Lab 1A: Hello World

ENGINEER 1P13: COMPUTING LAB 1A

Problem-Solving with Computing

McMaster University

Overview

- Intro to Computing
- Flowcharts
- Variables
- Data Types
- Calculations
- `print` & `input` statements

1P13 Learning Outcomes

- Upon successful completion of the course, the student should be able to:
  - LO.01 – Demonstrate understanding and application of graphics design principles
  - LO.02 – Demonstrate understanding and application of engineering computation principles
    - Apply principles of software development, design, and testing
    - Analyze a simple computer program
    - Create a computer program to satisfy a simple specification
  - LO.03 – Demonstrate an understanding of structure, properties, and applications of materials
  - LO.04 – Explain professional duties of an engineer as they relate to society
  - LO.05 – Demonstrate the ability of design thinking
  - LO.06 – Design a well-thought-out solution to a real-world problem
  - LO.07 – Demonstrate effective communication in a breadth of situations
  - LO.08 – Demonstrate effective teamwork on a design project
  - LO.09 – Reflect on past experiences and what has been learned from these experiences

Pre-Lab Checklist

Reminder!

- The following should be completed BEFORE today!!
  - Watch the Computing 1A Pre-lab Online Module (*.mp4)
  - Complete the *short* Avenue Quiz (based on online module)

You will **not** be able to access the Avenue Dropbox *until* you have completed ALL pre-lab requirements

Getting Started

- Download the following files from Avenue:
  - ❑ Computing 1a - Slides.pdf (these slides)
  - ❑ Computing 1a - Assignment.docx

1 Intro to Computing

1.1 Problem-Solving with Computing

- Computing gives us a systematic approach to problem solving
- Problem solving structure can be broken down into:
  - Inputs: information/data given for processing
  - Processes: manipulation of or operations on information/data
  - Decisions: actions made based on specific conditions
  - Outputs: results of processes/operations

1.1 Problem-Solving with Computing

- The general structure for transforming real world problems into computational problems is:
  - Understand the Problem
  - Formulate a model
  - Develop an algorithm (use flowchart)
  - Write the program
  - Test the program
  - Evaluate the solution

This week focuses on theses steps

1.2 Step 1: Understand the Problem

- Before attempting to solve the problem, an understanding of the problem must be developed
    - What information is available to you? (i.e., inputs)
    - What are you trying to accomplish?
    - What output(s) do you want?

1.2 Step 2: Formulate a model

- Understand any **process** or **processes** that need to be performed
- Questions to consider:
  - How do we get from our **inputs** to our **outputs**?
  - Are there any formulas we need to use?
  - Can we break the problem into smaller problems and solve those?

- Figure out how to use data available to achieve desired outcomes

1.2 Step 3: Develop An Algorithm

- The steps of a process or set of rules followed to perform a task is called an **algorithm**
- Representing an **algorithm** beforehand helps lay the foundation for the program
  - Separates the ideas from the actual implementation
- Two common representations for **algorithms**
  1. **Flowcharts**
  2. Pseudo Code
- This lab will focus on using **flowcharts**

1.3 Python

- Python is a programming language that allows us to write instructions for a computer to perform specific tasks
- Computers understand and execute instructions based on machine code
  - Machine code is a bunch of 1s and 0s
- Programming language instructions are translated into machine code through a compiler
  - This happens every time you run code
  - Computer reads machine code and executes it

