Programming

CPSC 217: Introduction to Computer Science for Multidisciplinary Studies I
Winter 2023

Jonathan Hudson, Ph.D.
Instructor
Department of Computer Science
University of Calgary

January 9, 2023

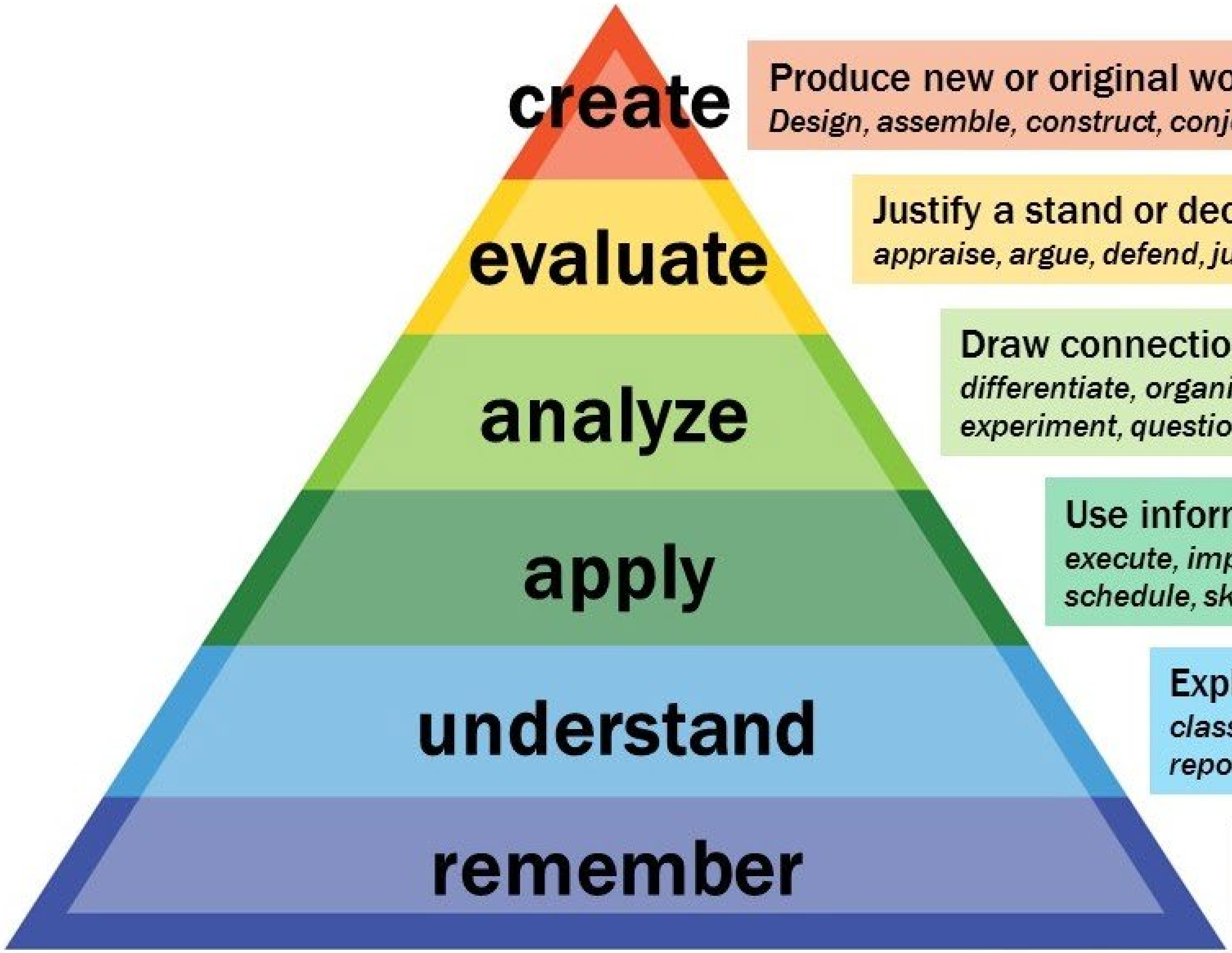
UNIVERSITY OF CALGARY

How Do We Solve Problems with a Computer?

First question: How do we learn?

What does it mean to understand something?

