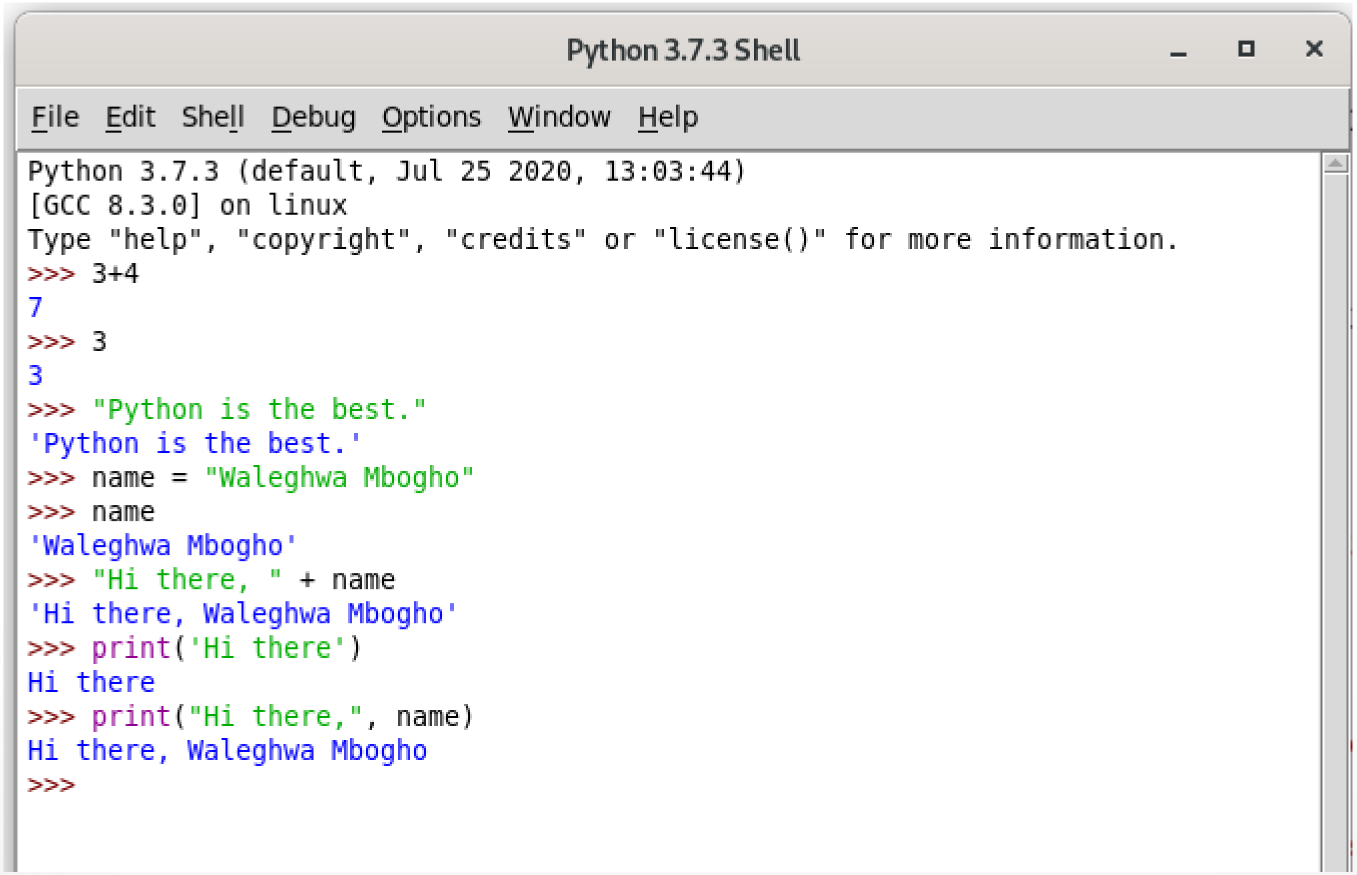
- Guido van Rossum invented the Python programming language in the early 1990s.
- Python is a high-level, general-purpose programming language for solving problems on modern computer systems.
- The language and many supporting tools are free, and Python programs can run on any operating system.
- You can download Python, its documentation, and related materials from www.python.org.



