CSC 280 Introduction to Computer Science: Programming with Python

Lecture 1

Prof. Bei Xiao
Fall, 2014
American University

Logistics

- Lectures: Mon, Wed & Thur, 1145am-1pm
- Personnel: Instructor: Bei Xiao

  TA: Alex Perepechko
- Bei: SCAN 110 Wed 4-5:30pm or by appointment.
- Alex: SCAN 160, Monday 3:30pm-4:30pm

Logistics

- Blackboard: Homework posting & submission, announcements, lecture notes, discussions, grading.
- Course webpage: course info, organization, reading assignments, lecture notes.
- http://nw08.american.edu/~bxiao/CSC280/CSC280_Fall2014.html

Grades

- Programming homework's: 50%
  - You are allowed to have one homework late but no more than 48 hours.
  - Any other late homework will be reduced 50% if submitted within 48 hours.
  - After that, zero credit.
  - Discussion is allowed in homework, but you must declare your collaborator.
  - Please follow Homework format to turn in your homework. Homework must be submitted via the Blackboard system. Submitting via email is not accepted.
- Two mid-term exams (open book, open source, but no discussion). One around mid October and Before Thanksgiving. 20%
- In-class quizzes: 10%. I will keep a note who answered questions.
- Final project: creative problem solving. 20%
- Cheating means copying lines of code. Only high level discussion is allowed.

Textbooks

- Textbooks:

  How to think like a computer scientist: learning Python, Allen Downey (required)

  Introduction to Computation and Programming Using Python, MIT Press, John Guttag.

- Tutorial:

Official Python Tutorial, by Guido van Rossum

Course Objectives

- Prepare students with little programming experiences to be able to write small and midsized code to solve problems.
- Being able to read other people's code and software.
- Being able to map a problem into **a computational framework**
- Prepare students for further engineering, scientific, and other technology courses
- Position students to compete successfully for interns and summer jobs.
- Having fun!

Course Outline (tentative)

- Python Basics (Sep- Mid October)
  - Syntax, Control Flow.
  - Basic Data Structure
- Object-Oriented and Functional Programming (mid October – early November)
  - Classes and objects
  - Inheritance
- Python for Algorithmic problem solving and other goodies (early November to mid December)
  - Simulation and Random Walk.
  - Data analysis and plotting
  - Recursion
  - Sorting, Searching, memorization
  - Basic Text Processing with Python
  - Python API

What is computer science?

- **Problem-solving**: puzzles, search, optimization, calculation beyond pencil and paper, storing information, retrieving information, tedious daily tasks one is too lazy to do, controlling robots….
- **Algorithm**: a step by step list of instructions for solving ANY INSTANCE of the problem might rise.
- **Abstraction: separate logical and physical perspectives**

Logical vs. Physical Perspective

- Driving the car
  - Logical perspective of the automobile
  - also called “interface”
  - User opereates

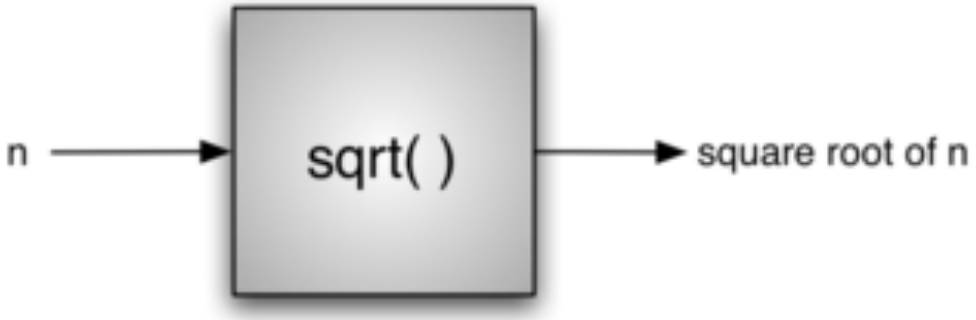
- Repairing the car
  - Physical perspective
  - Control the low-level details user simply assumes

Procedural Abstraction

- Compute square root:

- In Python:

```
>>> import math
>>> math.sqrt(16)
4.0
>>>
```



What is computation?

