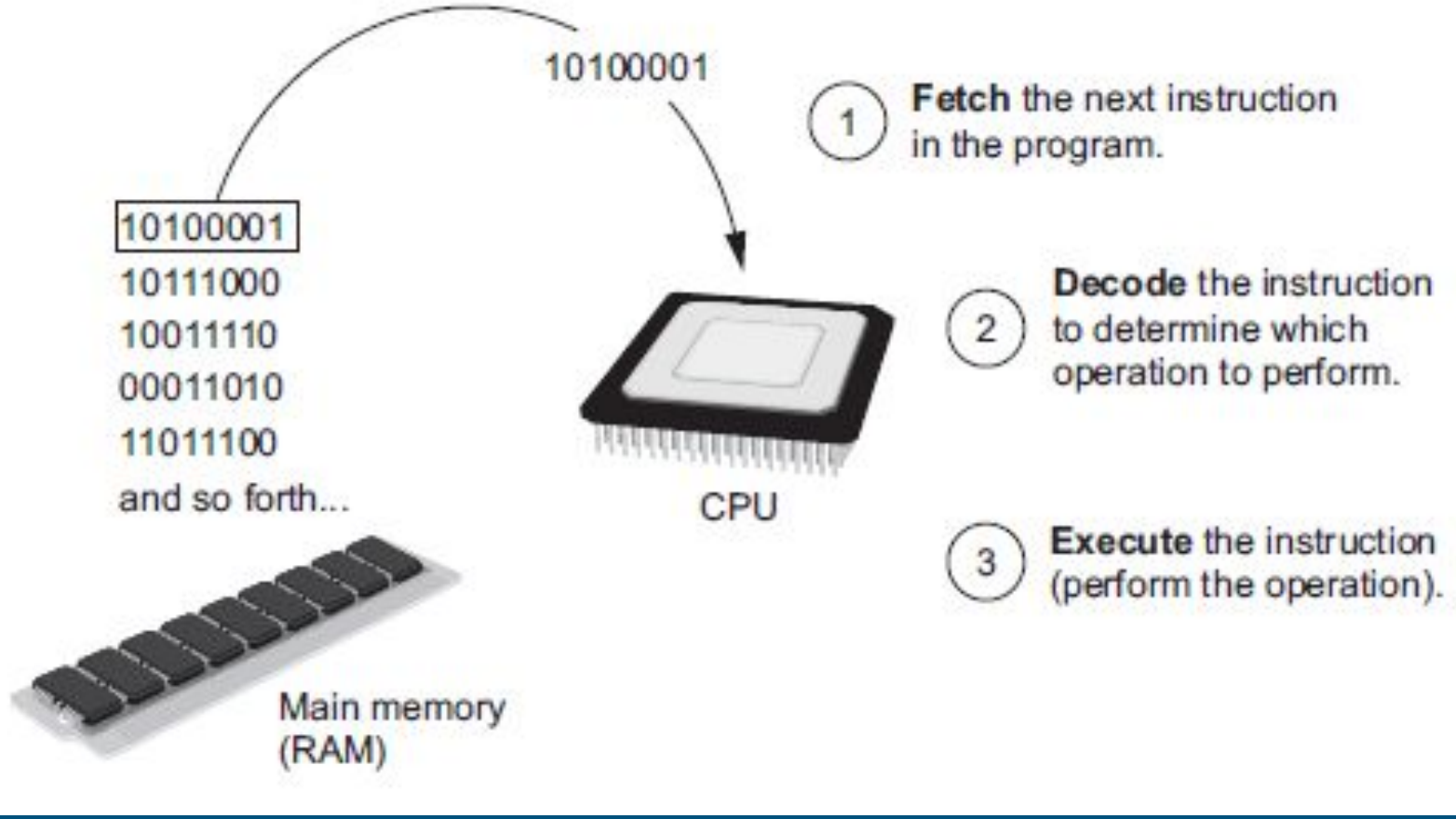
# Input, Processing and Output

CTEC 102

# Hey Kid, I’m a Computer


10100001
10100001
10111000
10011110
00011010
11011100
and so forth...
Main memory
(RAM)
CPU
1
Fetch the next instruction in the program.
2
Decode the instruction to determine which operation to perform.
3
Execute the instruction (perform the operation).