The Python Shell

- Python is an interpreted language, and you can run simple Python expressions and statements in an interactive programming environment called the shell.
- The easiest way to open a Python shell is to launch the IDLE (Integrated DeveLopment Environment).
- This is an integrated program development environment that comes with the Python installation.
- There are two major versions of Python: 2 and 3. We will be using Python 3 in this course.

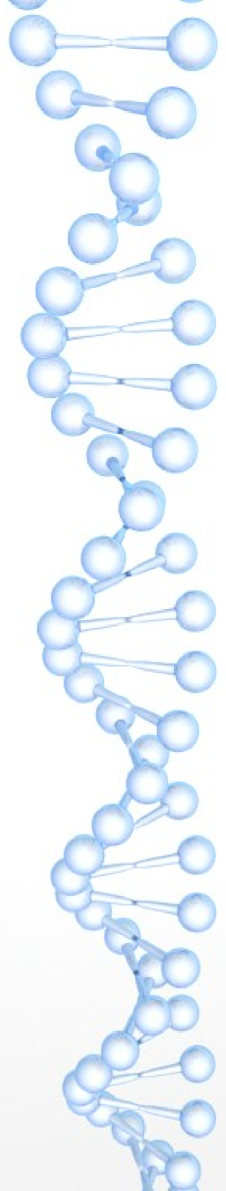
The Python Shell



```
Python 3.7.3 (default, Jul 25 2020, 13:03:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> 3+4
7
>>> 3
3
>>> "Python is the best."
'Python is the best.'
>>> name = "Waleghwa Mbogho"
>>> name
'Waleghwa Mbogho'
>>> "Hi there, " + name
'Hi there, Waleghwa Mbogho'
>>> print('Hi there')
Hi there
>>> print("Hi there,", name)
Hi there, Waleghwa Mbogho
>>>
```

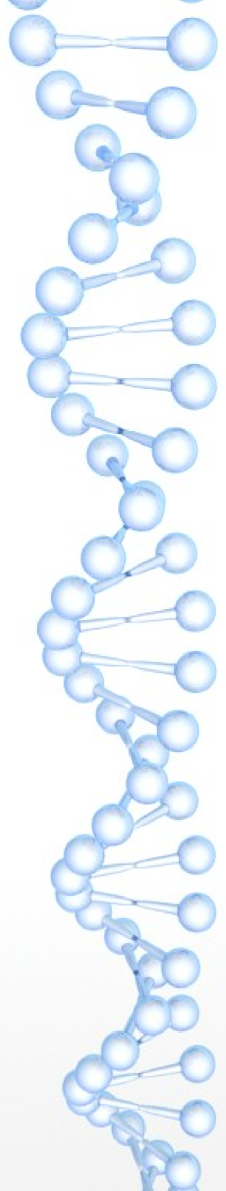
The Python Shell

- When you enter an expression or statement, Python evaluates it and displays its result, if there is one, followed by a new prompt.
- Note the colour coding in IDLE. This helps you to quickly identify various program elements.
- The Python shell is useful for experimenting with short expressions or statements to learn new features of the language, test out ideas, as well as for consulting documentation on the language.
- To quit the Python shell, you can either select the window’s close box or press the ctrl+D key combination.



Input, Processing and Output

- Most useful programs accept inputs from some source, process these inputs, and then finally output results to some destination.
- In terminal-based interactive programs, the input source is the keyboard, and the output destination is the terminal display.
- The Python shell itself is such a program; its inputs are Python expressions or statements. Its processing evaluates these items. Its outputs are the results displayed in the shell.

