

Introduction to Scientific Programming/Python

CS 410P Spring 2022

MW 11:10-12:30 – KING N328

(L01) TR 2:10-3 HS 107

(L02) TR 3:10-4 HAALND 104

Arvind Narayan

Office: Kingsbury W239

anarayan@cs.unh.edu

Office Hours: M 10-10:30 (W244 lobby), WR 3-4 zoom

Course Web Site: <http://cs.unh.edu/~cs410p/>

Username: student Password: zone

Introduction to Scientific Programming/Python

- Teaching Assistants:
- Pooja Oza – Labs/Office hours
- Xihong Su – Labs/Office hours
- (Abhinav Gupta – Office hours)
- PAC Programming Assistance Center (Discord)
 - <http://pac.cs.unh.edu/>

Books & Grading

- Suggested textbooks:
 - Introduction to Programming using Python, David I. Schneider
 - Fundamentals of Python, Kenneth A. Lambert
 - Introduction to Programming using Python, Y. Daniel Liang
- Documentation at: <https://www.python.org/doc/>
- Course Grading
 - 22% Lab Assignments (~18)
 - 40% Programming Assignments (~10)
 - 7% Worksheets (during class/lab, be there or lose it)
 - 16% Quizzes
 - 15% Final Exam

Labs & Programming Assignments

- Due midnight on Date Due
- Lab Assignments many are due same day or the next day – 0L, 1L, 2L, etc.
- Programming Assignments once every week to 10 days – 0P, 1P, 2P, etc.
- Late Submission
 - Labs are not accepted late
 - Programming assignments accepted up to 3 days after due date
 - 10% late penalty per day
 - Medical or other unforeseen circumstances should be brought up to the instructor immediately
- Although course seemingly starts slow, it picks up in pace and many programs often require significant effort:
 - **Start early! Complete multiple cycles of design/write/test**
 - Lab assignments lead up to the programming assignment

Why Learn to Program?

- Your work will likely require these
 - (1) Models: weather, car crashes, stocks... (2) Statistics: means, error budgets, .. (3) Records: inventory, medical, GIS... (4) Hardware control: sensors, motors, communication (5) Management: risk assessment, process control ...
- You'll improve your problem solving skills

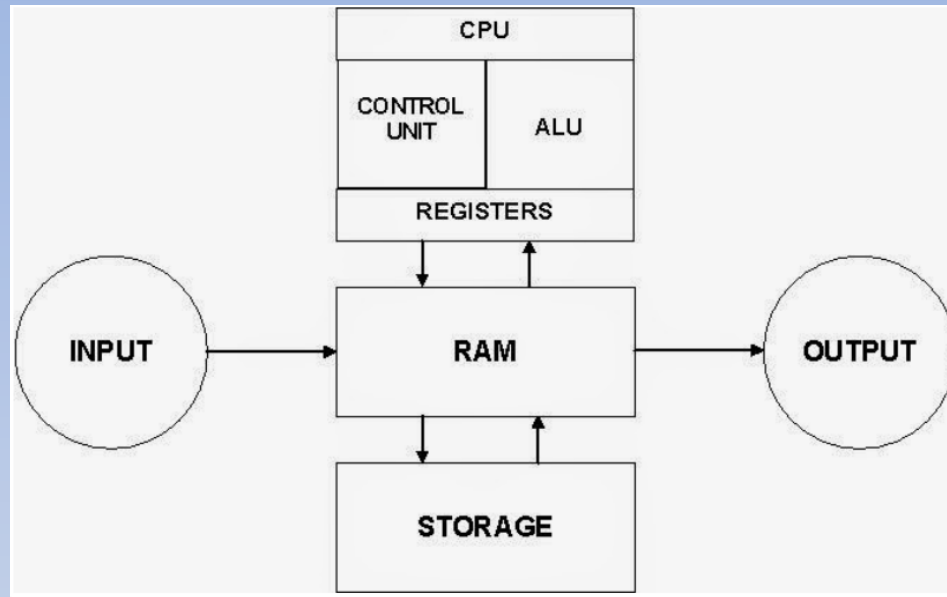
Your work will involve tools
that *almost* do the job

- Models you need to tweak
- Statistics with different assumptions
- Records with special cross-queries
- “Canned” patterns of hardware control
- Stuff from the web that gets you *close*
- ...anything the tool creator didn't think of

Things You'll Learn

- Programming is a problem solving activity!
- Concepts present in any language
 - Variables, data types, & data structures
 - Control structures
 - Functions
 - Files, User I/O
 - Beginnings of Object oriented programming
- Skills needed whenever you program
 - Testing & debugging
 - Incremental development
 - User interface design (“user friendliness”)

Computer Components



- Basic Computer Components
 - CPU, Memory, Secondary Storage, Input & Output devices
- CPU – Brain of the computer
 - Main components: ALU, CU
 - Uniprocessor, Multiprocessor
- Memory
 - Two attributes: Address and Contents
 - Digital representation - Binary numbers