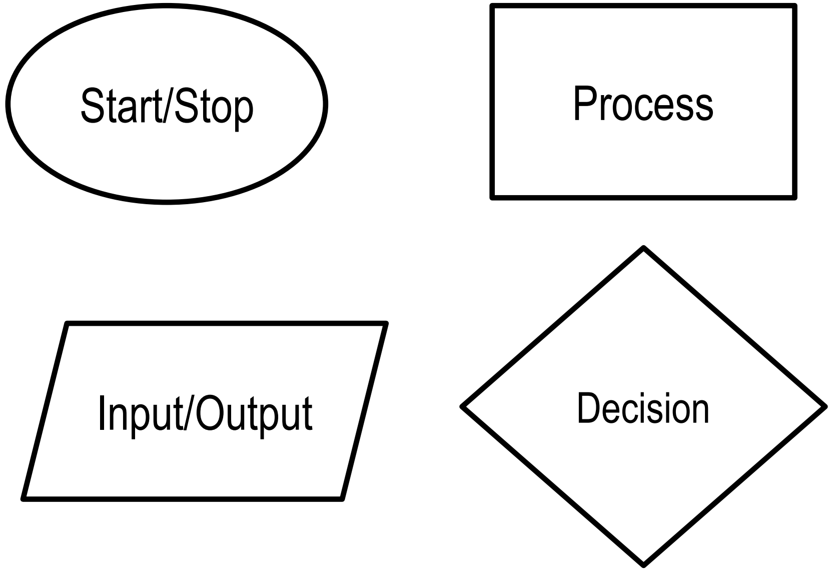
1.3 Python Syntax

- Like any other language, Python has rules to how instructions are written to be properly interpreted by the computer
- **Syntax** is essentially the spelling and grammar rules of a language
- **Syntax** errors can occur if you misspell an instruction or write something that doesn’t quite make sense to the computer
  - Other errors that you may encounter will be covered in a later lab

2 Flowcharts

2.1 Flowcharts

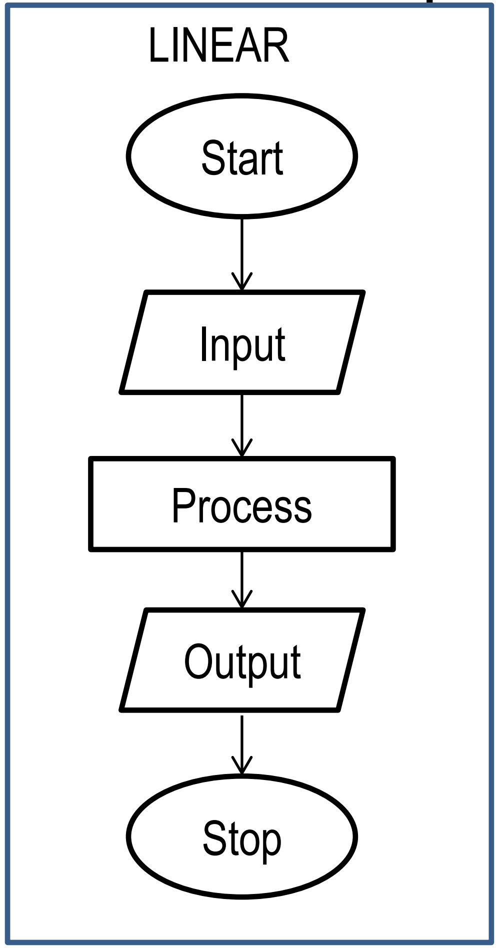
- Flowcharts: diagrams that lay out the steps of an algorithm or process
  - Helps to visualize the structure of the program
  - Represents the specific sequence of operations
- Specific shapes are used for different operations in a flowchart
  - Start/stop = oval/ellipse
  - Input/output = parallelogram
  - Process = rectangle
  - Decision = diamond

