Programming with Python 1

Python for Non-programmers

Babar Ali

Topics

- Computer programs
  - What is a program?
  - Why program?
  - Not useful programming
- Some Introductory Concepts:
  - Components of a program
  - Memory: Variables and Disks
  - Current working directory
- Python, Anaconda, Spyder

PROGRAMS

What is a program?

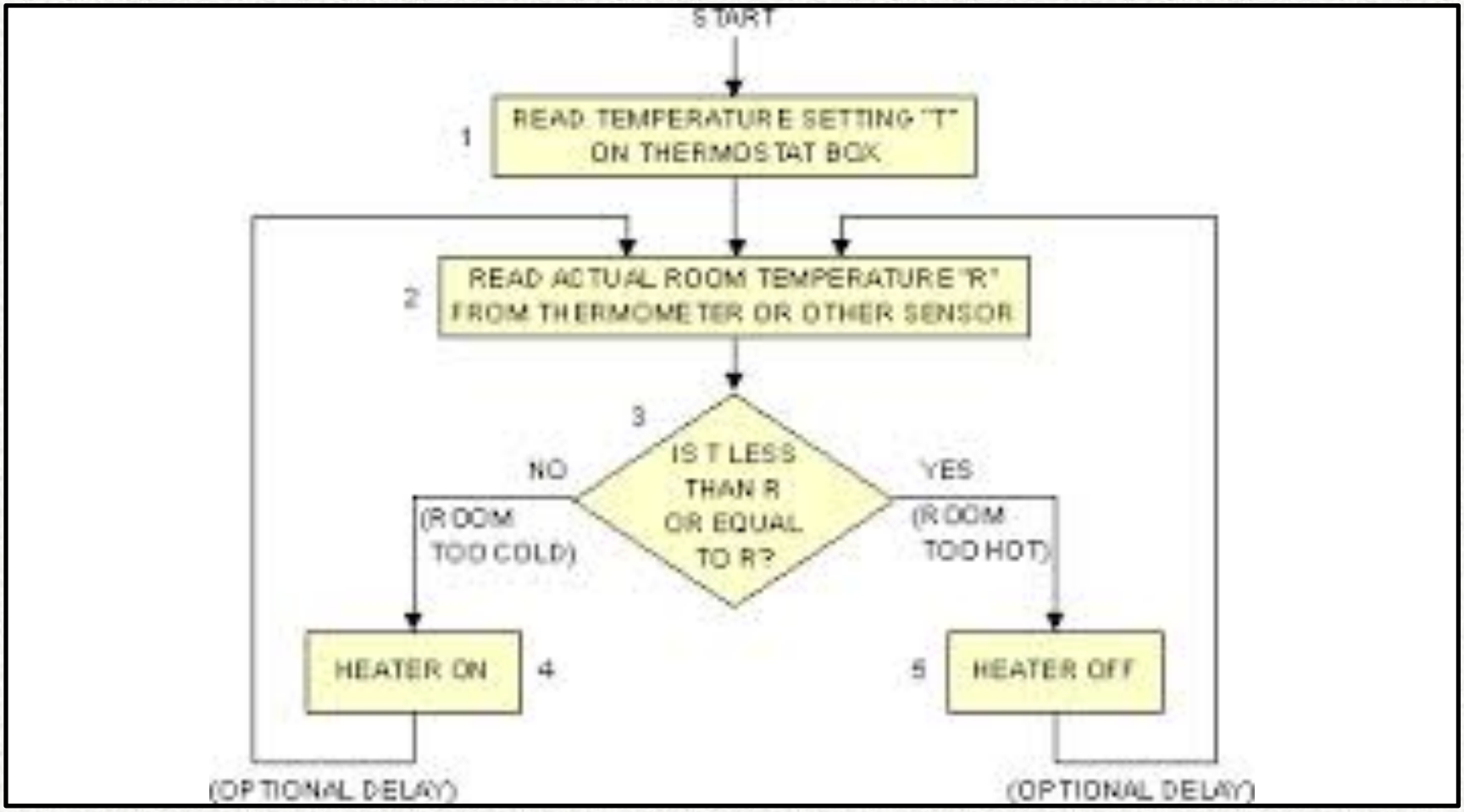
6 **a :** a plan for the programming of a mechanism (as a computer)

b : a sequence of coded instructions that can be inserted into a mechanism (as a computer)

c : a sequence of coded instructions (as genes or behavioral responses) that is part of an organism