Bloom's Taxonomy



Produce new or original work
Design, assemble, construct, conjecture, develop, formulate, author, investigate

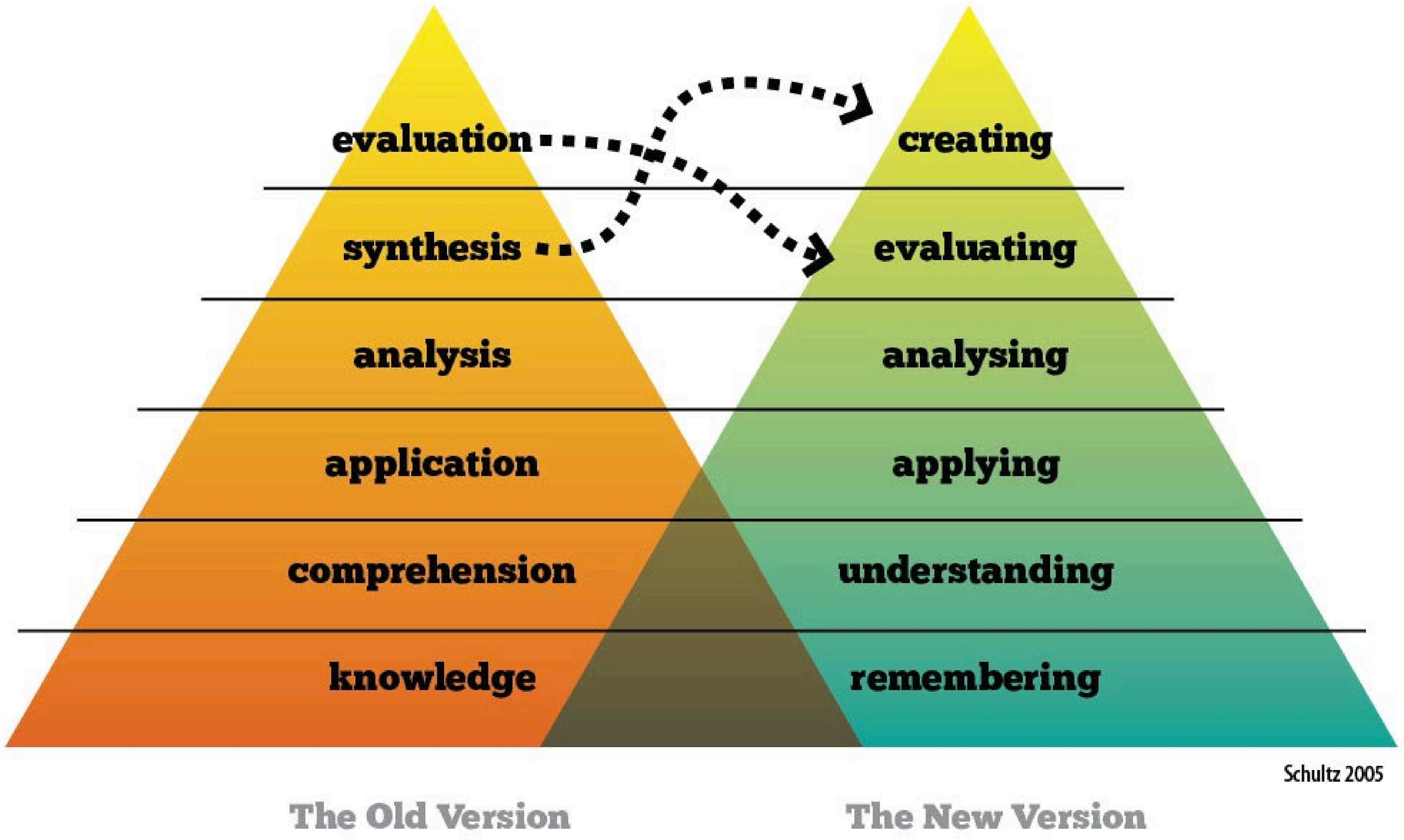
Justify a stand or decision
appraise, argue, defend, judge, select, support, value, critique, weigh

Draw connections among ideas
differentiate, organize, relate, compare, contrast, distinguish, examine, experiment, question, test

Use information in new situations
execute, implement, solve, use, demonstrate, interpret, operate, schedule, sketch

Explain ideas or concepts
classify, describe, discuss, explain, identify, locate, recognize, report, select, translate

Recall facts and basic concepts
define, duplicate, list, memorize, repeat, state


evaluation
synthesis
analysis
application
comprehension
knowledge
creating
evaluating
analysing
applying
understanding
remembering
Schultz 2005
The Old Version
The New Version

Solving Problems

How do we solve problems?

Solving Problems

How do we solve problems?

Break original problem into smaller, more easily solvable parts and repeat on the smaller sub-problems.

Similar process used in expository writing (break topic up into easily-understandable bits, express in coherent way).