Declarative Knowledge

- Declarative knowledge is composed of statements of facts
- “A good health care plan improves the quality of medical care while saving money”.
- “DNA has double helix structure”.
- “ y is square root of x if and only if y*y = x”

Imperative Knowledge

- Imperative knowledge is about how to accomplish something. Think of it as a recipe.
- How to compute square root:

  1) start with a guess, g

  2) if g * g is close enough to x, then g is a good approximation of the square root of x.

  3) Otherwise, create a new guess by averaging g and x/g. I.e. g_new = (g_old + x/g_old)/2

  4) Using the new guess, **go back to step 2**

Square Root

- initial guess g = 3
- 3 * 3 = g square root of 25
- g = (3+25/3)/2 = 5.66
- g * g = 32.04
- g = 5.04……. 25.4….close enough to 25, we are done.

What is algorithm

- Algorithm- How to perform a computation
- The algorithm has converged.
- How did we get here?
  - Set of instructions
  - A flow of control (the order of executing)
  - Termination condition (when to stop)

Initial computer: fixed program computer

- Designed to do very specific thing.

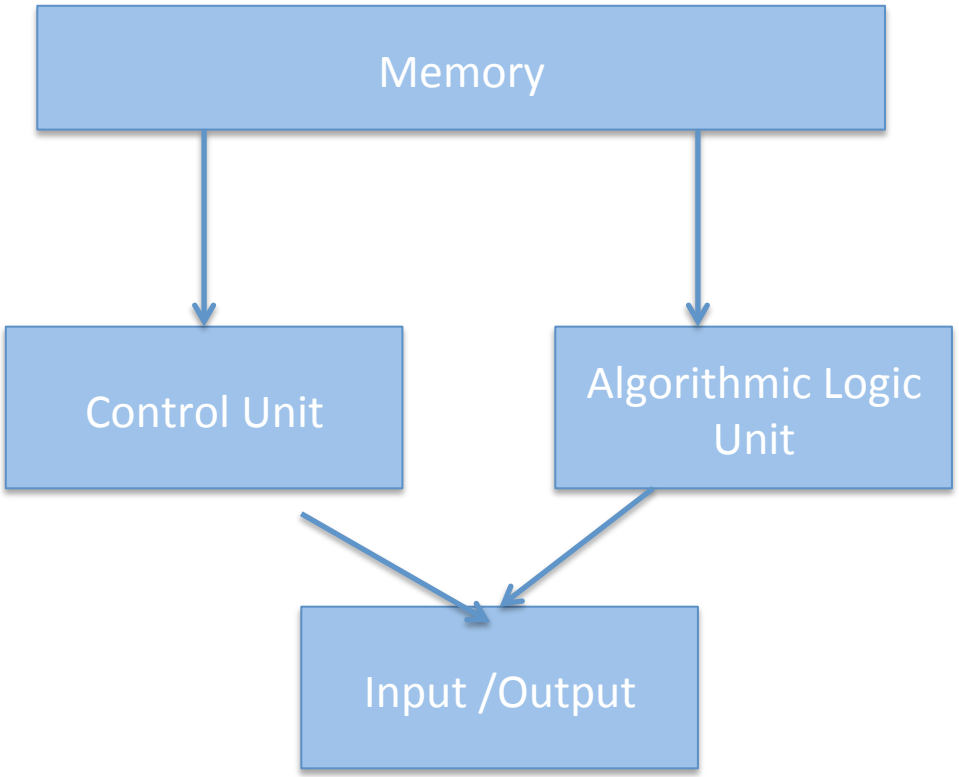
- First computer (1941) solve system of linear equations.

Stored Program Computer

- Treat data and instructions as the same thing
- No distinction between the program and the data the program operates.
- Program could produce program.
- Extremely flexible

Stored Program Computer

Treat data and instructions as the same thing



What is programming language?