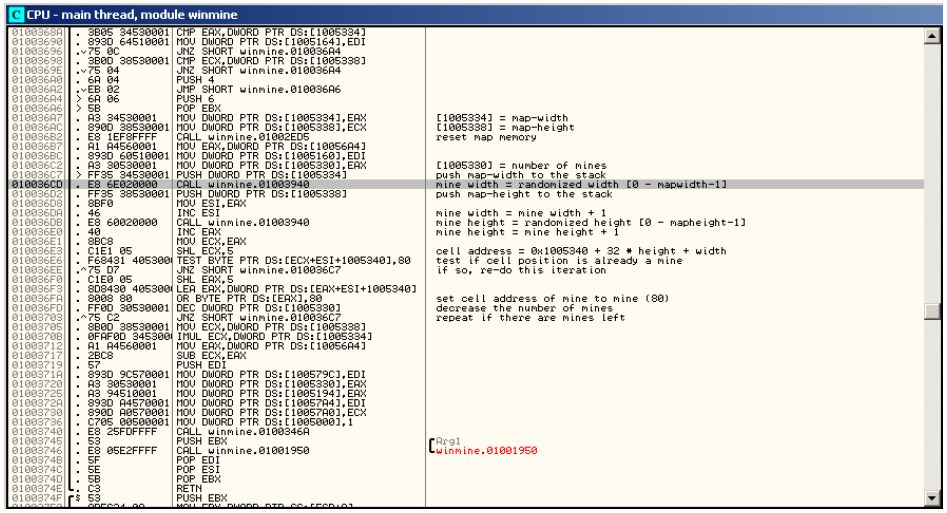
Merriam-Webster online dictionary

What is a program?



What is a program?

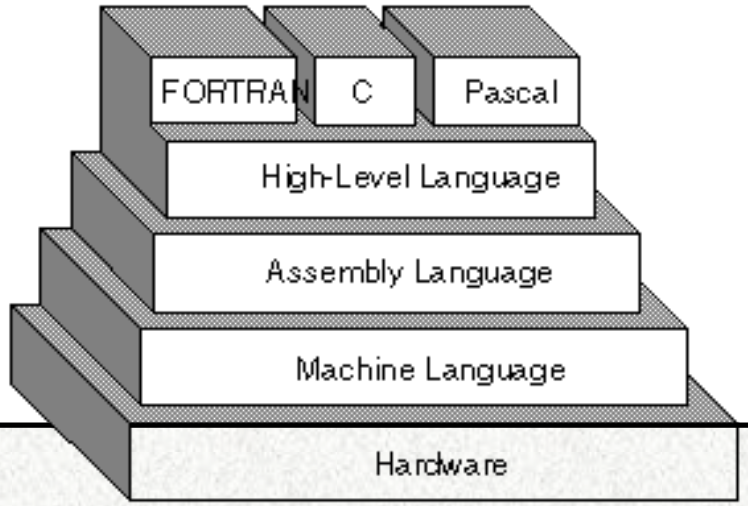
- Its how you tell a computer what you would like it (computer) to do for you.
- However …



- At the fundamental (lowest) level, computers speak machine language.

Machine language is ...

- Specific to the computer architecture.
  - Intel, x86 IBM, powerpc
- Difficult for (most) humans.
- Really only useful for the most fundamental operations (e.g. put 0 in memory register ....).