Top Down Design

01

Start with the entire problem

02

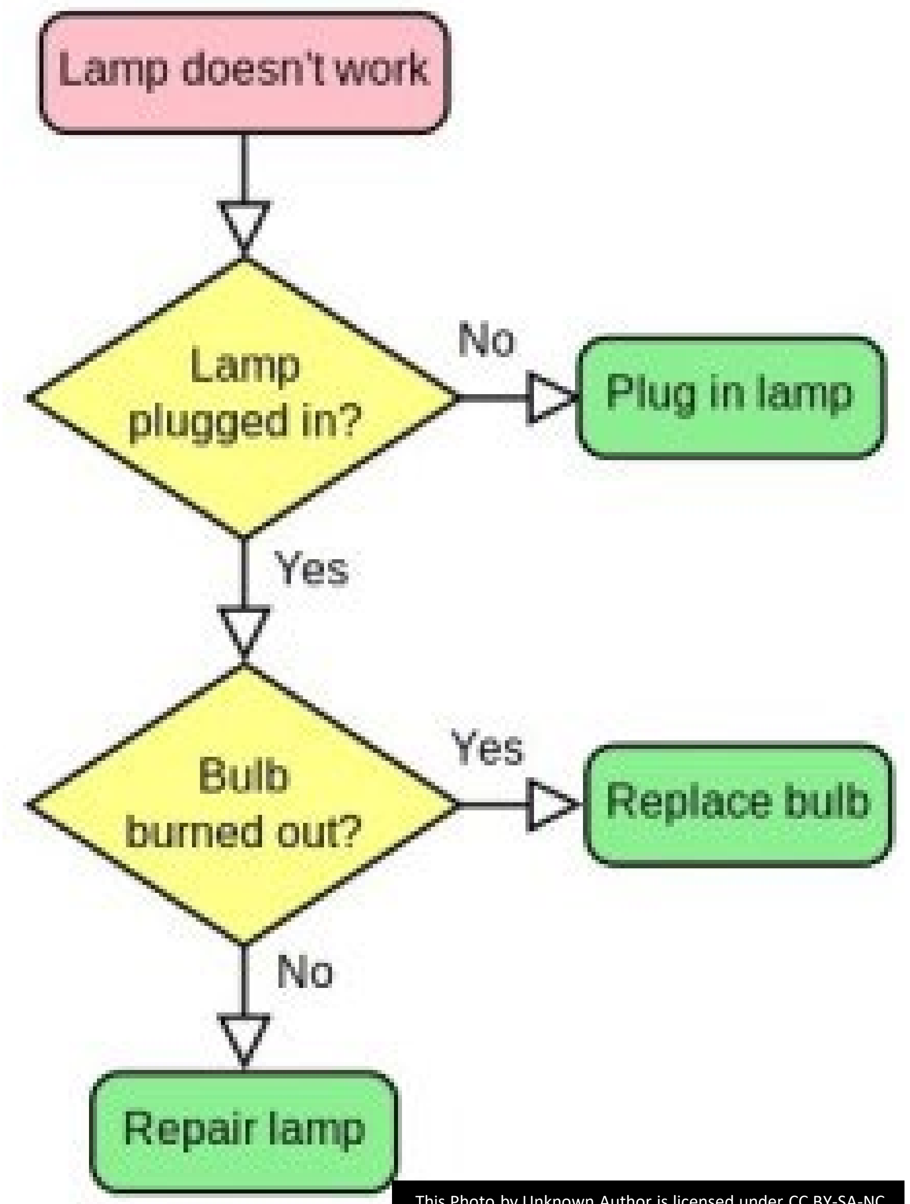
Break the problem into approximately 3 to 5 smaller steps

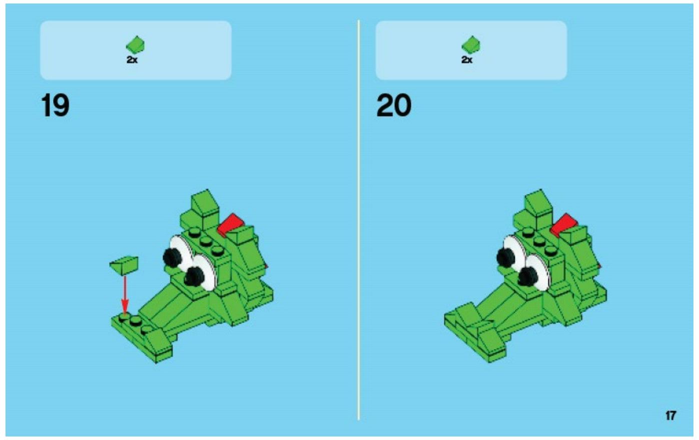
03

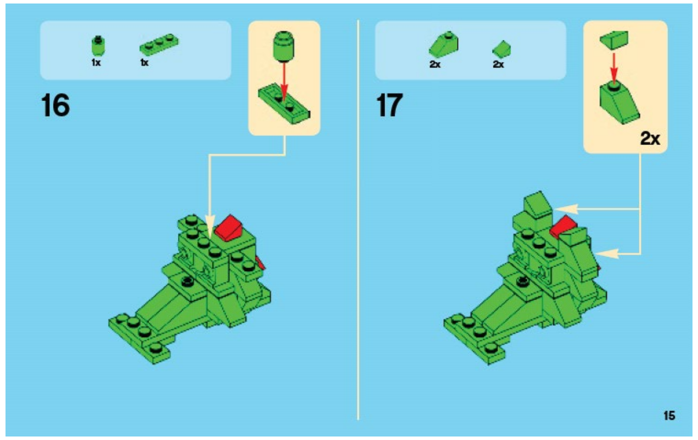
Repeat the process for each step that is still too complex

