SFU SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

CMPT 120: Introduction to Computing Science and Programming 1

A Quick Review – Main Concepts

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.

Input Validation Loops

- It is important to design program such that bad input is never accepted.
  - GIGO: garbage in, garbage out
- **Input validation**: inspecting input before it is processed by the program
  - If input is invalid, prompt user to enter correct data
  - Commonly accomplished using a `while` loop which repeats as long as the input is bad.
    - If input is bad, display error message and receive another set of data
    - If input is good, continue to process the input.

Copyright © 2018 Pearson Education, Inc.

Sentinels

- **Sentinel**: special value used to mark end of a sequence of items or loop.
  - When program reaches a sentinel, it knows that the end of the sequence of items was reached, and the loop terminates.
  - ```
    user_input = 1
    sum = 0
    while user_input != -99:
        user_input = int(input("Enter your number or -99 to end."))
        sum = sum + user_input
    print("The sum of numbers is:  {}".format(sum))
    ```

Copyright © 2018 Pearson Education, Inc.
    ```

Nested Loops

- Loop that is contained **inside** another loop.
- Key points about nested loops:
  - Inner loop goes through all of its iterations for each iteration of outer loop.
  - Inner loops complete their iterations faster than outer loops.

Copyright © 2018 Pearson Education, Inc.

Binary Data Representation

- Data inside computer is **not represented** the same way as we represent numbers and letters in English or native language. **For example**:
- **Problem!!!**
- Computer don't use (recognize) the symbols 0,1,2..9 or alphabets a, b, c,…z
- Computer uses a binary language representation.
- The **binary language** consists of two symbols only: **0** and **1**
- That means, every thing in computer **MUST** be represented using the symbols **0** and **1,** only

Binary Codes: ASCII

Letter	ASCII Binary Code							
A	0	1	0	0	0	0	0	1
B	0	1	0	0	0	0	1	0
C	0	1	0	0	0	0	1	1
D	0	1	0	0	0	1	0	0
E	0	1	0	0	0	1	0	1
F	0	1	0	0	0	1	1	0
…								

- **ASCII:** American Standard Code for Information Interchange. (256 codes.)
- Used in computers to represent characters since 1963.
- ASCII uses 8-bits to represent one character of English language.

Letter	ASCII Binary Code							
a	0	1	1	0	0	0	0	1
b	0	1	1	0	0	0	1	0
c	0	1	1	0	0	0	1	1
d	0	1	1	0	0	1	0	0
e	0	1	1	0	0	1	0	1
f	0	1	1	0	0	1	1	0
…								

- Space required to represent a single binary 0 or 1 is called **bit**.
- Space required to represent 8-bits is called a **byte**.
- See a complete list of ASCII codes here: www.ascii-code.com

Number Systems

- **Binary Number System**: Uses **two** unique symbols to represents numbers or data. (0 and 1).
- **Decimal system**: Use **ten** unique symbols to represent numbers. (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9).
- **Octal system**: Use **eight** unique symbols to represent numbers. (0, 1, 2, 3, 4, 5, 6, and 7).
- **Hexa-decimal system**: Use **sixteen** unique symbols to represent numbers. (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A,B,C,D,E and F).

- We can convert between number systems.

1	1	1	1	1	1	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

Examples

1	0	1	0	1	0	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
128	0	32	0	8	0	2	1

= 171

0	0	1	0	0	0	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	0	32	0	0	0	2	1

= 35

Converting from Decimal to binary

- 111
  - 128 too large from 111,
    - so there are **zero** 128 in 111.
  - 111 - 64 = 47
    - There is **one** 64 in 111, remainder 47.)
  - 47 - 32 = 15 (there is **one** 32 in 47, remainder 15.)
  - 16 too large (there are **zero** 16 in 15.)
  - 15 – 8 = 7 (there is **one** 8 in 15, remainder 7.)
  - 7 – 4 = 3 (there is **one** 4 in 7, remainder 3.)
  - 3 – 2 = 1 (there is **one** 3 in 3, remainder **1**.)

1	1	1	1	1	1	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	**1**	**1**	**0**	**1**	**1**	**1**	**1**

ASCII: Decimal Equivalent

Letter	ASCII Binary Code								
	128	64	32	16	8	4	2	1	
A	0	1	0	0	0	0	0	1	64+1 = 65
B	0	1	0	0	0	0	1	0	64+2 = 66
C	0	1	0	0	0	0	1	1	64+2+1 = 67
D	0	1	0	0	0	1	0	0	64+4 = 68
E	0	1	0	0	0	1	0	1	64+4+1 = 69
F	0	1	0	0	0	1	1	0	64+4+2 = 70
…									

- When we use Boolean expression ('a' < 'A'), computer would compare the ASCII value of a (which is 97) with the value of ASCII value of A (which is 65). So, answer: False

Letter	ASCII Binary Code								
	128	64	32	16	8	4	2	1	
a	0	1	1	0	0	0	0	1	64+32+1 = 97
b	0	1	1	0	0	0	1	0	
c	0	1	1	0	0	0	1	1	
d	0	1	1	0	0	1	0	0	