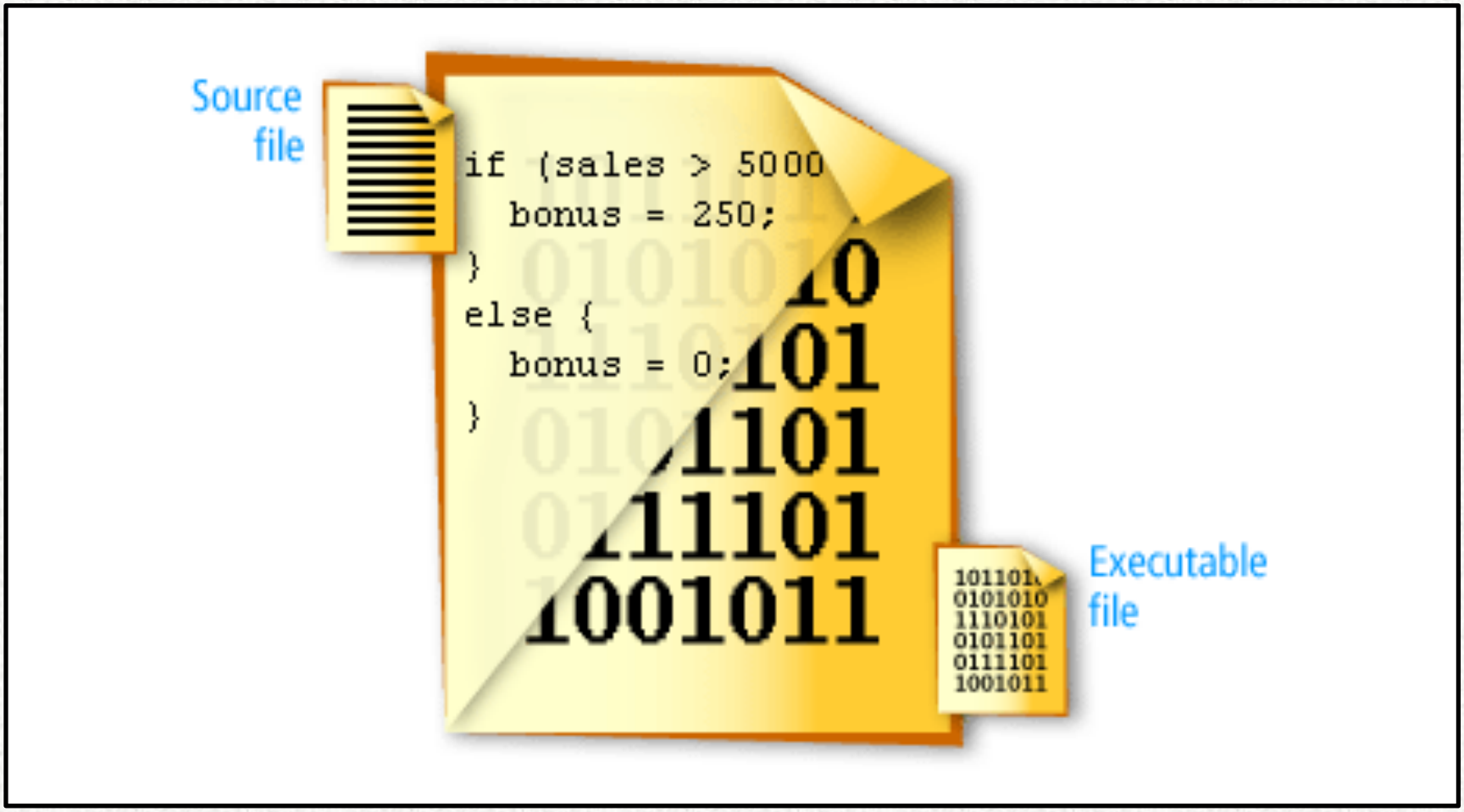
NASM code to print “hello world”

```
        global  _start

        section .text
_start:
        ; write(1, message, 13)
        mov     eax, 4                  ; system call 4 is write
        mov     ebx, 1                  ; file handle 1 is stdout
        mov     ecx, message            ; address of string to output
        mov     edx, 13                 ; number of bytes
        int     80h

        ; exit(0)
        mov     eax, 1                  ; system call 1 is exit
        mov     ebx, 0                  ; we want return code 0
        int     80h
message:
        db      "Hello, World", 10
```