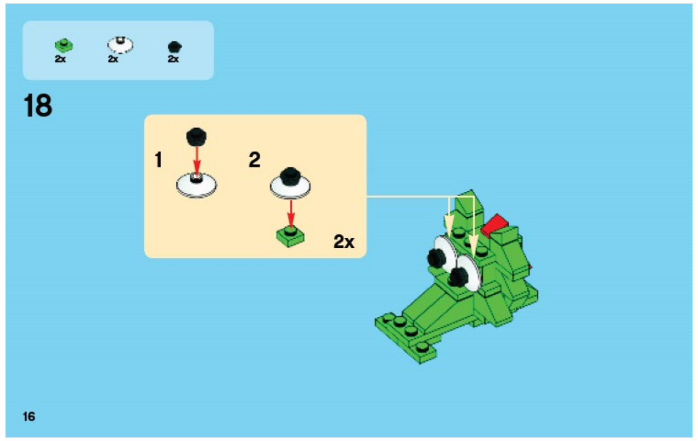
What is an Algorithm?

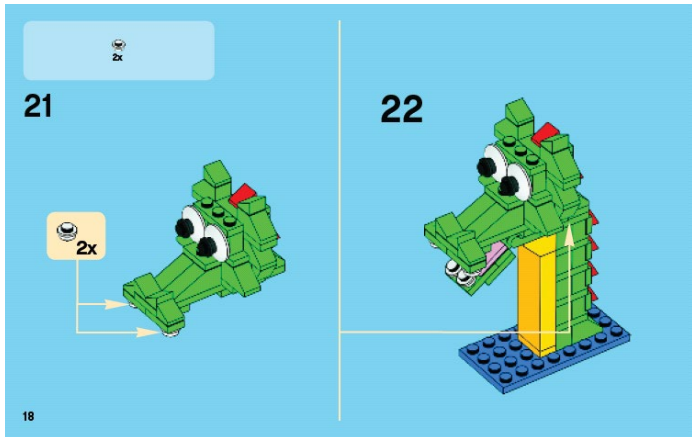
What is an Algorithm?




2x
2x
19
20
17


1x
1x
16
2x
2x
17
2x
15


2x
2x
2x
18
1
2
2x
16

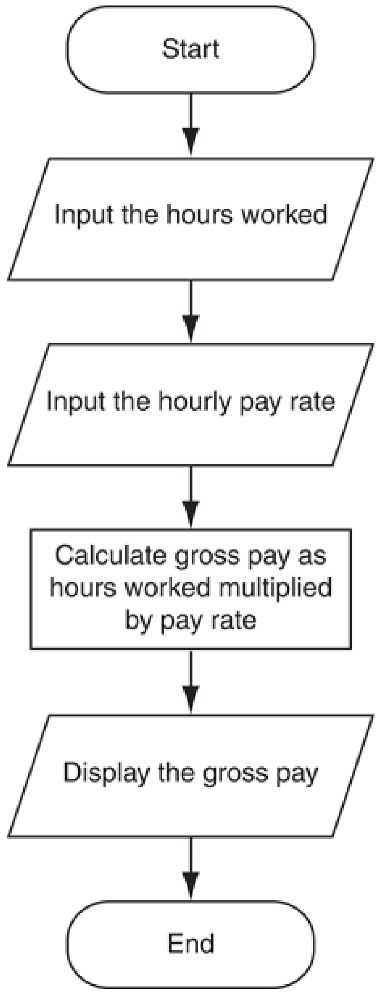

2x
21
2x
22
18

What is an Algorithm?

- Algorithm: A finite sequence of effective (unambiguous, possible) steps to solve a problem.
- Expressed in English, human-oriented form
- Result of top-down design (or other problem solving strategy)
- A well written algorithm can be written in ***any*** computer programming language
- Generally written in pseudo-code and/or visualized in a flowchart

Pseudocode

- Pseudocode: fake code
    - Informal language that has no syntax rule
    - Not meant to be compiled or executed
    - Used to create model program
        - No need to worry about syntax errors, can focus on program's design
        - Can be translated directly into actual code in any programming language
- Flowchart: diagram that graphically depicts the steps in a program
    - Ovals are terminal symbols
    - Parallelograms are input and output symbols
    - Rectangles are processing symbols
    - Symbols are connected by arrows that represent the flow of the program


Start
Input the hours worked
Input the hourly pay rate
Calculate gross pay as hours worked multiplied by pay rate
Display the gross pay
End

What is Programming?

