CMPT 120: Introduction to Computing Science and Programming 1

Procedural programming in Python

Copyright © 2018, Liaqat Ali. Based on CMPT 120 Study Guide and Think Python - How to Think Like a Computer Scientist, mainly. Some content may have been adapted from earlier course offerings by Diana Cukierman, Anne Lavergn, and Angelica Lim. Copyrights © to respective instructors. Icons copyright © to their respective owners.
Reminders

Liaqat Ali, Summer 2018.
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?

Attend Lectures & Labs

Read / review Textbook/Slides/Notes

Reflect and ask Questions

Organize – your learning activities on weekly basis, and finally...

Write Code, Write Code, and Write Code.

Liaqat Ali, Summer 2018.
Additional / Online References

- **Additional references** are as **important** as the texts, and **very important to your success**.
  
  - They aren’t meant to be read from beginning to end like the readings in the textbook.

- Use them to get an **overall picture** of the topic and as **references** as you do the assignments.
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. Programs Recap
2. Expressions
3. Operands
4. Operators
   i. Arithematic Operators (+, -, *, /)
   ii. Comparison operators
5. Reflection
Programs Recap

Liaqat Ali, Summer 2018.
midterm = 0
final = 0

midterm = int(input("Enter midterm:"))
final = int(input("Enter final:"))

total = float(midterm) + float(final)

if total>=95: print("A+")
elif total>=90 and total<95: print("A")
elif total>=85 and total<90: print("A-")
elif total>=80 and total<85: print("B+")
elif total>=75 and total<80: print("B")
elif total>=70 and total<75: print("B-")
elif total>=65 and total<75: print("C+")
elif total>=60 and total<65: print("C")
elif total>=55 and total<60: print("C-")
elif total>=50 and total<55: print("D")
else: print("F")
Program Recap: Sum of Natural Numbers (Solution)

```python
sum = 0
n = 1
while ( n <= 100 ):
    sum=sum+n
    n=n+1
print(sum)
```

```python
sum = 0
n = 1
limit = int(input("How many numbers? "))
while ( n <= limit ):
    sum=sum+n
    n=n+1
print(sum)
```
Expressions

Liaqat Ali, Summer 2018.
Expressions

• We are now familiar with, and have used in our programs:
  a. values, such as 5, 7, or 100
  b. variables, such as midterm, final, or total
  c. operators, such as +, /, or %

• An expression is a combination of values, variables, and operators.
  ▫ So, 5 + 7 is an expression; n + 1 is an expression; 2*x + 2*y is an expression;
  ▫ 15 < 20 is an expression, and even 45 by itself is an expression.
  ▫ We can categorize expressions based on their result types:
    1. Arithmatic expressions
    2. Boolean expressions
Arithmetic Expressions

• When result of an expression is a numeric value, we can call it an **Arithmetic Expression**.
  ▫ For example, \( n + 1 \) is an arithmetic expression if \( n \) is numeric.
    • Suppose \( n \) is 5, then the value of the arithmetic expression \( n + 1 \) would be 6, which is a numeric value.
  ▫ **meters * 39.37** is an arithmetic expression if meters is numeric.
    • Suppose meters is 2, then the value of the arithmetic expression **meters * 39.37** would be 78.74, which is again a numeric value.
  ▫ A numeric value can be an integer (whole number), or
  ▫ A floating point number (with decimal point).
When the result of an expression is either True or False, we call it a Boolean Expression. For example:

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 &lt; 7</td>
<td>Is 5 less than 7?</td>
</tr>
<tr>
<td>marks &gt; 95</td>
<td>Is marks greater than 90?</td>
</tr>
</tbody>
</table>
Operands
In our previous class, we talked about operators.

- **Operators** are special symbols that represent **computations**.

- **Arithmetic Operators** are symbols we use to represent arithmetic operations. For example, +, -, *, or /.

- We’ll continue with operators today and will talk about **Comparison Operators**.