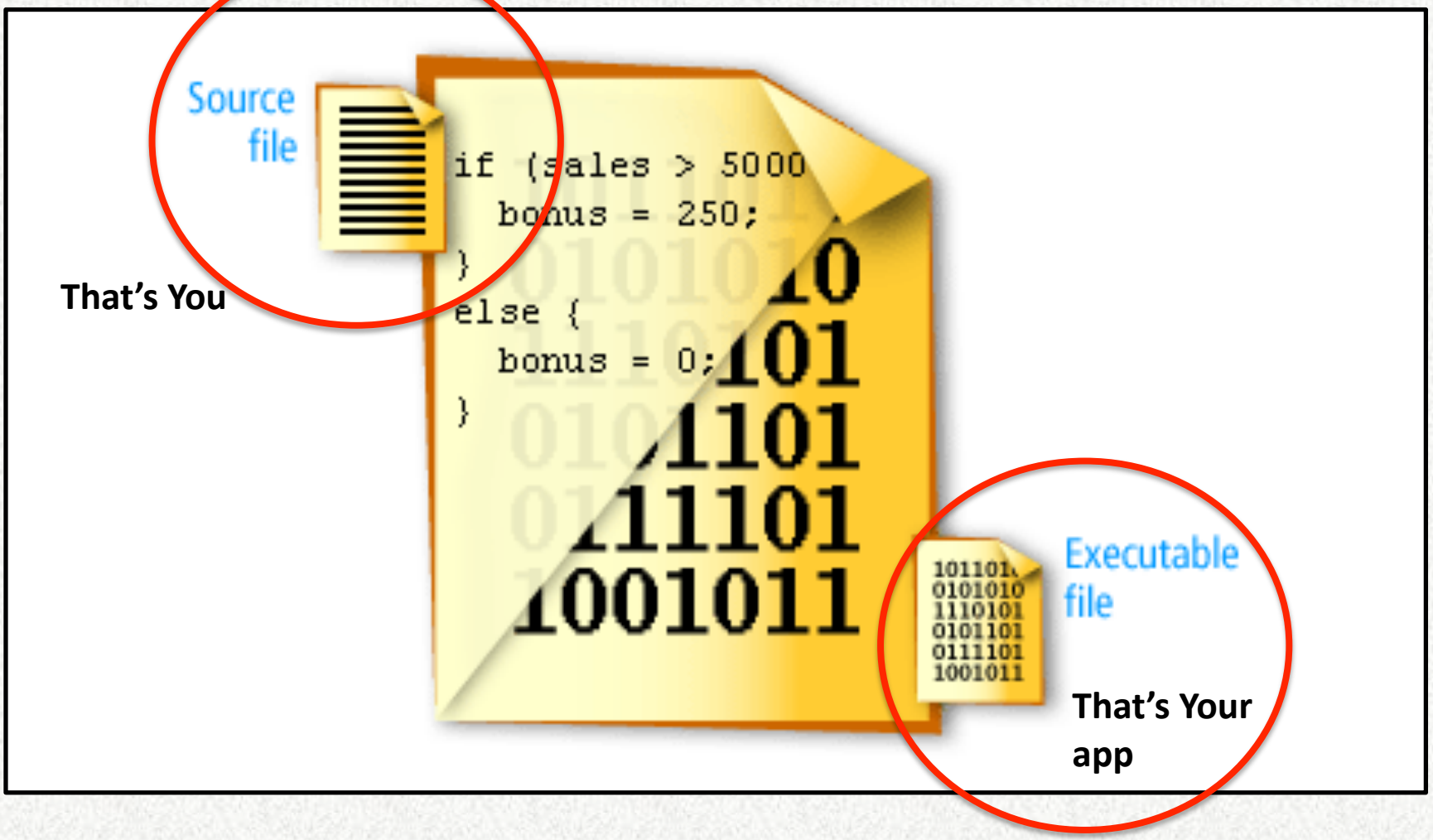
Normal Humans Need Translators

- "**High-level programming languages**" invented to make it easier to talk to machines.
- But, at the expense of efficiency, and some other aspects – not really our concern.
- High-level languages:
  - Provide a bridge between humans and computer's preferred language level
  - Are not specific to machine architecture
  - Require a "compiler"

Role of Compiler



Role of Compiler



“Hello World” in C

```
/* Hello World program */

#include<stdio.h>

main()
{
   printf("Hello World");
}
```

"Hello World" in C

- But, the instructions must be "**compiled**" (read translated) from C-language to assembler
- Here's one way:

  gcc -o helloworld -ansi helloworld.c
- As long as 'gcc' compiler exists on a machine, you can write and run C-code on it
- The resulting assembly language instruction will differ from machine to machine

Two flavors of high-level languages

Compiled

- Require code to be compiled to run
- Examples: C, C++, FORTRAN, …
- Pros: Faster, more control, more flexible
- Cons: Harder, and compiling gets cumbersome

Scripting

- Interpreter/compiler is "always ON" – just run it.
- Examples: Python, Perl
- Pros: Easier to use, More "portable"
- Cons: SLOOWWW, less flexibility

“Hello World” in Python

```
print “Hello world”
```

Comparison

C	Python

/* Hello World program */
#include<stdio.h>
main()
{
printf("Hello World");