What is Programming?

- Programming: the process of creating software by translating algorithms into a computer language.
- Algorithm: human readable form, layout/syntax is free as long as a reasonable person can understand it
- Computer Program: computer readable form, precise syntax that must be followed exactly, will do exactly what you say (not what you meant!) - SOFTWARE

Where Are We Going?

- Computers are tools that we use to solve problems
  - Need to understand the problem that we want to solve
  - Need to understand how a computer works to model the problem
  - Need to learn how to program the computer to solve the problem

Programming Languages

Many programming languages available

- Offer different features
- **Each has its own strengths and weaknesses**

Common features

- Allow us to control the behaviour of a computer
- **Defined syntactic and semantic rules**
  - **Syntactic** – what does a valid statement look like
  - **Semantic** – what is meaning of a statement

Example: Syntax versus Semantics

Python 2:

5 / 2 = 2

5 / 2.0 = 2.5

5.0 / 2 = 2.5

Python 3

5 / 2 = 2.5

5 / 2.0 = 2.5

5.0 / 2 = 2.5

Syntax is the same.

Division is done via /

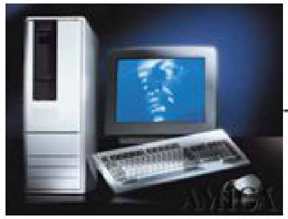
Semantics are different.

Python 2 division result is based on whether one input has decimals

One decimal value number creates decimals out

Python 3 always produces decimal numbers

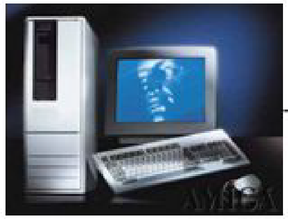
High Level Vs. Low Level Languages

Human languages	E.g., English, French, Spanish, Chinese, German, Arabic etc.
High level programming language	E.g., Python, Java, C++ for (i = 1; i <= 10; i++)
Low level programming language	Assembly MOV #10, R0
Machine language	Binary 10100000 1010 00
	Computer hardware

High level → **Ambiguity, Redundancy,..**

Low level → **Accuracy, Concise,…**

High Level Vs. Low Level Languages

Human languages	E.g., English, French, Spanish, Chinese, German, Arabic etc.
High level programming language	E.g., Python, Java, C++ for (i = 1; i <= 10; i++)
Low level programming language	Assembly MOV #10, R0
Machine language	Binary 10100000 1010 00
	Computer hardware

High level → **Ambiguity, Redundancy,..**

We will be writing in Python (a high-level language), to run source code of a program it is compiled to binary code (machine code) which the hardware stores as electrical information

Low level → **Accuracy, Concise,…**

James Tam

Using Python

- **Python 3 is the official programming language for this course.**
- There are different ways to tell Python to execute your code:
  - Interactive coding
  - System command lines and files

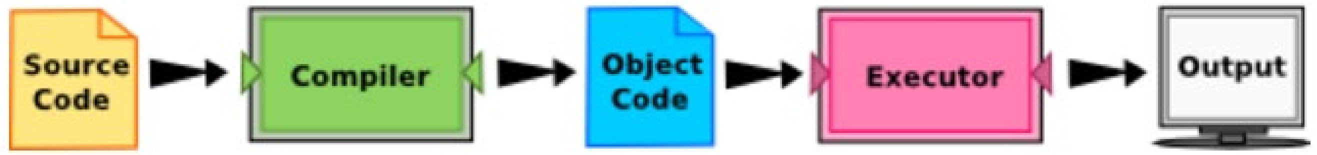
Compiler/Interpreter

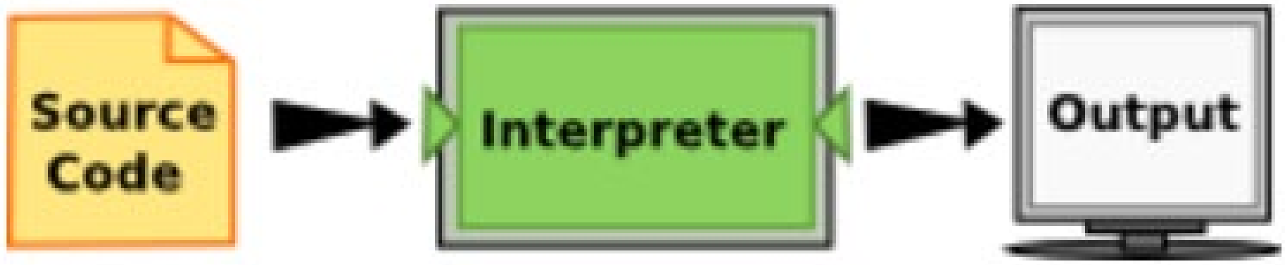
Programming

- Computer programs are stored in **source code files**
  - Human readable / editable
  - Can also be understood by a computer
  - typically have the extension **.py**
  - .py extension, or any extension is completely option in the computer, I could write a program in a file called **jingle.bells** and and it would work
- Once the file is created, it is run using the installed python program
  - **python myfile.py**
  - Also! **python jingle.bells**