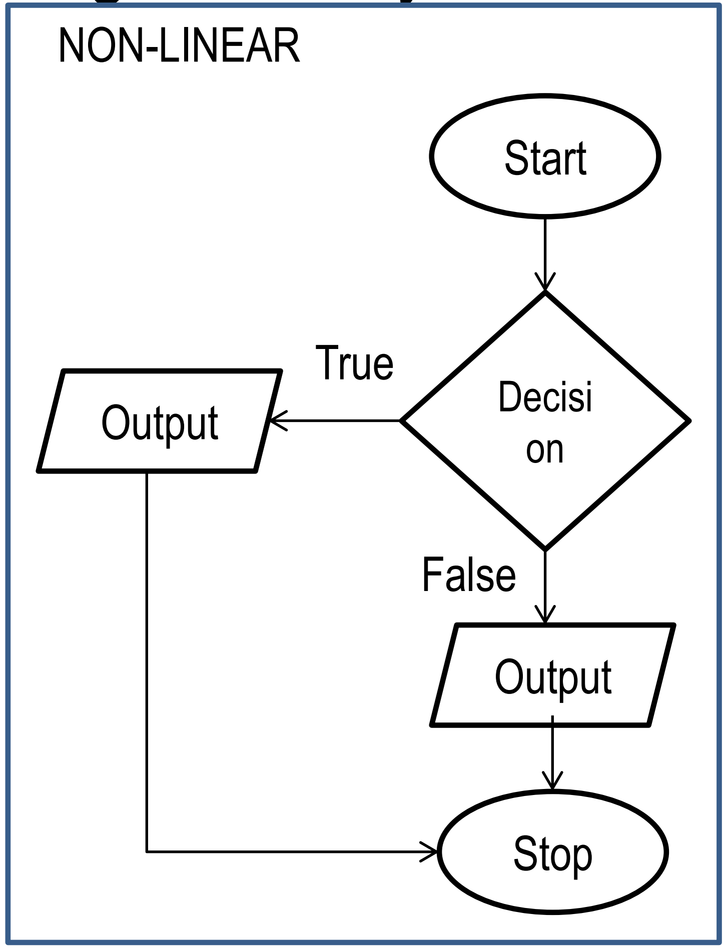


2.1 Flowcharts

- Flowcharts can be linear or non-linear depending on if they have branching paths

Not all flowcharts will look exactly this.
Flowcharts will look different depending on the algorithm they describe.
These are just some examples.





3 Variables

3.1 Values and Expressions

Explain the content below with the In-Lab Notebook open!

- Values: information or data stored and manipulated in computer programs

```
Examples:      123     99.9     “Hello”        True
```

- Expressions: a syntactic entity that represents a value
  - Simplifies into a value after evaluation
  - Usually contains operators

```
Examples:      x+5     3*2      13+1-9         102/9
```

3.2 Variables

Explain the content below with the In-Lab Notebook open!

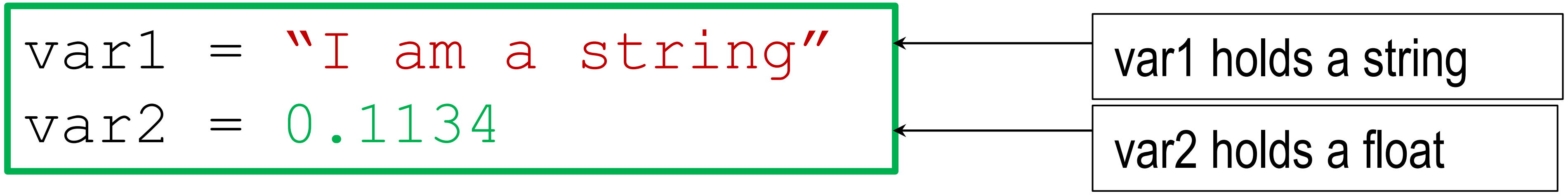
- Variables: containers for values or information
  - Values may change over the course of the program running
  - Data type of variable depends on the data it holds
- You assign values to variables using the assignment operator which is an equals sign
  - Assigning a value to a variable associates a name to a value or values

```
student_name = “John”
g = 9.81
```

3.2 Variables and Data Types

Explain the content below with the In-Lab Notebook open!

- Variables have 2 parts:
  1. Name
  2. Stored value or data
- Variable name is used to access data stored
- Stored values or data have different data types



```
var1 = “I am a string”
var2 = 0.1134
```

var1 holds a string

var2 holds a float

3.2 Variable Assignment

Explain the content below with the In-Lab Notebook open!

- After assigning values to variables, you can use the name of the variable to refer to the value

```
g = 9.81
mass = 2

force_grav = mass * g
```

You can also assign the result of a calculation to a variable

- You can also store multiple values under a single name
  - This is called a list

```
students = [“Sam”, “Bob”,“John”]
```

3.3 Variable Naming Convention

- Following naming convention helps to ensure that your variable names are descriptive, relevant, and syntactically correct
  - Can only contain alphanumeric (letters and numbers) characters and underscores
  - Cannot start with a number
  - Python keywords should be avoided

Good Examples

```
student_name
lab_section
```

Bad Examples

```
x
thing
stuff
```

For ENG1P13, we will be using a naming convention called snake case.