e	0	1	1	0	0	1	0	1	
f	0	1	1	0	0	1	1	0	
…									

- 'B' <= 'b'
- 'cd' <= 'ab'
- 'xyz' > 'XYZ'

Signed Integer Data Representation: Binary

- A **signed integer**: For a positive integer represented by N binary digits the possible values are $-2^{N-1}-1 \le value \le 2^{N-1}-1$.

Sign bit	N -1 Binary Digits

+/- **127**

	1	1	1	1	1	1	1
+/-	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	64	32	16	8	4	2	1

+12	0	0	0	0	1	1	0	0
-12	1	0	0	0	1	1	0	0

Signed Integer Data Representation: One's Complement

- Integer is represented by a string of **binary** digits.
  - But, is represented in 1's compliment form.

Sign bit	N -1 Binary Digits: 1's Compliment

- **How a number is converted to its 1's Compliment form:**

1. If a number is positive, simply convert the number to its binary equivalent.
   - For example, if the number is: **6** **0 0 0 0 0 1 1 0**
2. If a number is negative, **convert** the number to its binary equivalent and **flip** the bits.
   - For example , if the number is: -6 **0 0 0 0 0 1 1 0**
   - Flip the bits: **1 1 1 1 1 0 0 1**

Signed Integer Data Representation: One's Complement

- Suppose an 8-bit 1's pattern is shown as: **1 0 1 1 0 0 0 1**

- **What number this pattern represents?**
  - If first bit 0, then it is an unsigned/positive number, as shown (simply convert it to its decimal equivalent).
  - If first bit is 1, then:
    1. Flip all the bits. So, **1011 0001** becomes **0100 1110**
    2. Convert to decimal: 01001110 $= 2^6 + 2^3 + 2^2 + 2^1 = 64 + 8 + 4 + 2 = 78$
    3. Add a minus sign. So **10110001** represents **-78** in one's Complement form.

Adapted from: Janice Regan, 2013.

Examples: One’s complement

-84

1	0	1	0	1	0	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	1	0	1	0	1	0	0
-	64	0	16	0	4	0	0

Remember if first digit is1 flip bits.

35

0	0	1	0	0	0	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	0	1	0	0	0	1	1
	0	32	0	0	0	2	1

Adapted from: Janice Regan, 2013.

Decimal to 1s complement

- -49 (number < 0)
  - Express 49 in 8 bit binary
    - 32+16+1
    - 00110001
  - Flip the bits
    - 11001110

Two's Complement Signed Integer Representation

- **Integer is represented by a string of binary digits.**
  - Representation is in 2's compliment form.
  - Right most bit is used for sing.
  - Remaining bits represent the value.

Sign bit	*N-1 Binary Digits: 2's Compliment*

- **Decimal to 2's Compliment form:**
- **For a Positive Number:**
  1. First bit is 0.
  2. Convert the number to its binary equivalent.
- **+ 7** is represented as: 0000 0111
- **+ 13** is represented as: 0000 1101

- **For a Negative Number:**
  1. Convert the number to its binary equivalent.
  2. Flip the bits
  3. Add 1.
- **- 7** would be represented as:
  1. Convert to binary: 0000 0111
  2. Flip the bits: 1111 1000
  3. Add 1. 1 = 1111 1001
- **- 13** would be represented as:
  1. Convert to binary: 0000 1101
  2. Flip the bits: 1111 0010
  3. Add 1. 1 = 1111 0011

Two’s Complement Signed Integer Representation - 2

- **2’s Compliment to Decimal:**
- **If first bit is 0, then:**
  1. The number is positive.
  2. Simply, convert the binary number to its decimal equivalent.
- **0001 0111** is 2’s compliment representation of: $+2^4+2^2+2^1+2^0$ = +23
- **If first bit is 1, then:**
  - The number is negative.
  - Flip all the bits. So, **1011 0001** becomes **0100 1110**
  - Add 1. **1** = **0100 1111**
  - Convert to decimal: **0100 1111** = $2^6+2^3+2^2+2^1+2^0$ = 64+8+4+2+1 =79
  - So 1011001 represents **-79**

More Examples: Two's Complement to Decimal

Remember if first digit is 1 flip bits then add 1

-85

1	0	1	0	1	0	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	1	0	1	0	1	0	0
	1	0	1	0	1	0	1
-	64	0	16	0	4	0	1

35

0	0	1	0	0	0	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	0	1	0	0	0	1	1
	0	32	0	0	0	2	1

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)

METHOD	PARAMETER	DESCRIPTION
Turtle()	None	Creates and returns a new tutrle object
forward()	amount	Moves the turtle forward by the specified amount
backward()	amount	Moves the turtle backward by the specified amount
right()	angle	Turns the turtle clockwise
left()	angle	Turns the turtle counter clockwise
penup()	None	Picks up the turtle's Pen
up()	None	Picks up the turtle's Pen
down()	None	Puts down the turtle's Pen
color()	Color name	Changes the color of the turtle's pen
fillcolor()	Color name	Changes the color of the turtle will use to fill a polygon

Adapted from: Janice Regan, 2013.

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.

 Copyright © 2018 Pearson Education, Inc.

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.

This program is one long, complex sequence of statements.