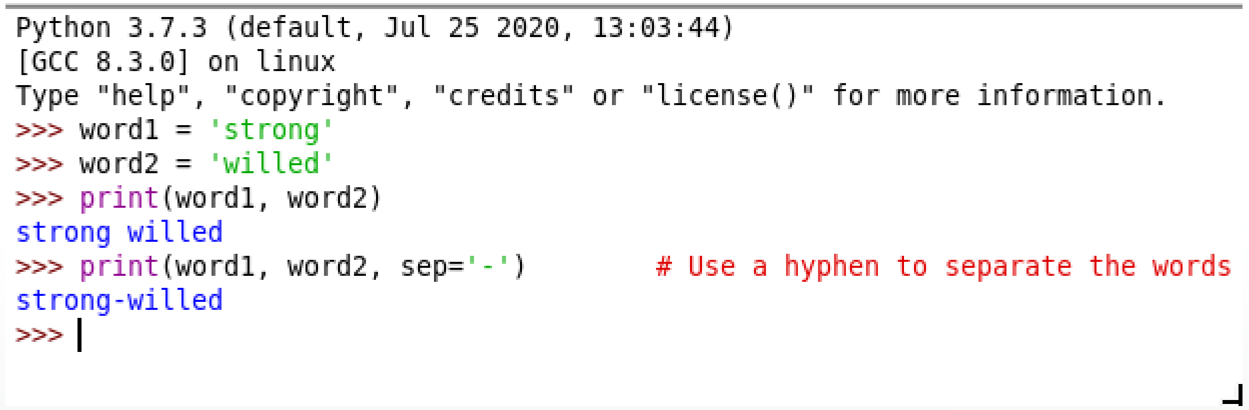


The print function

- The programmer can also force the output of a value by using the print function.
  - print(<expression>)
- Python evaluates <expression> and then displays it.
- In the example below, the expression is evaluated, resulting in the value 5, and then it is printed.
  - print(3 + 2)

The print function

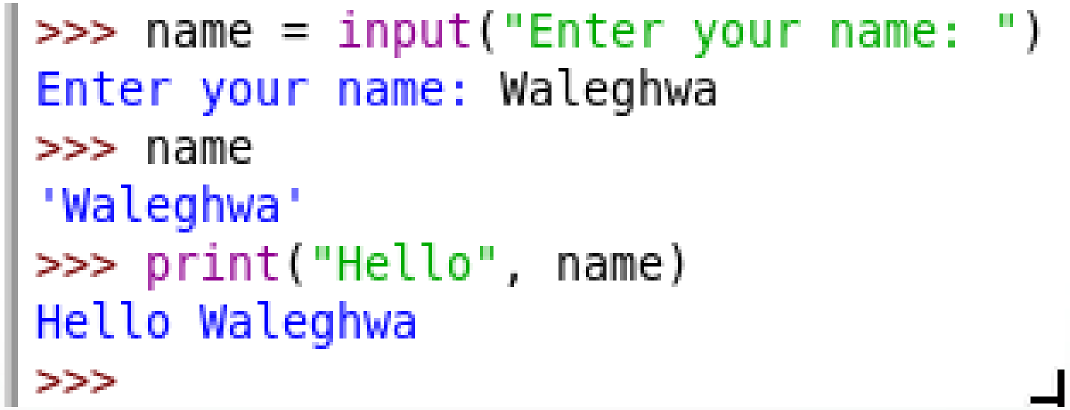
- You can pass multiple expressions to the print function by separating them with a comma.
  - print(“hello”, name)
- The print function will insert a space between the displayed values.
- It you don’t want this behaviour, you can use the sep parameter to change it.