1. Variable names will be written in all lower case letters.
2. Words are separated by underscores.

4 Data Types

4.1 Data Types

- Each defined variable has a value and values have data types
- Data comes in different forms and types
- Each data type is used to represent a different kind of value:
  - Numeric (integers & floats)
  - Strings
  - Booleans
  - Lists

4.2 Integers & Floats

Explain the content below with the In-Lab Notebook open!

- Integers are used to represent positive and negative whole numbers

```
integer1 = 13
integer2 = 64
```

- Floating-point numbers (floats) are used to represent real numbers
  - Precision depends on computer

```
float1 = 1.13
float2 = 45.8
```

4.3 Strings

Explain the content below with the In-Lab Notebook open!

- Strings are a data type used to represent text
- Strings can be defined using single or double quotes
  - Try to stay consistent
  - Strings with apostrophes may cause problems if you use single quotes
    - To display an apostrophe, you can ‘escape’ the character by placing the \ character before (discussed more in a later lab)

```
string = “This is a string”
string2 = ‘This is a string too!’
```

String Operations

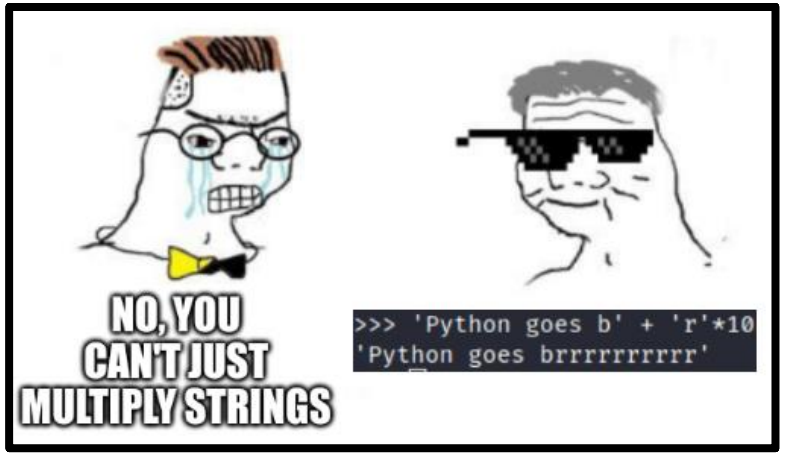
Explain the content below with the In-Lab Notebook open!

- You can add strings together using the addition symbol (plus sign)
  - This is called string concatenation

```
“I am a ” + “first-year student” becomes
“I am a first-year student”
```

- You can also multiply strings by integers to repeat them
  - NOTE: This is exclusive to Python

```
“ha” * 6 becomes “hahahahahaha”
```



```
>>> 'Python goes b' + 'r'*10
'Python goes brrrrrrrrrr'
```

4.4 Booleans

Explain the content below with the In-Lab Notebook open!

- Booleans are truth values used to make decisions in a program
  - Only two possible data values: `True` or `False`
- Similarly, Boolean expressions evaluate to either `True` or `False`
- Booleans will be studied in-depth further when non-linear programs are discussed

```
computing_lab = True
paying_attention = False
```

4.5 Lists

Explain the content below with the In-Lab Notebook open!

- Lists are structures used to organize/collect multiple pieces of data and/or variables under a single name
  - Elements in a list are called items
    - Each item corresponds to a number called an index
    - For a list of n items, the first item is index 0 and the last item is index (n – 1)
  - Lists can hold multiple types of data too!
    - Lists can even hold lists inside!

my_list = [0, 1, …]

- Lists are defined/declared similarly to variables

```
integer_list = [1, 3, 4, 10, 234]
string_list = [ “hi”, “bye”, “bonjour” ]
mixed_list = [5, “hello”, 293.0, “yup”, “word”]
```

Summary: Data Types

Data Type	Name	Usage	Example
`int`	Integer	positive and negative whole numbers	3, 5, 49
`float`	Floating Point	Real numbers	1.2, 231.1, 0.01
`str`	String	Sequence of Unicode characters	“Hello”, “String”
`Bool`	Boolean	Truth values	True, False

NOTE: there are more data types, but these are the ones we will most commonly use

Calculations

Explain the content below with the In-Lab Notebook open!

