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
  
  - Course Outline
  - Exam Schedule
  - Python Info
  - CourSys/Canvas link
  - Learning Outcomes
  - Office Hours
  - Textbook links
  - and more...
  - Grading Scheme
  - Lab/Tutorial Info
  - Assignments

• Canvas: Discussions forum - https://canvas.sfu.ca/courses/39187

• CourSys: Assignments submission, grades - www.coursys.sfu.ca

Liaqat Ali, Summer 2018.
Some Reminders

• Get familiar with the course Website.
  ▫ http://www2.cs.sfu.ca/CourseCentral/120/liaqata/WebSite/index.html
  ▫ Minor updates may occur during first week.

• Get fob to access LABS (start next week!)
  ▫ If you don’t have it already, get a new fob from Discovery Park 1.
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
O: Organize – your learning activities on weekly basis, and finally...

Liaqat Ali, Summer 2018.
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. Continue with Algorithms, Flowcharts
2. Pseudocodes
Today’s Topics

Continue with Algorithms, Flowcharts
Algorithm: Find the Smallest of Three Numbers

Step 1: Start
Step 2: Declare variables \( n_1, n_2, \) and \( n_3. \)
Step 3: Read variables \( n_1, n_2, \) and \( n_3. \)
Step 4: If \( n_1 < n_2 \) then:
Step 5: If \( n_1 < n_3 \) then print \( n_1 \) else print \( n_3. \)
Step 6: else
Step 7: If \( n_2 < n_3 \) then print \( n_2 \) else print \( n_3. \)
Step 8: End
Flowchart: Smaller of Three Numbers (Solution)

START

Declare N1, N2, N3

Read N1, N2, N3

A

START

READ N1, N2, N3

A

If N1 < N2

No

If N1 < N3

If N2 < N3

True

Print N2

False

False

Print N3

False

If N1 < N3

True

Print N1

END
Today’s Topics

Pseudocodes

Liaqat Ali, Summer 2018.
Pseudocodes

- You can think of **Pseudocodes same as Algorithms**: a sequence of steps to solve a problem, except:
  - Steps in algorithm may be **less detailed**, a pseudocode describe those steps.
  - Steps in an algorithm look **more like an English** (natural) language instructions, whereas, steps in a pseudocode may look **more like a code**.
- For example:
  - A step in algorithm may be written like this:  **Convert feet into inches.**
  - An equivalent pseudocode may be written as:  **Set inches to feet * 12**
- **What’s common**: We can transform the instruction written as algorithms, flowcharts or pseudocode into a programming language code.
But, the algorithms we write in the natural language may be not easy to transform into code – especially for large and complex problems.

It would generally be more helpful to be “short” and “specific”, i.e., “describe” our algorithms in a way that’s easy to transform into code.

So, pseudocode is a way to describe the steps in an algorithm using some short and simple English (natural) language terms. (Pseudo is “almost”.)

It describes an algorithm in specific enough detail to be easily implemented in any language.

Actually, some of the algorithms we wrote in the previous two classes equally qualify as pseudocodes.
Pseudocodes: Features

- We typically use short phrases or keywords to describe steps in a pseudocode.
- For example:
- **READ, WRITE, SET, IF, ELSE, ENDIF, WHILE, ENDWHILE, REPEAT, UNTIL**
- Pseudocodes omit language specific syntax.
- It enables the programmers to concentrate on writing the coding.
Algorithm/Pseudocode: Smaller of Three Numbers

1: Start
2: Declare variables n1, n2, and n3
3: Read variables n1, n2, and n3
4: If n1 is smaller than n2 and n3, then n1 smaller.
5: If n2 is smaller than n1 and n3, then n2 smaller.
6: If n3 is smaller than n1 and n2, then n3 smaller.
7: End

Read n1, n2, n3.
If n1 < n2:
  If n1 < n3, Write n1.
  Else Write n3
If n2 < n1:
  If n2 < n3, Write n2.
  Else Write n3.

Liaqat Ali, Summer 2018.
Pseudocode: Find Sum of First 100 Natural Numbers

Step 1: Start
Step 2: Declare N and S.
Step 3: Set initial value of S to 0.
Step 4: Set initial value of N to 1.
Step 5: Add the value of N to S, giving S.
Step 6: Get the next number by add 1 to N.
Step 7: Repeat steps 5 to 6 until N is equal 100
Step 8: Display S.
Step 9: End

Set S to 0
Set N to 1
Repeat until N <= 100:
Set S=S+N
Set N=N+1
Write S

Liaqat Ali, Summer 2018.
Flowchart: Find Sum of First 100 Natural Numbers

START

Declare N, S

S = 0

N = 1

A

S = S + N

N = N + 1

N <= 100

Yes

No

Print S

END

Liaqat Ali, Summer 2018.
Algorithm: Convert Height In Meters To Feet and Inches

1: Start
2: Declare meter, feet, total inches and inches variables.
3: Initialize feet, total inches and inches variables to 0.
4: Get the height in meters from the user.
5: Convert meters into total inches and store it.
5: Convert total inches into feet and store it.
6: Find remainder of total inches / 12 and store in inches.
7: Display the value in feet variable.
8: Display the value in the inches variable.
9: End

Read meters
Set totInch to 39.37 × metres
Set feet to totInch/12 \(\text{floor}\)
Set inches to totInch – 12*feet
Write feet, inches
Flowchart: Convert Height In Meters To Feet and Inches

START

Declare m, flnt, in, ti

Set ti = 0, flnt = 0, in = 0

Read m

A

A

ti = m * 39.37

flnt = ti div 12

in = ti - flnt * 12

Print flnt, in

END
Why Pseudocodes?

Writing code to solve a problem would have two parts:

1. Identifying what to: Writing **Pseudocode** (Algorithm)
2. Knowing how to do: Writing **Python** code

• So, write an algorithm, express it in pseudocode before you start coding.

• Especially as you’re **starting to program**, you **don’t want to be worrying** about what you’re trying to say and **how to say it at the same time**.
Questions?