}
 	
• Then
gcc -o helloworld -ansi helloworld.c
 	
• Then
helloworld
	print “Hello World”

Writing Programs

- High-level programs and scripts are simply written as ASCII (a.k.a text) files.
- Simple programs may only use one file for all of their instructions.

  E.g. print “Hello World”
- Complex programs rely on system engineering concepts to organize 100s or 1000s+ files.

More on Writing Programs

- You can use simple text editors: vi, textedit (MacOS), notepad (Windows), emacs, etc.

Spyder

- Software specifically designed to help you write programs.
  - Will provide automatic checks for the fundamentals: E.g. parenthesis, typos
  - Use color-coding to highlight specific types or blocks of code.
  - Will automatically perform basics like indentation
  - And … much, much more (to be covered in Spyder)

How do you execute, run programs?

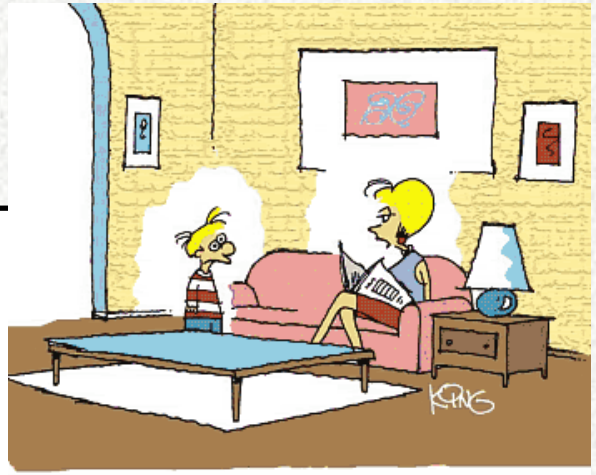
- First, the format that is ready for execution is called an executable.
  - An executable is usually a compiled (remember translated) version of the program for compiled languages.
  - The equivalent for scripting languages is simply the script itself.

Running programs, cont.

- Many ways to start:
  - Command line.
  - Double click.
  - Tap (an app on a ipod, smart phone)
  - Issue the proper command in the development environment.
  - Instructed by other programs at the basic level.
  - And, many more.

