Procedural programming in Python
One-Stop Access To Course Information

- **Course website**: One-stop access to all course information.
  
  [http://www2.cs.sfu.ca/CourseCentral/120/liaqata/WebSite/index.html](http://www2.cs.sfu.ca/CourseCentral/120/liaqata/WebSite/index.html)

  - Course Outline
  - Exam Schedule
  - Python Info
  - [CourSys/Canvas](https://canvas.sfu.ca/courses/39187) link
  - Learning Outcomes
  - Office Hours
  - Textbook links
  - and more...
  - Grading Scheme
  - Lab/Tutorial Info
  - Assignments

- **Canvas**: Discussions forum - [https://canvas.sfu.ca/courses/39187](https://canvas.sfu.ca/courses/39187)

- **CourSys**: Assignments submission, grades - [www.coursys.sfu.ca](http://www.coursys.sfu.ca)

Liaqat Ali, Summer 2018.
Additional Resources / Online References

• Online references are as important as the texts. (Links on course website.)

• These resources are very important to your success.
  • They aren’t meant to be read from beginning to end like the readings in the textbook.

• You should use them to get an overall picture of the topic and as references as you do the assignments.
How to Learn in This Course?

A  Attend Lectures & Labs
R  Read / review Textbook/Slides/Notes
R  Reflect and ask Questions
R  Organize – your learning activities on weekly basis, and finally...
W  Write Code, Write Code, and Write Code.
Course Topics

1. General introduction
2. Algorithms, flow charts and pseudocode
3. Procedural programming in Python
4. Data types and control structures
5. Fundamental algorithms
6. Binary encodings
7. Basics of computability and complexity
8. Basics of Recursion
9. Subject to time availability:
   ▪ Basics of Data File management
Today’s Topics

1. What is a program?
2. Arithmetic operators
Think of your own definition of a program.

- A **sequence of instructions** that specifies how to solve a **problem**.
- A sequence of instructions that specifies how to perform a **computation**. [Think Python]
- A sequence of instructions written in a programming language that specifies how to perform a computation.
- A set of instructions that a computer follows to perform a task. [Gaddis]
- Programs are commonly referred to as **Software**.
- **Programmer**: Person who designs, creates, and test programs – also called developer.

Liaqat Ali, Summer 2018.
Common Instruction Types Used in Programs

- **Input instructions**: To get data from keyboard, a file, or some other device.
  - For example: `input()`

- **Output**: To display data on screen, or save in a file, etc.
  - For example: `print(sum)`

- **Math instructions**: To perform basic arithmetic operations.
  - For example, addition, multiplication etc. `sum = mid + final`

- **Conditional**: Check for certain conditions and run the appropriate code.
  - For example: `If (n1 < num2):`

- **Repetition**: Perform some action repeatedly, usually with some variation.
  - For example: `while (N <=100):`
• Believe it or not, that’s pretty much all there is to program.

• Every program you’ve ever used, no matter how complicated, is made up of instructions that look pretty much like these (on the previous slide).

• So you can think of programming as the process of breaking a large, complex task into smaller and smaller subtasks until the subtasks are simple enough to be performed with one of these basic instructions.” [Think Python]
Write a Program for Following Marks Flowchart

\[ m = 0 \]
\[ f = 0 \]
\[ t = 0 \]
\[ m = \text{input()} \]
\[ f = \text{input()} \]
\[ t = \text{float}(m) + \text{float}(f) \]

if (t<50):
    print("fail")
else:
    print("pass")

if t>=95: print("A+")
elif t>=90 and t<95: print("A")
elif t>=85 and t<90: print("A-")
elif t>=80 and t<85: print("B+")
elif t>=75 and t<80: print("B")
elif t>=70 and t<75: print("B-")
elif t>=65 and t<70: print("C+")
elif t>=60 and t<65: print("C")
elif t>=55 and t<60: print("C-")
elif t>=50 and t<55: print("D")
ext: print("F")
Arithmetic Operators

• Mathematical computations addition, subtraction, multiplication, and addition are also know as Arithmetic Operations.

• The +, -, *, and / symbols we use to represent these arithmetic operations are called Arithmetic Operators, or more generally, operators.

• In general, we can say that operators are special symbols that represent computations.
Arithmetic Operators: Examples

- **Addition operator:**
  \[ \text{>>> } 50 + 10 \]
  \[ 60 \]

- **Subtraction operator:**
  \[ \text{>>> } 50 - 10 \]
  \[ 40 \]

- **Multiplication operator:**
  \[ \text{>>> } 50 * 10 \]
  \[ 500 \]

- **Division operator:**
  \[ \text{>>> } 50 / 10 \]
  \[ 5.0 \]
More Operators

- **Exponentiation operator**: $5^2 = 25$

- **Floor division operator**: Division that results into round-down whole number.
  >>> 57 // 10
  5

- **Modulus (remainder) operator**: Remainder of the division (57/10).
  >>> 57 % 10
  7
Questions?