CMPT 120: Introduction to Computing Science and Programming

Turtle graphics, and User-defined Functions
Reminders

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.
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...
W  Write Code, Write Code, and Write Code.
Deliverables

1. Deliverables are due by the given date and time.
2. For the course, we are using IDLE to write and run our Python code.
3. You can use the CSIL lab computers outside your lab hours.
4. Plan ahead your assignments and other deliverables. Computer crash, network problems etc. are not acceptable excuses for delays in deliverables.
5. You may use online Python interpreters for running and testing your codes, such as: https://repl.it/languages/Python3
Labs

1. Each lab has an assigned TA.
2. Attend your assigned lab and show your work to your TA for the participation marks.
3. Class enrolments and lab swaps are closed now.
Course Topics

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

1. Turtle Graphics: Drawing and Animation
2. Introduction to Functions: User-defined
3. Defining and Calling a Void Function
4. Designing a Program to Use Functions
5. Passing Arguments to Functions
Graphics: Drawing and Animation Using Turtle

Liaqat Ali, Summer 2018.
Turtle Intro

Turtle is a Python feature that allows you to draw and animate graphic shapes.

```python
# Import turtle package
import turtle

# Create our turtle
myTurtle = turtle.Turtle()

# Move forward 50 pixels
myTurtle.forward(50)

# Turn right 90 degrees
myTurtle.right(90)

# Move forward 50 pixels
myTurtle.forward(50)
```

Liaqat Ali, 2018
Demo and Resources

1. **turtle — Turtle** Documentation (Methods): https://docs.python.org/3.5/library/turtle.html
2. **Turtle examples**: https://michael0x2a.com/blog/turtle-examples
4. https://michael0x2a.com/blog/turtle-examples (squares)
5. https://trinket.io/python/82fe4d3bd0 (interactive)

Liaqat Ali, 2018
Using turtle in Python

• To make use of the turtle methods and functionalities, we need to import turtle.
• ”turtle” comes packed with the standard Python package and need not be installed externally.
• Four steps for executing a turtle program:
  1. Import the turtle module
  2. Create a turtle to control (using Turtle())
  3. Draw around using the turtle methods.
  4. Run turtle.done().
# Common Turtle Methods (See [Documentation](#))

<table>
<thead>
<tr>
<th>METHOD</th>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turtle()</td>
<td>None</td>
<td>Creates and returns a new turtle object</td>
</tr>
<tr>
<td>forward()</td>
<td>amount</td>
<td>Moves the turtle forward by the specified amount</td>
</tr>
<tr>
<td>backward()</td>
<td>amount</td>
<td>Moves the turtle backward by the specified amount</td>
</tr>
<tr>
<td>right()</td>
<td>angle</td>
<td>Turns the turtle clockwise</td>
</tr>
<tr>
<td>left()</td>
<td>angle</td>
<td>Turns the turtle counter clockwise</td>
</tr>
<tr>
<td>penup()</td>
<td>None</td>
<td>Picks up the turtle’s Pen</td>
</tr>
<tr>
<td>up()</td>
<td>None</td>
<td>Picks up the turtle’s Pen</td>
</tr>
<tr>
<td>down()</td>
<td>None</td>
<td>Puts down the turtle’s Pen</td>
</tr>
<tr>
<td>color()</td>
<td>Color name</td>
<td>Changes the color of the turtle’s pen</td>
</tr>
<tr>
<td>fillcolor()</td>
<td>Color name</td>
<td>Changes the color of the turtle will use to fill a polygon</td>
</tr>
</tbody>
</table>

Adapted from: Janice Regan, 2013.
Turtle coordinates

Adapted from: Angelica Lim, 2018.
Introduction to Functions

Liaqat Ali, Summer 2018.
Introduction to Functions

• **Function**: group of statements within a program that perform as specific task.
  ▫ Usually one task of a large program.
    • Functions can be executed in order to perform overall program task.
  ▫ Known as *divide and conquer* approach

• **Modularized program**: program wherein each task within the program is in its own function.
Functions: A Divide and Conquer Approach

- We use functions to Divide and Conquer a large task by dividing into subtasks.
- We also call it a modular approach.

Liaqat Ali, 2018: Adapted from: Copyright © 2015 Pearson Education, Inc.
Function Example

# Program to add two numbers.
num1 = 5
num2 = 6
sum = num1 + num2
print(sum)

# A user-defined function to add
def add_numbers(x, y):
    sum = x + y
    return sum

# A user-defined function to add
num1 = 5
num2 = 6
sum = add_numbers(num1, num2)
print(sum)
# Calculator.

```
num1 = 5
num2 = 6
sum = num1 + num2
sub = num1 - num2
mul = num1 * num2
div = num1 / num2
print(sum)
```

# A user-defined function to add
```
def add_numbers(x, y):
    addition = x + y
    return addition
```

```
def sub_numbers(a, b):
    sub = a - b
    return sub
```

```
num1 = 5
num2 = 6
sum = add_numbers(num1, num2)
sub = sub_numbers(num1, num2)
print(sum, sub)
```
Benefits of Modularizing a Program with Functions

• The benefits of using functions include:
  ▫ Simpler code
  ▫ Code reuse
    • write the code once and call it multiple times.
  ▫ Better testing and debugging.
    • Can test and debug each function individually.
  ▫ Faster development.
  ▫ Easier facilitation of teamwork
    • Different team members can write different functions.

Liaqat Ali, 2018: Adapted from: Copyright © 2015 Pearson Education, Inc.
Void Functions and Value-Returning Functions

Liaqat Ali, Summer 2018.
Void Functions and Value-Returning Functions

- A **void function**: 
  - Simply executes the statements it contains and then terminates.

- A **value-returning function**: 
  - Executes the statements it contains, and then it returns a value back to the statement that called it.
  - The `input`, `int`, and `float` functions are examples of value-returning functions.
Defining and Calling a Function

- Functions are given names (like we give names to variables).
  - Function naming rules:
    - Cannot use key words as a function name.
    - Cannot contain spaces.
    - First character must be a letter or underscore.
    - All other characters must be a letter, number or underscore.
    - Uppercase and lowercase characters are distinct.
Defining and Calling a Function (cont’d.)

- Function name should be descriptive of the task carried out by the function.
  - Often includes a verb
- **Function definition:** Specifies what function does.
  ```python
  def function_name():
    statement
    statement
  ```

1. **Function header:** First line of function.
   - Includes keyword `def` and function name, followed by parentheses and colon.
2. **Block:** Set of statements that belong together as a group.
3. **Call** a function to execute it.
   - When a function is called:
     • Interpreter jumps to the function and executes statements in the block.
     • Interpreter jumps back to part of program that called the function.
     • Known as function return
Defining and Calling a Function (cont’d.)

• **main function**: Called when the program starts.
  - Calls other functions when they are needed.
  - Defines the *mainline logic* of the program.
Indentation in Python

• Each block **must** be indented
  ▫ Lines in block must begin with the same number of spaces.
  • Use tabs or spaces to indent lines in a block, but not both as this can confuse the Python interpreter
  • IDLE automatically indents the lines in a block
  ▫ Blank lines that appear in a block are ignored
Questions?