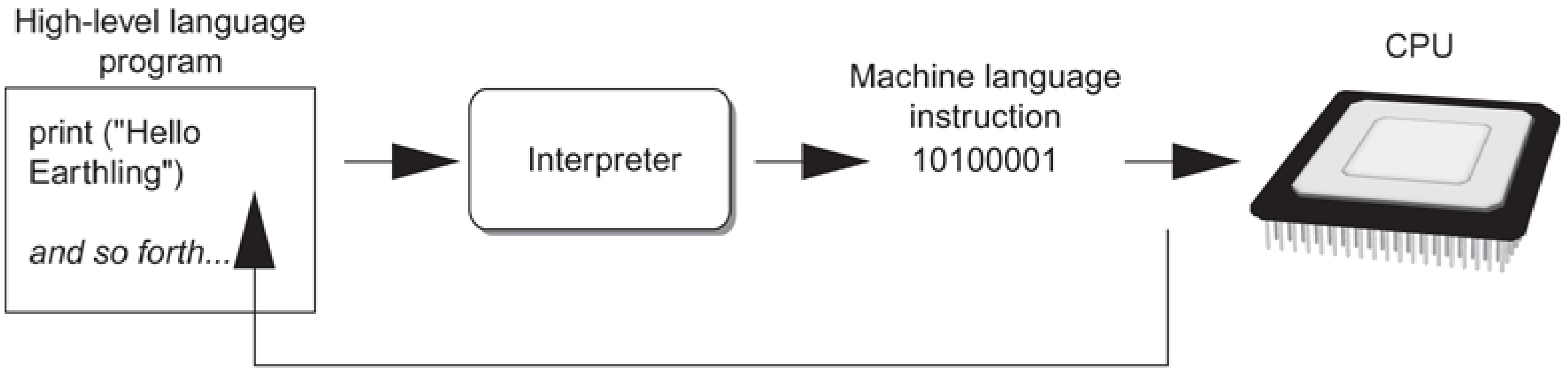
Compilers Vs. Interpreter

- A **compiler**:
  - Is like **translating an entire book** and give it to a reader.
  - A compiler reads the program and translates it completely before the program starts running
- An **interpreter**
  - is **like translating a line at a time** and give the line to the uses until the book is finished.
  - It processes the program a little at a time, alternately reading lines and performing computations.