# High-Level Languages

- Low-Level Language
  - Assembly
- High-Level Language
- Compiled languages vs. Interpreted languages

# Special Terms

- Key Words - predefined words in a programming language
- Operators - performs operations on data
  - ‘+’
- Syntax - set of rules to be followed when writing a program
- Statement - an individual instruction

# Python

# Getting Python

https://www.python.org/

# Editing Code

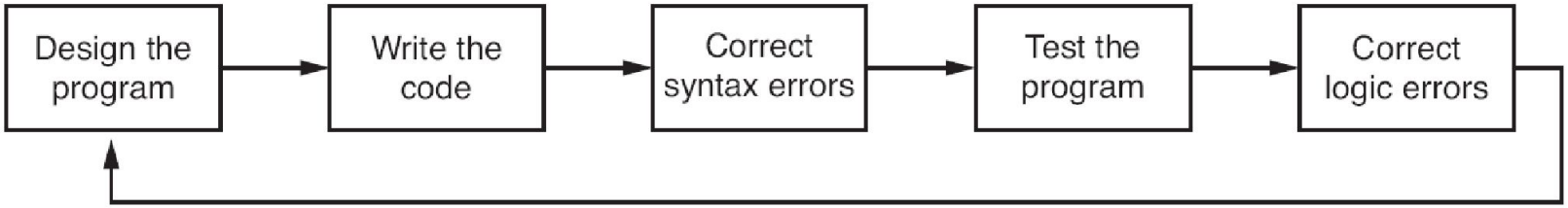
- IDLE
- Sublime
- Visual Studio Code
- Atom
- Notepad++
- Notepad

But not Word.

# Designing a Program

# Program Development Cycle

**Figure 2-1** The program development cycle



# Design Process

- Understand the Task
  - Software Requirements
  - “Interview the Customer”
- Determine the Steps
  - Algorithms
- Write it Out
  - Pseudocode
  - Flowcharts

# Let’s Talk Code

# Code

- Syntax
  - New lines are new instructions
  - Spacing matters
- Flow
  - Top to Bottom

# Functions

- Blocks of prewritten code that perform a task
- Can be named and reused
- Can take some input and return some output
- Where you'll find them:
  - Python includes some built-in functions
  - You can use functions (and other stuff) built by other programmers
  - You can write your own!
- Parts of a Function:
  - Definition
  - Arguments
  - Return Statement

# print()

- Hello World - the immortal programming ritual

# Let’s print() More!

- print(“I can print”,”more than one thing!”)
- print() can take multiple arguments
- When a function takes multiple arguments, each argument is separated by a comma (,)

Notes on print() specifically:

- Each argument is displayed in the order they are passed to the function
- Items are automatically separated by a space when displayed on screen

# Comments

- Ignored by the Interpreter
- Useful for humans
  - Provides a framework for you to organize your code
  - Lets other people read your code more easily
- Python Comments
  - # begin with this thing
  - Can begin on their own line or after a line of code
    - But not inside a string literal (don’t worry, we’ll get to that)

# Variables

- Data Types
  - int
  - float
  - str
- Static vs. Dynamic
- Buckets

# Assignment

variable = expression

- Your variable is on the left
- What you want to put in it is on the right
- Examples
  - age = 22
  - name = ‘Toby’
  - x = 5 + 4
    - Getting fancy there...

# Naming Your Variables

- Cannot be a Python keyword
- Cannot contain spaces
- First letter must be a letter or an underscore (_)
- Other letters can be letters, digits or underscores
- Variables are case sensitive
- Names should reflect use

# Camels, Snakes and Blaise Pascal

- camelCase
- snake_case
- PascalCase

*“I am not important to this slide”*

- Blaise Pascal

# Variable Reassignment

a = a + b

- Variables can reference different values while a program is running
- Garbage Collection - removes unreferenced stuff
- Type Assignment - variables can be switched from one type to another in Python
    - This can be different language to language

# Literals

- Basically the free-floating values in your code
  - “I am a string literal”
  - 23
  - True
  - 7.5
- You can assign literals to variables, use them in expressions and use them as function arguments
  - Usually where you can use a variable, you can use a literal

# input()

A function that allows you to get input from the user.

result = input(prompt)

Example:
name = input(‘What is your name?’)

# Reading Numbers with input()

- input() always returns a string
- Other built-in functions can convert between data types
  - int(item) converts item to an int
  - float(item) converts item to a float

# Nested Function Calls

- Many functions return values
- In your code, you can think of a value-returning function call being the same thing and replaceable by the value it returns
- We can pass values into functions
- Thus, we can use a Function call as an argument in a Function call

function1(function2(argument))

int(input(“Give me a whole number”))

# Calculations

- Operator - tool for performing calculation
- Operands - values surrounding operator
- Resulting value typically assigned to a variable