- Combining small number of instructions, you can create complex tasks

- A programming language provides a set of primitive instructions and a set of primitive control structures.

What is different between programming languages?

- Set of instructions
- A flow of control (the order of executing)
- How to combine them?

Amazing thing about programming

- Computer is always doing WHATEVER you ask them to do.
- Annoying: if your program doesn't work, it is your own fault.

Syntax, Static Semantics, Semantics

- Syntax: which sequences of characters and symbols constitute a well-formed string

  e.g. x= 3+4 correct

  x= 3 , 4 not correct

- Static Semantics: which well-formed strings have a meaning.

  3/"abc" syntactically fine, but not meaningful. value operator value. but no real meaning.

- Semantics: what that meaning is.

  Both syntactically correct and semantically correct

What might happen if the program doesn't do what we want it to do?

- It might crash. Stop running.
- Never stopping. Infinite loop.
- Run to completion but produce wrong answer

We will learn how to avoid this from happening

Some programming provides strict static semantics

- Filtering out mistakes for you.

  e.g. Python doesn't allow you to do 3/"abc"

  Java is very good at this.

  But Python is better than C.

Why Python?

- Because it is easy and great fun.
  - A wide-range of applications, esp. in AI, scientific research (life science), data science, and web
  - Easy to learn.
  - Fast to write (shorter code than C, C++, Java)
  - Easy to read (more English-like syntax)
  - Easily transferable skills.
  - Easier to debug

Compiled vs. Interpreted

- Interpreted (easier to learn):

  source code (you wrote) -> checker->interpreter->output

- Compiled (more efficient):

  source code (you wrote) -> checker/compiler->object code (language close to computer hardware knows) -> interpreter by hardware->output

  Error message will be in object code

On to Python

“hello world”

- C

```
#include <stdio.h>
int main(int argc, char ** argv)
{
printf(“Hello, World!\n”);
}
```

- Java

```
public class Hello
{
public static void main(String argv[])
{
! ! System.out.println(“Hello, World!”);
}
}
```

- Python

```
print “Hello, World!”
```

Python is

- A scripting language (strong in text-processing)
- An interpreted language, like Perl.
- A very high-level language (close to human semantics).
- Procedural (like C, Pascal, but much more)
- But also object-oriented (like C++ Java).
- And even functional

Three ways to run a Python program

- Interactive

```
>>> for i in range(5):
    print i,
    0 1 2 3 4
```

- Save to a file, say foo.py
  in command line: foo.py
- Add a special line pointing to the default interpreter:
  add #!/usr/bin/env python to the top of foo.py
  make foo.py executable (chmod +x foo.py)
  in the command-line: ./foo.py

Quiz

What is the advantage of stored-program computer?

What sorts of errors can occur in a program?

What is syntax, static semantics, semantics?

Let’s get started

if IDEL is not installed, please use

http://pythonfiddle.com/

Values

- Numbers

  3

  3.14

- Strings

  “hello, world”

  “John”

  “ 3.1415”

Types

- type(3) int
- type(3.14) float
- type(“hello,world!”)

```
>>> type("hello,world!")
<type 'str'>
>>> type(17)
<type 'int'>
>>> type(100.0)
<type 'float'>
```

Operators & Operands

- +, *, -, /,**
- 1+2 addition
- 2*3 multiplication
- 3**5 power
- 3/5 division

```
>>> 3/5
0
```

?????? What happened?

Operators & Operands

- +, *, -, /,**
- 1+2 addition
- 2*3 multiplication
- 3**5 power
- 3/5 division

```
>>> 3/5
0
```

?????? What happened? To get float number, use float.

```
>>> 3.0/5
0.6
```

Variables

- myString = “bei”
- myString = myString + “xiao”
- print myString

```
>>> myString = 'bei';
>>> myString
'bei'
>>> myString = myString + 'xiao'
>>> mySTring
```