- Python can perform calculations using **mathematical operators**

Operation	Symbol	Description	Example	Result
Addition	+	Adds two value	2 + 3	5
Subtraction	-	Subtracts one value from another	9 - 7	2
Multiplication	*	Multiplies one value by another	12 * 5	60
Division	/	Divides one value by another and gives result as floating-point number	9 / 2	4.5
Remainder (Modulo)	%	Divides one value by another and gives remainder	19 % 2	1
Exponent	**	Raises a value to a power	2 ** 5	32

5 `print` & `input` statements

5.1 `print` statement

Explain the content below with the In-Lab Notebook open!

- `print()`: tells program to display what is within the parenthesis in the output

```
In:[ ]  print(“Welcome to ENG1P13”)
```

```
Out:[ ] Welcome to ENG1P13
```

- `print()` is a built-in function
  - Typing `print()` calls the function
  - The string input inside the parentheses is called the argument and is displayed in the output
- Arguments can also be a variable holding a value

```
In:[ ]  message = “Welcome to ENG1P13”
        print(message)
```

```
Out:[ ] Welcome to ENG1P13
```

5.1 `print` statement

Explain the content below with the In-Lab Notebook open!

- To display numeric values, you can use the comma in the parentheses to separate them from the message

In:[ ]

```
print(“Hours of sleep I got last night: “, 4)
```

Out:[ ]

```
Hours of sleep I got last night: 4
```

OR using a variable to store the value

In:[ ]

```
hours_sleep = 4
print(“Hours of sleep I got last night: “, hours_sleep)
```

Out:[ ]

```
Hours of sleep I got last night: 4
```

5.2 `input` statement

Explain the content below with the In-Lab Notebook open!

- `input()`: takes input from the user and returns it as a string
  - Prompt in parentheses is displayed

```
variable = input(prompt)
```

- The string received can be assigned to a variable to be used later in the program

```
In:[ ]   age = input(“Please enter your age: ”)
```

```
Out:[ ]  Please enter your age: 100
```

```
age = “100”
```

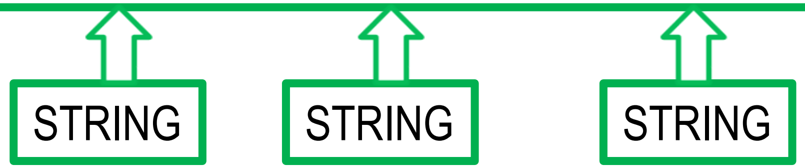
NOTE: This is 100 as a string which is text representation. If you want to do calculations with the input, you should convert it to a numeric data type like integer or float first. This is done with Type Conversions (covered next lab)

5.3 `input` and `print` Example

Explain the content below with the In-Lab Notebook open!

- Here’s an example using both the `input` and `print` statements
- Take the input from the user using a prompt and then display what they entered

```
age = input(“Please enter your age: ”)
print(“You are “ + age + “ years old.”)
```



You can combine strings into a single string using the plus ‘+’ sign. This is called string concatenation.

```
age = input(“Please enter your age: ”)
print(“You are age years old.”)
```

If you try to display the result with the code above. It will NOT DISPLAY the inputted age but the word ‘age’. Age will not be recognized as a variable.

```
Example
“You are “ + “100” + “ years old.” becomes “You are 100 years old.”
```

Wrap-Up

Recap

- Intro to Computing
  - Computing provides a structured approach to problem-solving
- Flowcharts
  - Flowcharts allow us to visually plan out an algorithm or process
- Variables
  - Variables are containers to hold data/values using a name
- Data Types
  - Data is represented in Python through various data types
- Calculations
  - Calculations can be done in Python with Math Operators
- `print()` & `input()` statements
  - You can use `print()` to display text and `input()` to receive user input