**Table 2-3** Python math operators

| Symbol | Operation | Description |
|---|---|---|
| + | Addition | Adds two numbers |
| - | Subtraction | Subtracts one number from another |
| * | Multiplication | Multiplies one number by another |
| / | Division | Divides one number by another and gives the result as a floating-point number |
| // | Integer division | Divides one number by another and gives the result as an integer |
| % | Remainder | Divides one number by another and gives the remainder |
| ** | Exponent | Raises a number to a power |

I guess let’s take a minute to talk about %

# Operator Precedence

- Please Excuse My Dear Aunt Sally
  - ()
  - **
  - *, /, // and %
  - + and -
- All things being equal, we go left to right

# Data Conversion

- Data type resulting from math operation depends on data types of operands:
  - int * int = int
  - float * float = float
  - float * int = float
    - Mixed-type expression
    - int is temporarily converted to float

# Implicit and Explicit Conversions

- Implicit - usually occurs when changing to a “wider” data type
  - int to float
- Explicit - required when changing to a more “narrow” data type
  - float to int
- Conversions must be explicit when data can be lost or misconstrued
  - int(2.7) equals 2

# More print() Tricks

- print() creates a newline by default
  - Can be changed with the end='delimiter' argument
- print() uses space to separate multiple printed arguments
  - Can be changed with the sep='delimiter' argument

# Escape Characters

**Table 2-8** Some of Python's escape characters

| Escape Character | Effect |
|---|---|
| `\n` | Causes output to be advanced to the next line. |
| `\t` | Causes output to skip over to the next horizontal tab position. |
| `\'` | Causes a single quote mark to be printed. |
| `\"` | Causes a double quote mark to be printed. |
| `\\` | Causes a backslash character to be printed. |

# String Concatenation

“These strings are “ + “being concatenated”

# format()

- Takes two arguments:
  - Numeric value to be formatted
  - Format specifier (a specially chosen string)
- Returns a string containing a special formatted number
- Format Specifiers:
  - ‘.2f’
  - ‘,.3f’
  - ‘,d’
  - ‘.0%’
  - Way too much information: https://docs.python.org/3.7/library/string.html#formatspec

# Let’s Do Some Stuff

# HeresMe.py

Create a Python script that

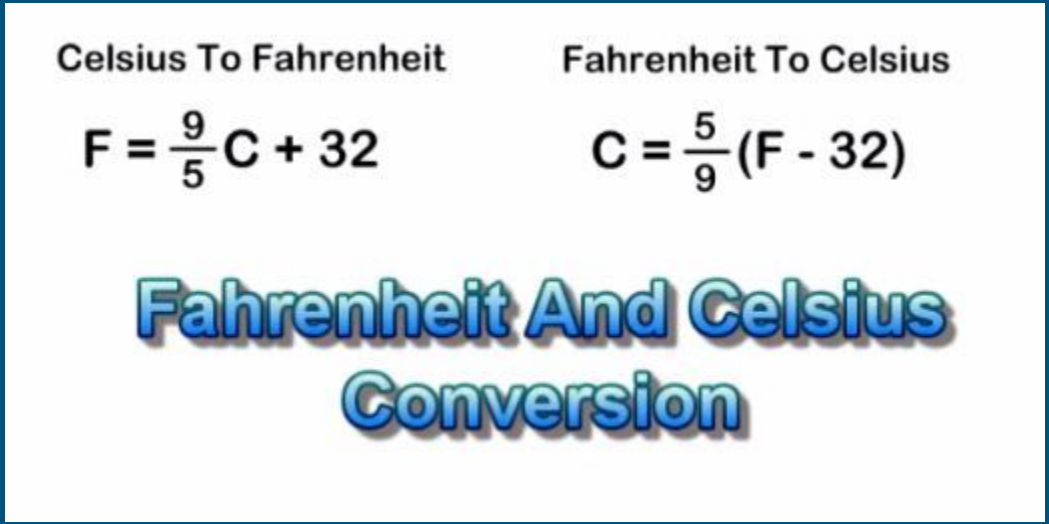
- Displays your name
- Displays a short introduction of what you’re hoping to learn in this course

# NumberAdder.py

Create a python script that asks the user for two numbers, then displays what those two numbers are when added together.

# Celsius to Fahrenheit Temperature Converter

Write a program that converts Celsius temperatures to Fahrenheit temperatures.



# Research Time

The Python Language documentation can be found at:

https://docs.python.org/3/

A list of built-in functions can be found at:

https://docs.python.org/3/library/functions.html

# Research Challenges

- Build a Python script that asks you for your name and tells you the length of it
- Build a Python script that asks you for two numbers, then raise the first number to the power of the second
- Build a Python script that asks you for your name and tells you the letters that come earliest and latest in the alphabet
    - This one may be tough to deduce, so make some guesses and try stuff out