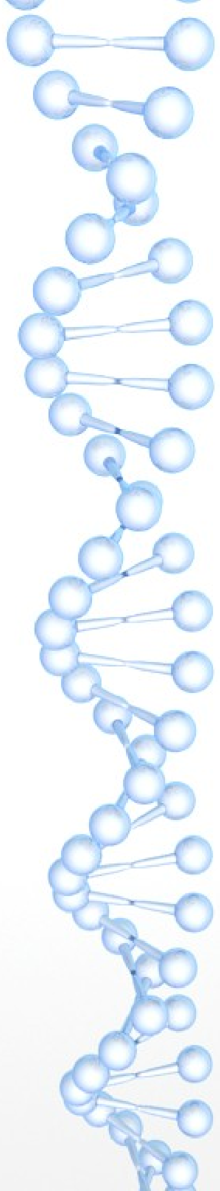
```
Python 3.7.3 (default, Jul 25 2020, 13:03:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> word1 = 'strong'
>>> word2 = 'willed'
>>> print(word1, word2)
strong willed
>>> print(word1, word2, sep='-')        # Use a hyphen to separate the words
strong-willed
>>> |
```

Receiving Input

- As you create programs in Python, you'll often want your programs to ask the user for input.
- You can do this by using the input function.



```
>>> name = input("Enter your name: ")
Enter your name: Waleghwa
>>> name
'Waleghwa'
>>> print("Hello", name)
Hello Waleghwa
>>>
```



Receiving Input

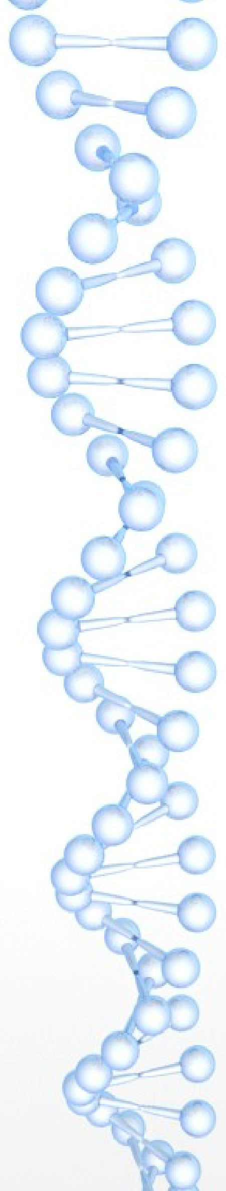
- The input function does the following:
  - Displays a prompt for the input. In this example, the prompt is "Enter your name: " .
  - Receives a string of keystrokes, called characters, entered at the keyboard and returns the string to the shell.
- The prompt was provided to the input function as an argument.
- An argument is a piece of information that a function needs to do its work.

Variables

- The string returned by the function in our example is saved by assigning it to the variable name.
- The form of an assignment statement with the input function is the following:
- <variable identifier> = input(<a string prompt>)
- A variable identifier (or just variable) is a name that stores a value. The variable can be used again and again wherever that value is needed.

