**What is Computer Science?**

This is from http://undergrad.cs.umd.edu/what-computer-science

Computer Science is the study of computers and computational systems. Unlike electrical and computer engineers, computer scientists deal mostly with software and software systems; this includes their theory, design, development, and application.

Principal areas of study within Computer Science include artificial intelligence, computer systems and networks, security, database systems, human computer interaction, vision and graphics, numerical analysis, programming languages, software engineering, bioinformatics and theories of computing.

Although knowing how to program is essential to the study of computer science, it is only one element of the field. Computer scientists design and analyze algorithms to solve programs and study the performance of computer hardware and software. The problems that computer scientists encounter range from the theoretical -- determining what problems can be solved with computers and the complexity of the algorithms that solve them – to the tangible – designing applications that perform well on handheld devices, that are easy to use, and that uphold security measures.

**What is computer programming?**

The writing of computer code to represent a system, place, an entity. It’s the use of a language to model something.

**What is the relationship of programming to computer science?**

Programming is a central tool of the Computer Scientist – it’s something computer scientists do that enables them to do the rest of their job.

**What are the main tricks to learning how to program?**

Mastering abstraction!

Planning a strategy (algorithm)

**What is abstraction?**

Thinking in terms of representations of things – not the actual things themselves.

Thinking abstractly requires manipulating symbols, combining them to achieve some purpose or to create a representation or model of something.

**Examples of Abstraction**

* Words
* Numbers
* Descriptions
* Word-problems in high school
* Calculating someone’s pay check

**What is an algorithm?**

Step-by-step plan or recipe for solving a problem. Specifically, it is the detailed process for implementing a model or representation. It’s how we’re going to solve the problem to achieve an outcome.

**Example:**

Describe how to compute somebody’s paycheck based on their hourly pay and hours worked.

find out how much they are paid by hour

find out how many hours they worked

calculate paycheck as hours worked \* pay per hour

report the paycheck amount

**Example:**

Describe how you find a name in a list of names.

**Example:**

Describe how you compute a temperature from Fahrenheit to Celsius.

**Example:**

Describe how you find the average of a set of numbers entered by the user.

**Coming up with an algorithm usually involves first identifying …**

* What you need to know
* What you need to show

Once you’ve done that, that’s when you can write the algorithm. The algorithm is what you need to **do** to convert what you know to what you need to show.

**Example:**

Write an algorithm to determine the mpg (miles per gallon) for someone traveling over a 5-day workweek.

**Example:**

Write an algorithm to determine if a person will ever run out of money in their checking account over the course of a year.

**Once you have an algorithm**, you can start writing the code.

**How do you write code?**

You use a programming language

**Types of programming languages**

* Low-level: the language of the computer. Just 1’s and 0’s (or mnemonics that are abbreviations for the 1’s and 0’s)
* High-level: more natural-language-like.

**Examples of High-Level Languages**

C, C++, BASIC, Visual Basic, FORTRAN, Pascal, Java, Javascript, PHP, Ruby, Python

**These are called Turing-Complete Languages**

* A Turing-Complete language is one that can be used to solve any problem that actually can be solved.
* Named after Alan Turing, a British mathematician and computer scientist, who devised the Turing Test for AI as well as the Turing Machine
* HTML is not a Turing-Complete language, but the ones listed above are.

**Python**

* A high-level language that was designed to be easy-to-learn
* Created by a physicist named Guido von Rossum in 1991
* Comes in two different versions now – 2.7 and 3.6
* We will use 3.6

**We will use Version 3.6 of the language. DO NOT USE 2.7!!!!**

**Downloading Python – 2 options**

* <https://www.python.org/downloads/>
  + The “stock” version
  + Comes with an editor named Idle
* <https://www.continuum.io/downloads>
  + Comes with more bells and whistles
  + Comes with an editor named Spyder

Either one will work for this course – but I think the continuum option is better

**Example:** Write a program to calculate someone’s pay, using the algorithm we wrote above.

# Ray Klump – a program to calculate someone’s gross pay

pay\_rate = 56.50

hours\_worked = 47

gross\_pay = pay\_rate \* hours\_worked

print("Your gross pay was $%.2f." % gross\_pay)

The %.2f in the quotes means “place a floating point number here with two digits after the decimal point.” The “%” means “I’m a placeholder”. The “f” means “floating point” (or decimal number). The “.2” means “2 digits after the decimal point.

Note the use of the # in the first line of this code. A hashtag introduces a comment – text the compute will ignore and not try to interpret as it reads your program and tries to execute its statements.