Computer Components

- Binary \Leftrightarrow Decimal Conversion
- Units of data/memory:
 - Bit (*Binary Digit*)
 - Byte
 - Memory word – number of bits viewed as single unit of information (usually called “word”)
 - 32-bit word, 64-bit word

Computer Components

- CPU-Memory Interaction for executing a “Program”

List of Instructions: *Program*

Example:

- Add two numbers whose values are supplied by the user
- Translates to instructions for adding two numbers stored in specific memory locations

Computer Components

Input/Output Devices

- Keyboard, Mouse, Video Monitor, Printer
- Typing a key sends a signal (digital code) to represent the information sent
- Mouse click sends the coordinate of the cursor location to the CPU
- Display/Output information is similarly sent from the CPU to the output device

Computer Components

- Secondary Storage
 - Hard disk, CD, Magnetic tape, Floppy disk, Zip disk

- Storage Units

Byte

8 bits

Kilobyte (KB)

1024 bytes

2^{10}

Megabyte (MB)

1048576 bytes

2^{20}

Gigabyte (GB)

Terabyte (TB)

8 bits = 1 byte
1,024 Bytes = 1 Kilobyte
1,024 Kilobytes = 1 Megabyte
1,024 Megabytes = 1 Gigabyte
1,024 Gigabytes = 1 Terabyte
1,024 Terabytes = 1 Petabyte
1,024 Petabytes = 1 Exabyte
1,024 Exabytes = 1 Zettabyte
1,024 Zettabyte = 1 Zottabyte
1,024 Zottabyte = 1 Brontobyte

Computer Components

- Secondary Storage
 - Floppy disk (ancient!)
 - Magnetic tape drive – Sequential access
 - Zip disk
 - CD Vs DVD
 - Hard disk (built-in component, external)
 - Flash drive
- Directories or Folders, Files, File Structure

Computer Software

- Computer components – “hardware”
- List of instructions to be executed on the hardware is called “software” (or “program”)
- Operating System (O.S) – Collection of computer programs that control the interaction between the user and computer hardware
- O.S is loaded into memory from disk at boot-time
 - Examples:
 - Macintosh OS, Windows (graphical user interface)
 - Unix, Linux, VMS, MS-DOS, OS/2 (command-line interface, some graphical)

Computer Software

- Functions of an O.S (System Software)
 - Execute commands from the user
 - Validate users and maintain security for access of information
 - Processor, Memory and Resource Management
 - Interpretation of input/output information
 - Data retrieval and storage on to secondary storage devices

Computer Software

- Application Software
 - Programs to assist computer users in performing tasks
 - MS-WORD – Word processing application
 - MS-Excel – Spreadsheet application
 - dBase, MS-Access – database management applications
 - Several applications are “bundled” with the o.s and available after the o.s install
 - Additional applications can be installed anytime
- Application Software Vs System Software

Computer Software

Computer Language – Set of instructions written in a particular manner following rules and syntax specified

Computer hardware understands only machine language

Machine language is a collection of binary numbers

Assembly language is more readable, specific to an o.s, consists of instructions and instruction codes

Higher-level language is a much more readable set of instructions and can be executed on any computer

High-level languages: FORTRAN, COBOL, Lisp, C, Ada, C++, Java, Python

Questions and Answers

- How do we communicate with the computer?
 - Programming languages
- How do we get computers to perform complicated tasks?
 - Tasks are broken down into a sequence of instructions
- Why Python?
 - Powerful, easy to download and work with across platforms, quick learning curve, widespread applications

Questions and Answers

- How did the language Python get its name?
 - Named for the British comedy group Monty Python (really!)
- Book by Schneider uses the editor IDLE to create programs. How did IDLE get its name?
 - Stands for **I**ntegrated **D**evelopment **E**nvironment
- What is an interpreted language?
 - Uses an interpreter, translates high-level language one statement at a time into machine language and then runs

Questions and Answers











- How are problems solved with a program?
 - Step-by-step procedure devised to process given data and produce requested output.
- What is a zero-based numbering system?
 - Numbering begins with zero instead of one
- Prerequisites to learning Python?
 - Be familiar with how folders and files are managed, editing files
 - Basic computer navigation
 - Curiosity to experiment with the unknown

Python Distributions

- From <https://wiki.python.org/moin/PythonDistributions>

Python Distributions

Aside from the official CPython distribution available from python.org, other distributions based on CPython include the following:

- »  [ActivePython](#) from [ActiveState](#)
- »  [Anaconda](#) from Continuum Analytics
- »  [ChinesePython Project](#): Translation of Python's keywords, internal types and classes into Chinese. Eventually allows a programmer to write Python programs in Chinese.
- »  [Enthought's Canopy](#)
- »  [Win9xPython](#): Backport of mainline CPython 2.6/2.7 to old versions of Windows 9x/NT.
- » [IPython](#) and its [IPyKit](#) variant
- »  [PocketPython](#)
- »  [Portable Python](#): Run Python from USB device - no installation needed.
- » [PyIMSL Studio](#)
- » [PyPy](#): a Python implementation in Python.
- »  [Python\(x,y\)](#): Python(x,y) is a scientific-oriented Python Distribution based on Qt, Eclipse and  [Spyder](#)
- » [PythonForArmLinux](#)
- » [PythonLabsPython](#): an old name for the python.org distribution
- » [PythonwarePython](#)
- » [StacklessPython](#)
- »  [Tiny Python](#) (archived link) - not to be confused with tinypy

See also:

- » [EmbeddedPython](#) for details of minimal or reduced size Python distributions and implementations.
- » The master list of [Python implementations](#).