Interpreting code

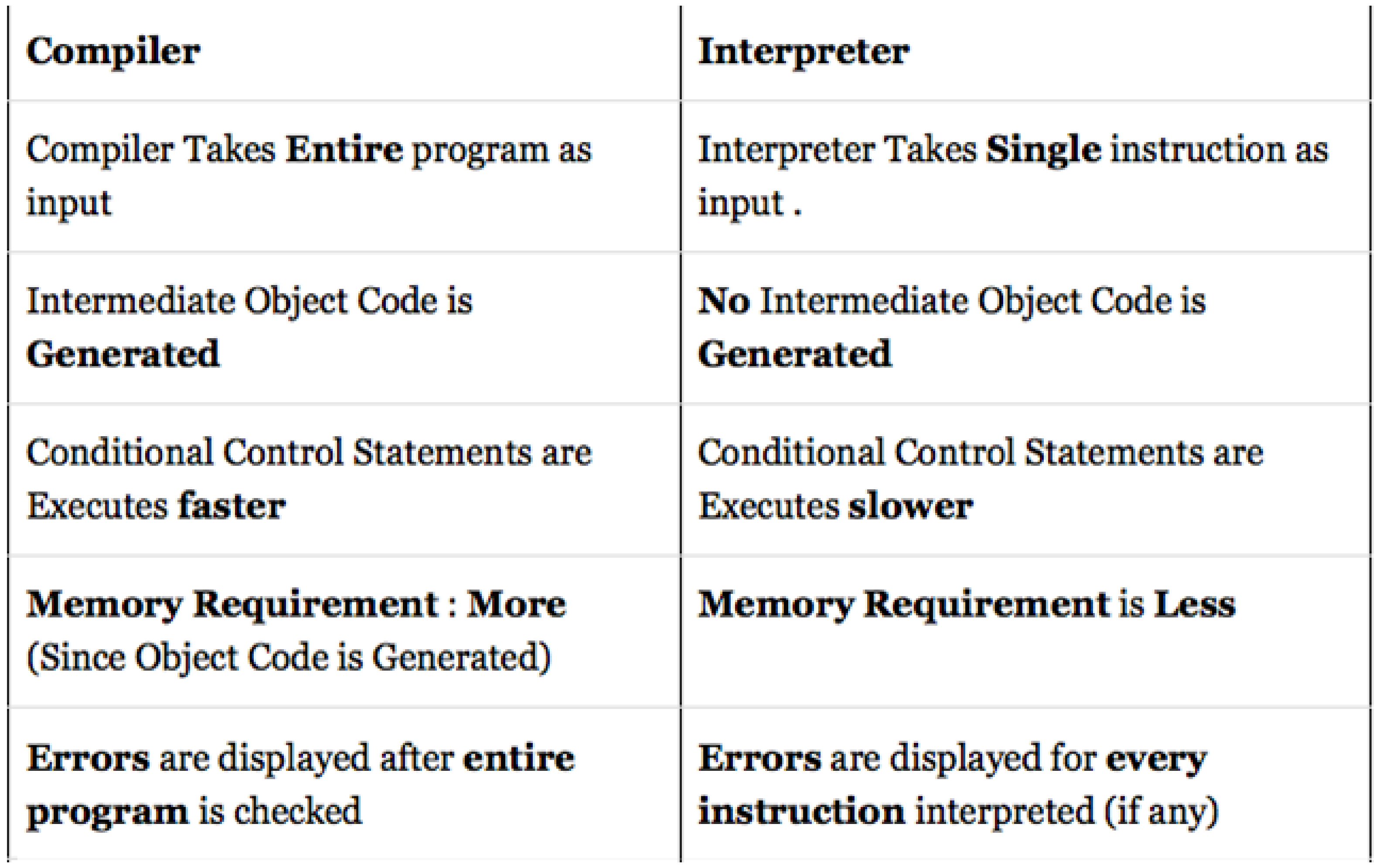

High-level language program
print ("Hello Earthling")
and so forth...
Interpreter
Machine language instruction
10100001
CPU

The interpreter translates each high-level instruction to its equivalent machine language instructions then immediately executes them.

This process is repeated for each high-level instruction.

UNIVERSITY OF CALGARY

Difference between Compiler and Interpreter



Compiler	**Interpreter**
Compiler Takes **Entire** program as input	Interpreter Takes **Single** instruction as input .
Intermediate Object Code is **Generated**	**No** Intermediate Object Code is **Generated**
Conditional Control Statements are Executes **faster**	Conditional Control Statements are Executes **slower**
Memory Requirement : **More** (Since Object Code is Generated)	**Memory Requirement** is **Less**
Errors are displayed after **entire program** is checked	**Errors** are displayed for **every instruction** interpreted (if any)

Compiler or Interpreter

- Many modern languages use both processes.
- Python uses both processes.
- Python is a **compiled interpreted language**

- Python is first compiled into a lower-level language, called **byte code**, and then interpreted by a program called a **virtual machine**.
  - (**byte code is higher level than machine code**, we can still move it between machine types Ex. Win/MacOS. Machine code can only move within same type.)
  - A **virtual machine** is created for each machine type.

- There is also an **interpreter-only** mode for Python where we can type instructions line by line. However, it is rare to use this outside of teaching programming.

Coding

Command lines and files

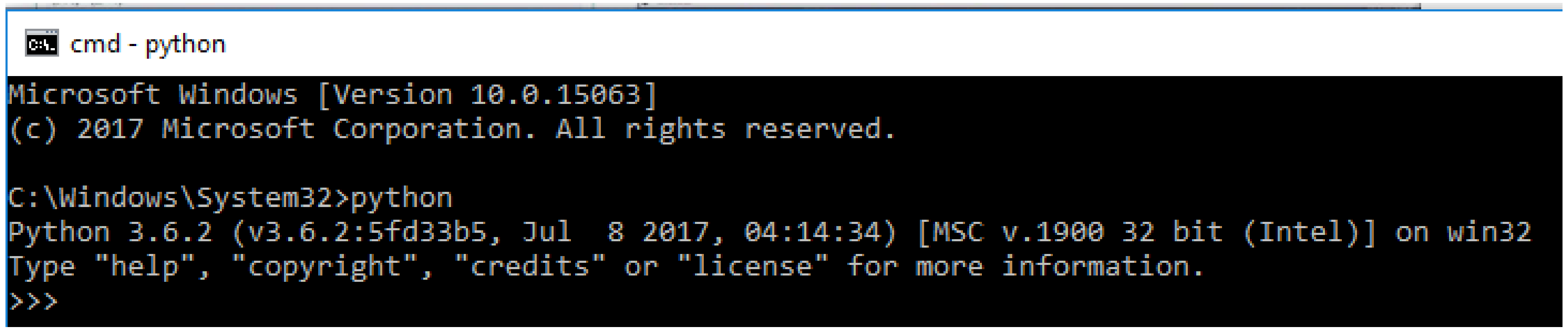
You can check your version using **python –V**