Jargon

- **Code**
  - usually refers to instructions for a compiled or scripting computer language.
- **Script**
  - usually refers to scripting language code.
- **Program**
  - usually refers to compiled language code, but also script and code
- **Run** = **Execute**
  - Have the computer carry out the instructions in your code
- **Executable**
  - Format in which your code is able to run. Script for scripting language and translated assembly language code for compiled language.



"No, you weren't downloaded. Your were born."

More Jargon

- **Bug**
  - A fault in the computer code. As simple as a typo to as complex as fault in programming logic
- **Debug (de-bug)**
  - The process by which you identify and remove the bugs. Or, the command to do so.
- **Portable**
  - Able to run on many types of operating system + machine architecture combination. 100% portability is a myth.

Yet More Jargon

- **Function** ... in programming context.
  - A basic programming unit. You write a function to do one thing (usually).
- **Library** ... in programming context.
  - An organized collection of programs, i. e. functions that is (usually) focused on a specific topic.
  - Example: CFITSIO is a set of C-language functions devoted to FITS format input and output calls.
- **App**
  - An executable
- **Comment**
  - Line(s) in the code that the compiler or interpreter (for scripts) skips and are meant only for humans.

Using Programs

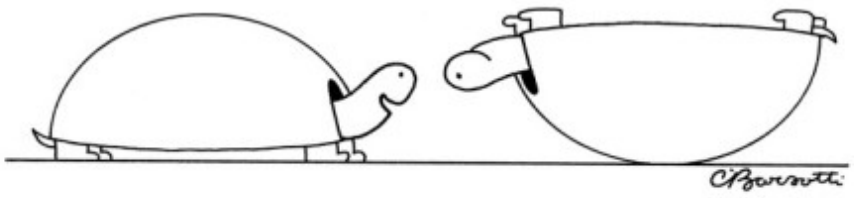
- Examples:
  - Manipulate and use data that are tedious or impossible to do by hand (on a calculator, say).
  - Minimize errors: Computers are good at math (intel PENTINUM issues not withstanding).
  - Repeat the same operation many times for a given star on a multiple stars

More Examples

- Read an APT output table file into memory.
- Write a DS9 region file using a table of either (x,y) or (ra,dec) positions and ID.
- Read ASCII format data table.
- Write ASCII format data table.
- Merge photometry tables.
- Create 2-dimensional plots of various types: scatter plots, line plots, histograms. These plots may have multiple data sets on them.
- Save plots to a JPG, PNG or similar graphic format file.

Not Useful Programming

- Obviously, anything that does not involve computers.
- Otherwise, few instances when dealing with computers do not benefit from programming:
  - Writing large chunks of text such as a novel, unless the text can be auto-generated.
  - Anything that requires interactivity (web browsing, computer games)



"Wow, I've never met an astronomer before."

INTRODUCTORY CONCEPTS

The basics

Programs are organized series of instructions for the computer that:

- use variables and disks to hold data in memory
- access computer's CPU to process data
- allow users to pass or manipulate data
- are built on functions and libraries of functions

A simple program

- Typically,
  - Write a **function** to do a fundamental manipulation or operation.
    - *We will skip this for now*
  - Write an master program to call and execute functions in a set order or logic.
- Most of your programs should fall in this category

What Beginners Do (and its okay)

- Typically,
  - ➢ Write a function to do a fundamental manipulation or operation.
    - *We will skip this for now*
  - ➢ **Write an master program to call and execute all instruction in a set order or logic.**
- Most of your programs should fall in this category

Adding complexity

- Users are prompted for input.
- Read data, instructions from a separate file
- Execute functions based on conditions.
- Do many, many, many things instead of a single one.
- Incorporate Graphical User Interfaces (GUIs).