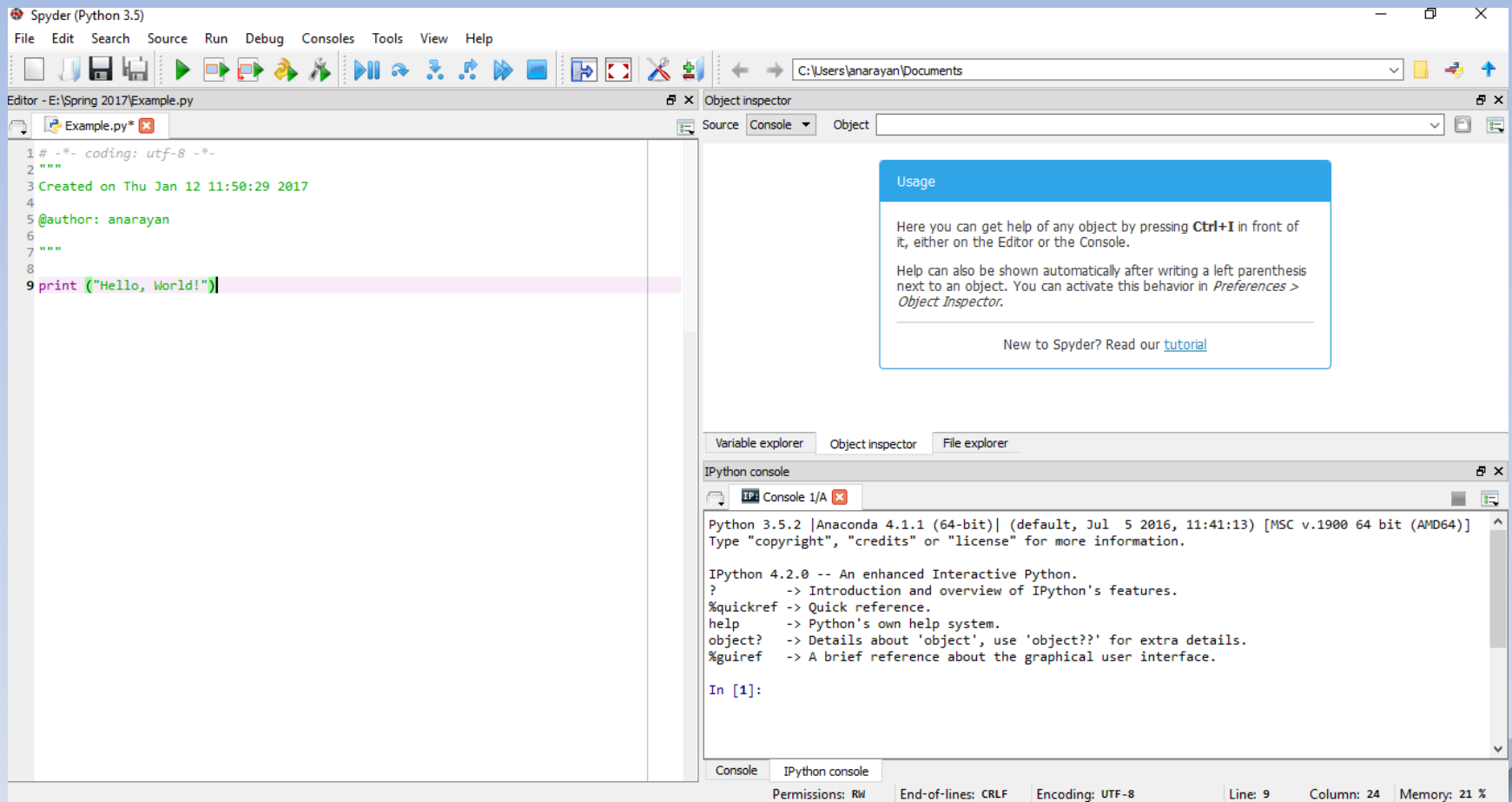
Anaconda Distribution

- **Anaconda** is a freemium (free for use but charges for proprietary features), open source distribution for Python and R programming languages
- Geared for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment.
- Package management system is called “conda”
- <https://www.anaconda.com/download/>

Spyder

- Spyder is the Scientific PYthon Development EnviRonment, is a free interactive development environment (IDE) that is included with Anaconda
- It includes editing, interactive testing, debugging and introspection features
- Depending on your installation Spyder can be invoked in various ways (Terminal/Command Prompt, Navigator, etc).

Python Spyder IDE (Anaconda)



How Python works behind the scenes

- Steps in interpreting a Python program