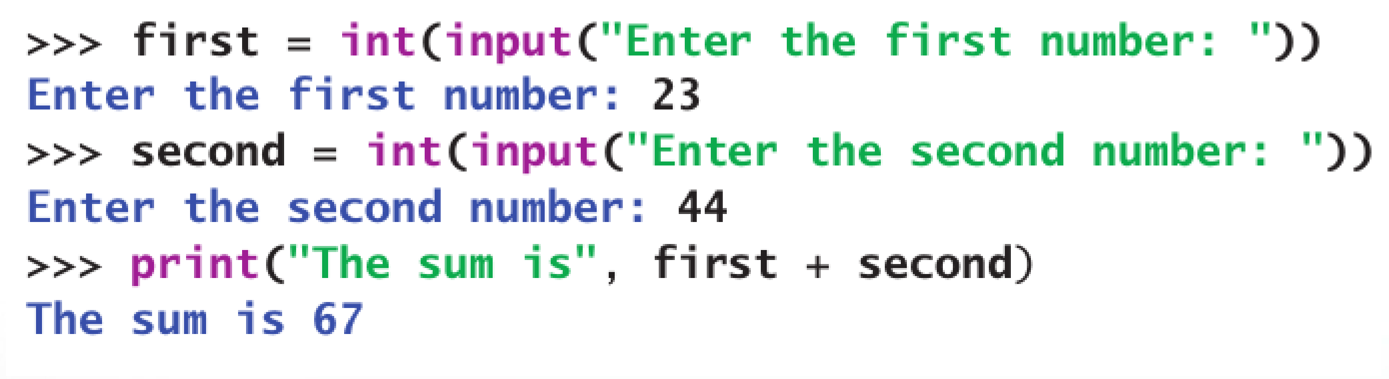
Inputting Numbers

- The input function always builds a string from the user’s keystrokes and returns it to the program.
- After inputting strings that represent numbers, the programmer must convert them from strings to the appropriate numeric types.
- In Python, there are two type conversion functions for this purpose, called int (for integers) and float (for floating-point numbers).



Inputting Numbers

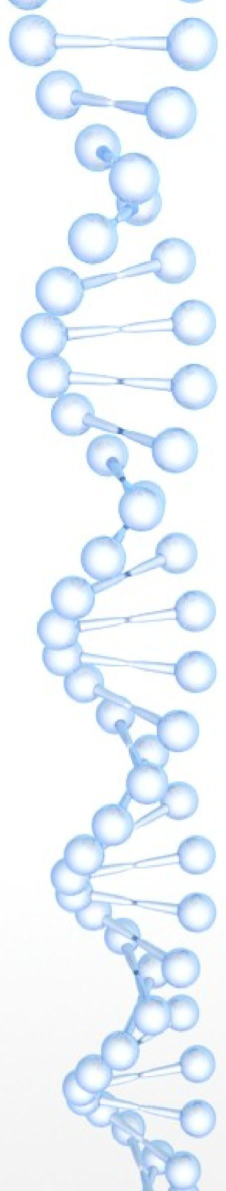
- In the session below, a string is first returned by the input function. Then the string is passed to the int function to be converted to an integer.



```
>>> 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
```

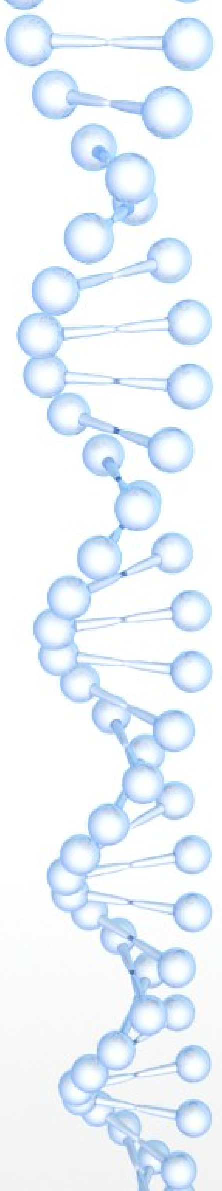
Python Scripts

- While it is easy to try out short Python expressions and statements interactively at a shell prompt, it is more convenient to compose, edit, and save longer, more complex programs in files.
- We can then run these program files or scripts either within IDLE or from the operating system’s command prompt without opening IDLE.
- Script files are also the means by which Python programs are saved permanently and distributed to others.