Won't Do In This Series

- Users are prompted for input.
- Read data, instructions from a separate file
- Execute functions based on conditions.
- ~~Do many, many, many things instead of a single one.~~
- ~~Incorporate Graphical User Interfaces (GUIs).~~

Good practices

- Comment often and be verbose. Imagine talking to yourself 5+ years from writing the code.
- Test and debug as much as possible.
- If using the same operation over and over, define a function.
- Define and use naming conventions for variables, files.
- Keep the logic simple and easily readable even at the expense of efficiency.
- Organize code in logical blocks that are clearly separated.

Memory

- There are Two types:
  - Random Access Memory (RAM)
  - Disk/physical Storage.
- RAM is “live” memory only available when the program is executing (running).
- Disks store information using some semi-permanent physical mechanism that is not dependent on power.

“I’m not losing my memory. I’m living in the now.”

Memory and programs

- Programs rely on **RAM for faster execution**.
- Programs rely on **Disks for permanent storage**.

- RAM is active when the program is running.
- RAM is lost when you exit the programming environment.
- Stored files/data on disk may be accessed inside and outside of programs.

Variables in memory

- Variables, in computer jargon, are elements of a programming language that deal with RAM to hold data during execution.
- There are many different types of variables ranging from byte (least memory) to quadruple precision arrays (most memory) to store precise numerical values.
- Good practice:

  minimize memory consumption = use variable types appropriate for the job.

Disks

- Programs use physical disk space to store data and/or results in files.
- Accessing disks is 100s-1000s of times slower than accessing RAM.
- It's a bad idea to use files instead of variables for memory during execution.
- It's a bad idea to use variables instead of files to store final results – variables cease to exist as soon as the program terminates.

Exercise

- NGC 281 Level 1 Herschel/PACS frames for the blue channel comprises about 150,000 images.
- Each image has 2048 pixels (70 micron data).
- Each pixel data is 64-bits = 8 Bytes in HIPE.
- Calculate the size (in Gigabytes) of the variable in RAM that holds the level 1 cube.
- Calculate the size of the FITS files on disk that holds the data, assuming that
  - Only 16-bits = 2 bytes are used for storage.
  - FITS format compresses data by a factor of 2.

Current Working Directory

- When reading or writing (storing) files, the computer reads, writes to a given folder.
- This folder is the “current working directory”.
- Scripting or compiling languages have different assumptions for where the “current working directory” lies.
- Default is (usually) where you started executing the program.
- Best practice: Define it explicitly in your program.

python

Why Python?

- Python is a scripting language:
- Easier to program than compiling languages.
- Example:

  Define the variable ‘a’ and set it to contain the value 10.2 in memory

float a;
a = 10.2;

In C: First declare the variable, then use it.

a = 10.2

In Python: Simply use it.

Which Python?