- But first, let’s get familiar with a new term **Operands**.
Operands

- **Operands** are the values that appear on either side of an operator.
  - For example, in an arithmetic expression $50 + 10$, the values 50 and 10 are operands.
  - In $70 * 15$, the values 70 and 15 are operands.
- They are the data to be **operated on** by the operator.
- So, think of operands **just another name** for the values operators use.
- **Operands** can be values or variable names.
  - For example, in $\text{mid} + \text{final}$, both the operands mid and final are variables.
Comparison Operators
midterm = 0
final = 0

midterm = input("Enter midterm:")
final = input("Enter final:")

total = float(midterm) + float(final)

if total>=95: print("A+")
elif total>=90 and total<95: print("A")
elif total >= 85 and total < 90: print("A-")
elif total >= 80 and total < 85: print("B+")
elif total >= 75 and total < 80: print("B")
elif total >= 70 and total < 75: print("B-")
elif total >= 65 and total < 70: print("C+")
elif total >= 60 and total < 65: print("C")
elif total >= 55 and total < 60: print("C-")
elif total >= 50 and total < 55: print("D")
elif total >= 49: print("F")

Liaqat Ali, Summer 2018.
Comparison Operators

• The symbols `<`, `>`, `<=`, `>=`, `==` and `!=` are called **comparison operators**. (They are 6 in number.)

• **Comparison operators** are used to compare values or operands.
  - For example in a Boolean expression:
    - `5 < 6`: the symbol `<` is a comparison operator, and `5` and `6` are values.
    - `total >= 90`, `>=` is a comparison operator, and `total` and `90` are values.

• A comparison either returns a **True** or **False** result.
  - An expression that results into a true or false value is called a Boolean Expression.
## Comparison Operators: Descriptions and Examples

Suppose:  \( a = 5 \)  \( b = 7 \)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Expression</th>
<th>Meaning</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>less than</td>
<td>( a &lt; b )</td>
<td>is a less than ( b )?</td>
<td>True</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal</td>
<td>( a &lt;= b )</td>
<td>is a less than or equal to ( b )?</td>
<td>True</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
<td>( a &gt; b )</td>
<td>is a greater than ( b )?</td>
<td>False</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal</td>
<td>( a &gt;= b )</td>
<td>is a greater than or equal to ( b )?</td>
<td>False</td>
</tr>
<tr>
<td>==</td>
<td>equal</td>
<td>( a == b )</td>
<td>is a equal to ( b )?</td>
<td>False</td>
</tr>
<tr>
<td>!=</td>
<td>not equal</td>
<td>( a != b )</td>
<td>is a not equal to ( b )?</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( a &lt;&gt; b )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Grade Program Example

Recall this program we wrote last week.

In the comparison expression if total >= 50:
  ▫ What conditional operator did it use?
    • >= (greater than or equal)
  ▫ What are the operands?
    • total and 50
  ▫ What are the possible outcomes?
    ▪ Either the total is greater than or equal 50.
      • We call this outcome as True
    ▪ Or, the total is not greater than or equal 50.
      • We call this outcome as False

midterm = input()
final = input()
total = float(mid) + float(final)
if total >= 50:
    print(“Pass”)
else:
    print(“F”)

Liaqat Ali, Summer 2018.
Grade Computation Program

```python
midterm = 0
final = 0
total = 0

midterm = input()
final = input()
total = float(midterm) + float(final)

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

Greater than or equal comparison operator

Less than comparison operator.

We’ll talk next.
Reflection
Compute Grade – Version 2

```python
midterm = 0
final = 0
grade = ""

midterm = input("Enter midterm:")
final = input ("Enter final:")

total = float(midterm) + float(final)

if total>=95: grade = "A+
elif total>=90 and total<95: grade = "A"
elif total>=85 and total<90: grade = "A-
elif total>=80 and total<85: grade = "B+
elif total>=75 and total<80: grade = "B"
elif total>=70 and total<75: grade = "B-
elif total>=65 and total<70: grade = "C+
elif total>=60 and total<65: grade = "C"
elif total>=55 and total<60: grade = "C-
elif total>=50 and total<55: grade = "D"
else: grade = "F"

print(grade)
print(total, grade)
print("Total marks = ", total, "Grade = ", grade)
```
Class Participation Activity

1. Copy and run this program.

2. Reflect on:
   a. How this program differs from the program on slide 8.
   b. Why it produces the same output as the program on slide 8.
   c. Which approach you think is better: direct printing, or using a variable? Think of one reason?
   d. What if you replace the word and with or in the if conditionals?
      • Modify and run the program for the pair of values 34, 40; 60, 30; 23, 4; and 45, 46.

3. Post your reflections on the Canvas Discussions forum after the class today.
Questions?