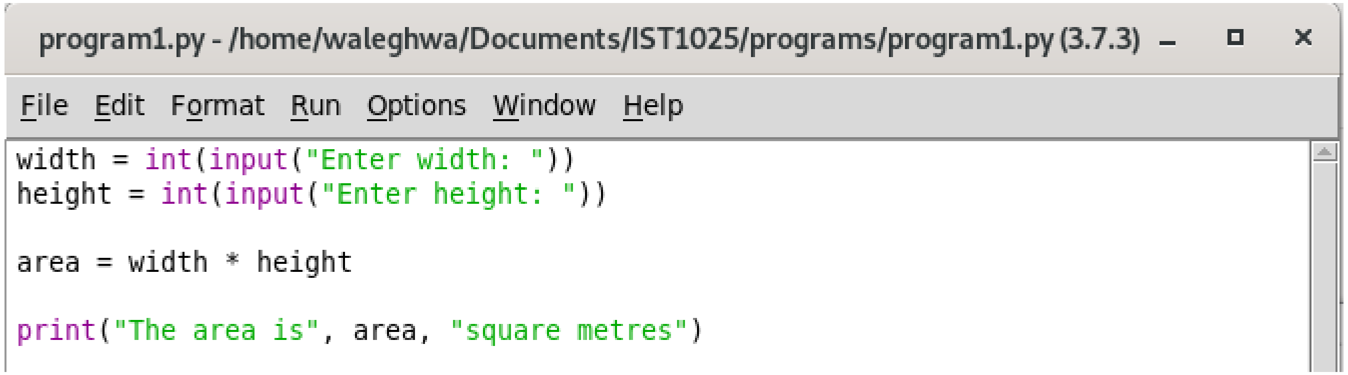


Creating and Executing a Script in IDLE

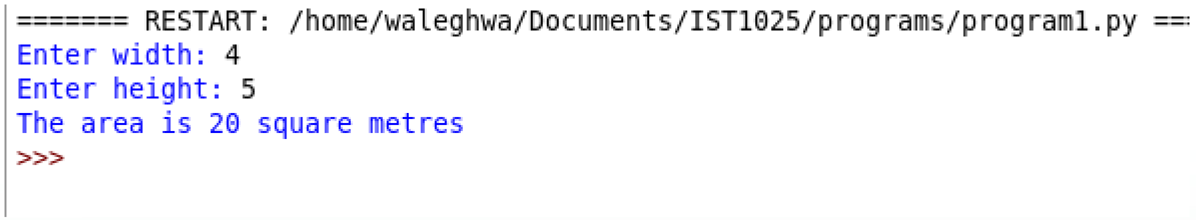
- Select the option New File from the File menu of the shell window.
- In the new window, enter Python expressions or statements on separate lines, in the order in which you want Python to execute them.
- At any point, you may save the file by selecting File/Save. If you do this, you should use a . py extension. For example, your first program file might be named myprogram.py.
- To run this file of code as a Python script, select Run Module from the Run menu or press F5.