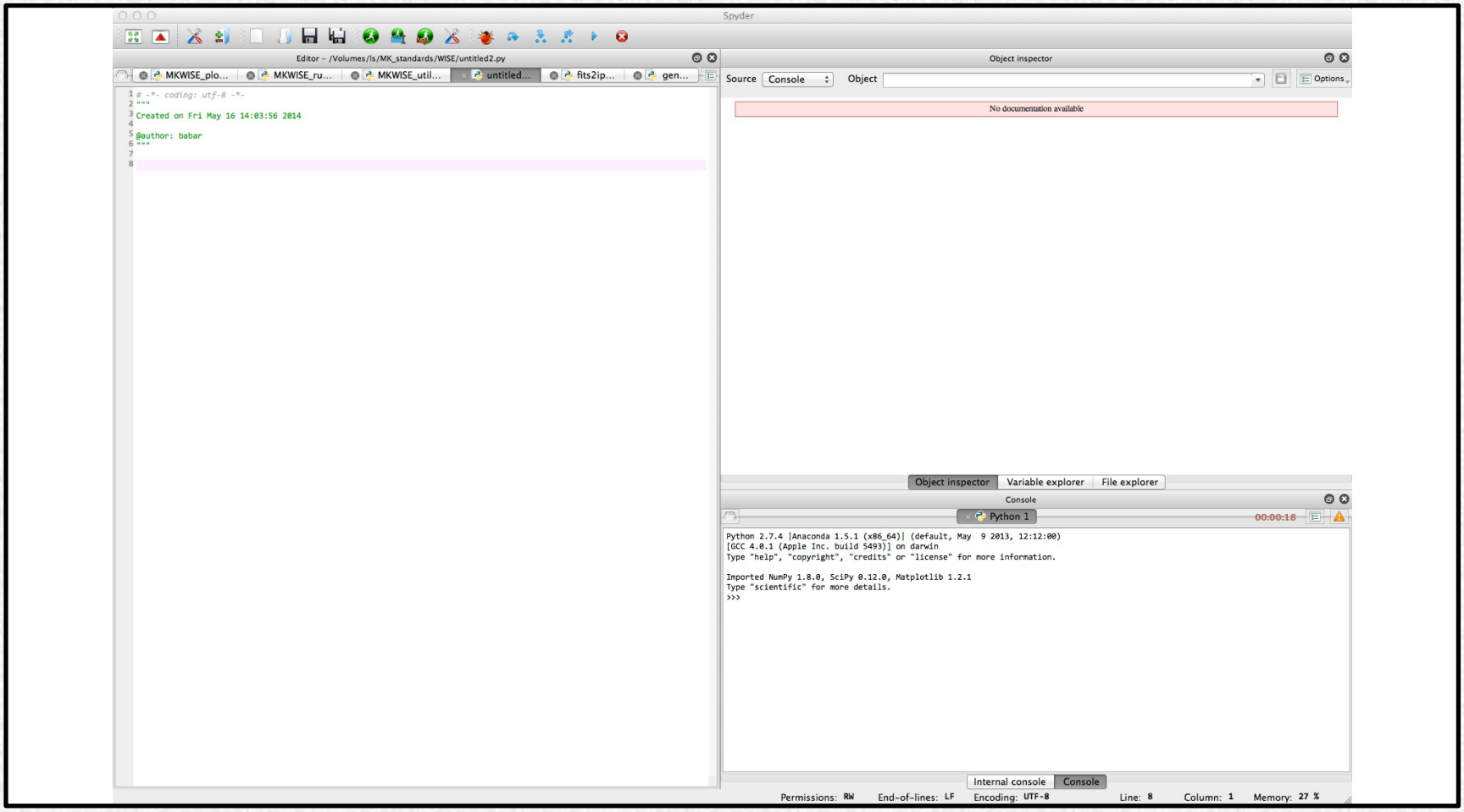
- Barebones python is not a practical choice.
- A pre-packaged distribution solves a number of issues:
  - Useful libraries are already included and compiled.
  - Version conflicts between libraries are avoided.

Anaconda

http://continuum.io/downloads.html

- Its free.
- Windows, MacOS, and Linux support.
- numpy, scipy libraries included.
- plotting libraries included.
- Many (not all) useful astronomy libraries are included.

Spyder


Spyder
Editor – /Volumes/ls/MK_standards/WISE/untitled2.py
MKWISE_plo...
MKWISE_ru...
MKWISE_util...
untitled...
fits2ip...
gen...
-*- coding: utf-8 -*-
"""
Created on Fri May 16 14:03:56 2014
@author: babar
"""
Object inspector
Source
Console
Object
Options
No documentation available
Object inspector
Variable explorer
File explorer
Console
Python 1
00:00:18
Python 2.7.4 |Anaconda 1.5.1 (x86_64)| (default, May 9 2013, 12:12:00)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Imported NumPy 1.8.0, SciPy 0.12.0, Matplotlib 1.2.1
Type "scientific" for more details.
>>>
Internal console
Console
Permissions: RW
End-of-lines: LF
Encoding: UTF-8
Line: 8
Column: 1
Memory: 27 %