If the result is 2.X.X then

You will have to use: **python3 hello.py**

Interactive coding (Interpreter-only mode)

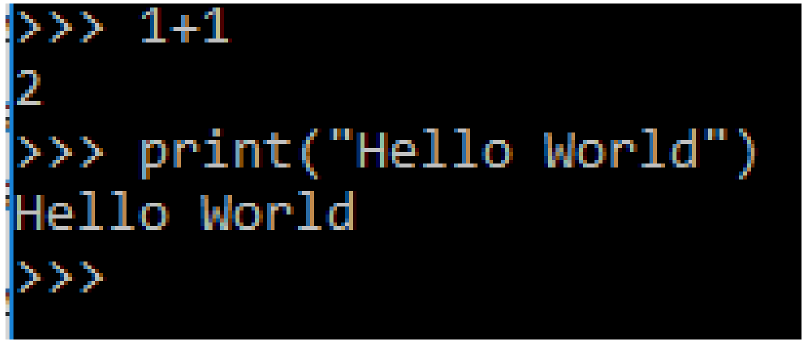
- >>> is a Python prompt indicating that python is ready to accept commands



cmd - python

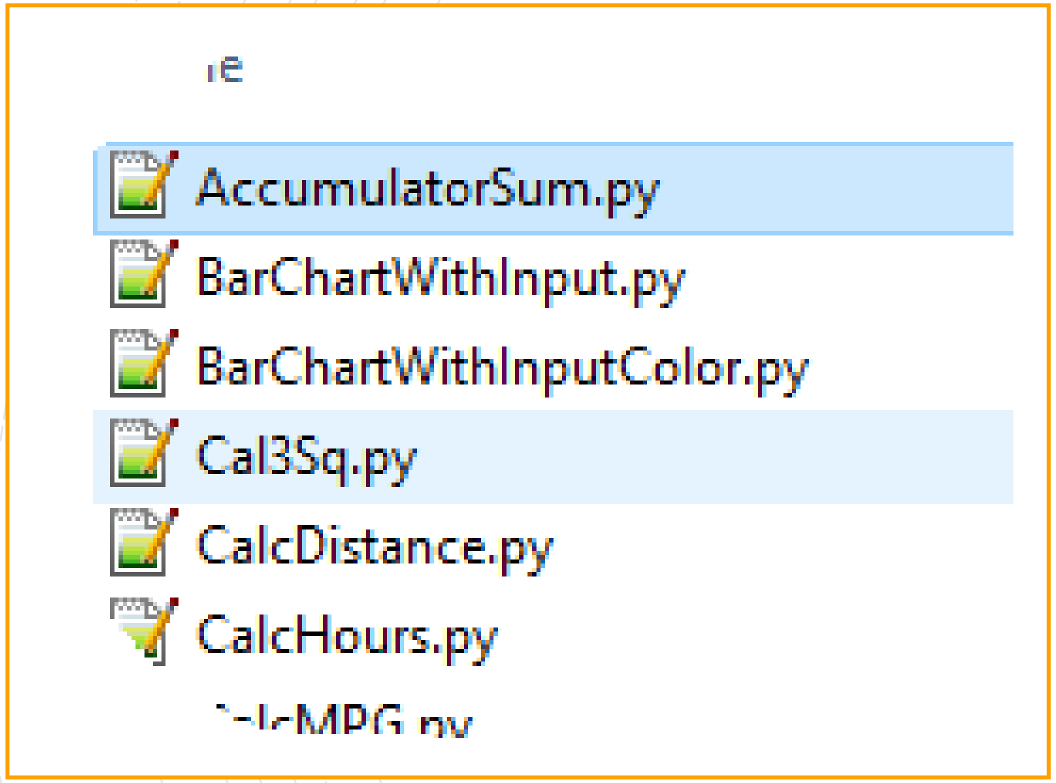
```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Windows\System32>python
Python 3.6.2 (v3.6.2:5fd33b5, Jul  8 2017, 04:14:34) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- Python Functions:
  - **print()** → prints text to the screen
  - **exit()** → exits python



```
>>> 1+1
2
>>> print("Hello World")
Hello World
>>>
```

Command Lines and Files


AccumulatorSum.py
BarChartWithInput.py
BarChartWithInputColor.py
Cal3Sq.py
CalcDistance.py
CalcHours.py

Source code **is a file containing your code often referred to as a *program*.**

- The filename ends with a *.py* suffix

e.g. **hello.py**

- To execute from terminal/shell:

python hello.py

- To save the output into a file:

python hello.py > output.txt

Onward to … variables!

Jonathan Hudson
jwhudson@ucalgary.ca
https://pages.cpsc.ucalgary.ca/~hudsonj/

UNIVERSITY OF CALGARY