```
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
statement
```

In this program the task has been divided into smaller tasks, each of which is performed by a separate function.

```
def function1():
    statement
    statement
    statement
```

function

```
def function2():
    statement
    statement
    statement
```

function

```
def function3():
    statement
    statement
    statement
```

function

Liaqat Ali, 2018: Adapted from: Copyright © 2018 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

- - - - - - - - - - - - - - - - - - - - -

num1 = 5
num2 = 6
sum = add_numbers(num1, num2)
print(sum)
```

User defined function area

Imaginary dividing line

main function area

Question 1

Q. What is the 1's complement for 10001001 binary numbers.

a. 01110110
b. 01011111
c. 00111001
d. 00001110

See answers on Slide 30.

 Copyright © 2018 Pearson Education, Inc.

Question 2

Q. Which of the following statements causes the interpreter to load the contents of the random module into memory?

a. load random
b. import random
c. upload random
d. download random

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

Question 3

Q. The Python standard library's ___________ module contains numerous functions that can be used in mathematical calculations.

a. math
b. string
c. random
d. number

Question 4

Q. What will be the output after the following code is executed?

```
def pass_it(x, y):
    z = x + ", " + y
    return(z)

name2 = “Jhon"
name1 = “King"
fullname = pass_it(name1, name2)
print(fullname)
```

a. Jhon King
b. King Jhon
c. Jhon, King
d. King, Jhon

ANS: d

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

Question 5

Q. What will be the output after the following code is executed?

```
def pass_it(x, y):
    z = x , ", " , y

num1 = 4
num2 = 8
answer = pass_it(num1, num2)
print(answer)
```

a. 4, 8
b. 8, 4
c. 48
d. None

ANS: d

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

Question 6

Q. When execute a function by:

a. calling it
b. locating it
c. defining it
d. exporting it

ANS: a

 Copyright © 2018 Pearson Education, Inc.

Answers

Answer 1 a

Answer 2 b

Answer 3 a

Answer 2 d

Answer 5 d

Answer 6 a

 Copyright © 2018 Pearson Education, Inc.

?

Questions?

Copyright © 2018 by Liaqat Ali