Creating and Running a Script

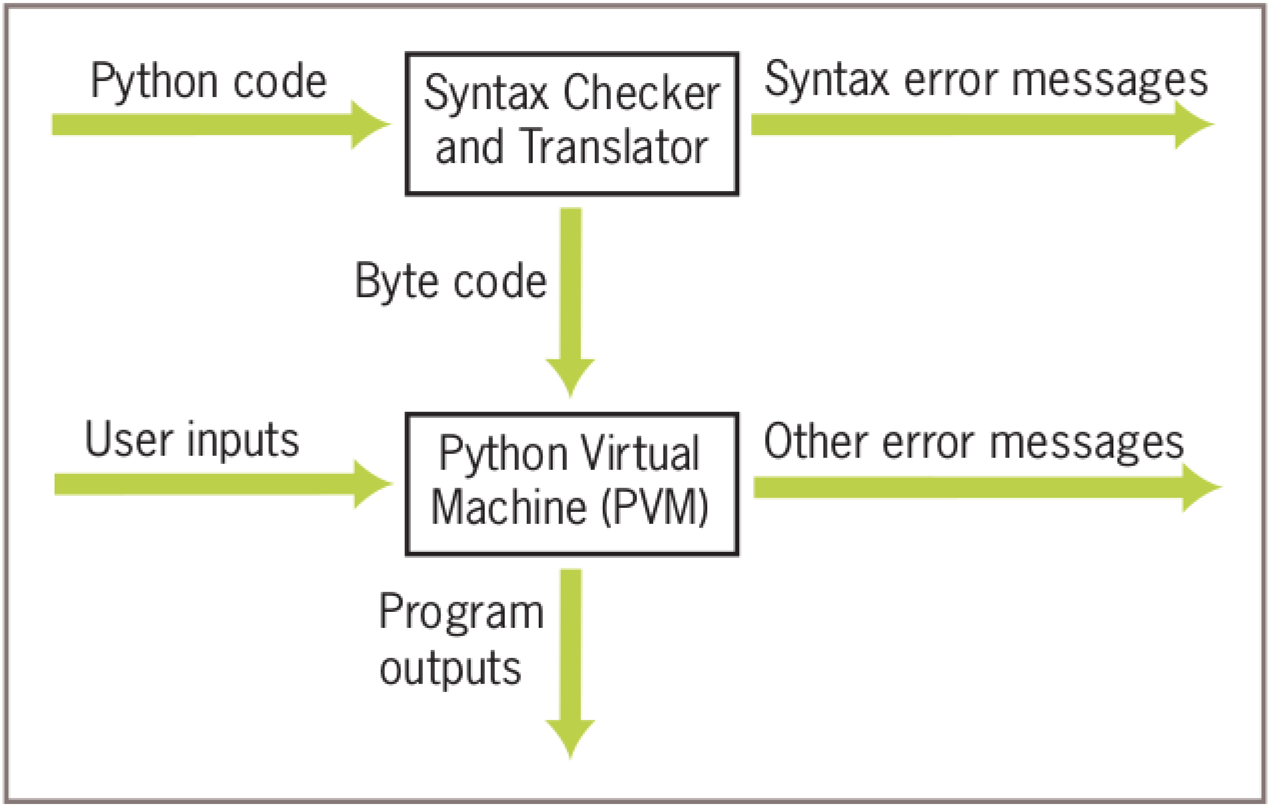

program1.py - /home/waleghwa/Documents/IST1025/programs/program1.py (3.7.3)
File Edit Format Run Options Window Help

```
width = int(input("Enter width: "))
height = int(input("Enter height: "))

area = width * height

print("The area is", area, "square metres")
```



```
======= RESTART: /home/waleghwa/Documents/IST1025/programs/program1.py ==
Enter width: 4
Enter height: 5
The area is 20 square metres
>>>
```

Steps in Interpreting a Python Program

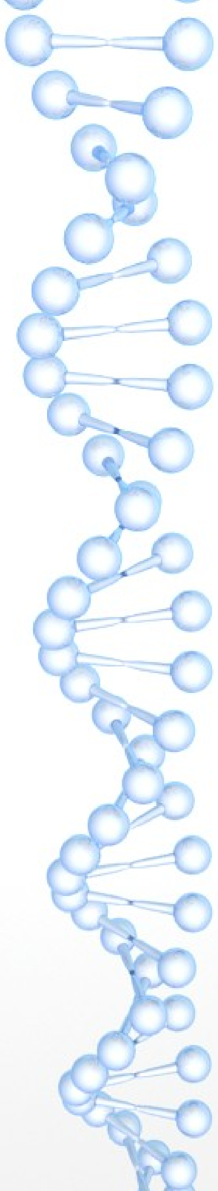


Steps in Interpreting a Python Program

- The interpreter reads a Python expression or statement, also called the source code, and verifies that it is well formed. As soon as the interpreter encounters such a syntax error, it halts translation with an error message.
- If a Python expression is well formed, the interpreter then translates it to an equivalent form in a low-level language called byte code.
- When the interpreter runs a script, it completely translates it to byte code.

Steps in Interpreting a Python Program

- This byte code is next sent to another software component, called the Python virtual machine (PVM), where it is executed.
- If another error occurs during this step, execution also halts with an error message.
- This is a run-time error. Dividing by zero is an example of a run-time error.
- Another type of error, that goes undetected and unreported, is the logic error where your code does a computation other than what is required to solve the problem at hand.



Exercise

- Write a script that asks the user for three things: their name, the amount of money they have, and the price of the item the wish to buy.
- The program then calculates and reports how much money the user has left after they have bought the item.