CST0006 – Computer Programming Foundations

INTRODUCTION to WEEK09

In this lecture and lab section we are going to learn about the different ways to take input and give output from programs in various different ways. Most programs you write will require some type of input or output or even both. We will also be learning about data structures and different ways to search and sort through data.

After this lecture and lab section, you will be able to write a program that takes input from a user, a file and is able to format print statements and output to files. You will also have a general understanding of data structures and searching and sorting through collections.

Input

Data that a program receives is called input. Input can be received by various different methods. It can be typed commands on a keyboard, through files saved somewhere on a hard drive, through information stored in a database. Basically any method that the programming language allows you to send information though.

Python offers the input() function to get text from the keyboard.

Python Input Example

```
print("How old are you?")

age = input()

print("You are", age)
```

Python’s input statement allows you to pass in a string as an argument, so you can condense your code.

Python Input Example

```
age = input("How old are you?\n")

print("You are", age)
```

Why doesn’t this work?

```
age = input("How old are you?\n")

age = age + 1

print("Next year you will be", age)
```

To fix this error we need to convert the string to an int so we can do mathematical operations on it.

```
age = input("How old are you?\n")

age = int(age) + 1

print("Next year you will be", age)
```

Output

Output is data that the program produces. Whether it's printing on a display, printed paper, files saved to a disk, lights flashing, sound it's all considered output. Usually before you input data, you need to have an output statement asking a user to input data. Python offers similar formatting that is used with the C style printf() statements.

Python Print Example 1 using commas to separate values

```
age = 15

print("This year you are", age, "and next year you will be", age+1)
```

Python Print Example 2 using string concatenation

```
age = 15

print("Next year you will be" + " " + str(age+1))
```

Python Print Example 3 using formatting operator

```
name = "Marty McFly"
age = 15

print ("My name is %s and I am %d years old" % (name, age))
```

Formatting Output

`%c`	Character	`%5d`	Indentation of 5
`%s`	String	`%6.2f`	Indentation of 6 + Precision of 2
`%d`	Integer	`%x`	Hexadecimal (lowercase)
`%f`	Floating Point	`%X`	Hexadecimal (uppercase)

Here is a full list of the formatting specifications: https://docs.python.org/3/library/string.html

Formatting integers and floats example

```
a = 7
b = 8.343
c = 45645

print("%6d" % a)
print("%4.2f" % b)
print("%6d" % c)
```

File Output

Just as we can print to the display, we can also output to a file. We need to use the open command to import a file name. The two parameters we use are the file name and how we want to open the file. In this case the ‘w’ is for write.

Opening a file to write

```
fout = open('file.txt', 'w')

line1 = "I do not like green eggs and ham,\n"
line2 = "I do not like them Sam I am."

fout.write(line1)
fout.write(line2)

fout.close()
```

File Input

Just as we can input from the keyboard, we can also input from a file. We need to use the open command to import a filename. The two parameters we use are the file name and how we want to open the file. In this case the ‘r’ is for read.

Opening a file for reading

```
fin = open('file.txt', 'r')

eggs = fin.read()

print(eggs)

fin.close()
```

Data Structures

Since the study of data structures is a complete course all on its own, we're only going to barely cover this topic here. I do want you to understand what they are and why they are useful.

Data structures are built into each programming language. Data structures are collections of data that allow you to store and manipulate data held within them. They each have specific functionality built into them that allows you to sort, search and manipulate the data held within them efficiently.

Python Specific Data Structures

List	Similar to an array with the exception that each data member (element) can be a different data type
Tuple	Similar to lists but is immutable, meaning the values cannot change
Dictionary	Contains key/value pairs, similar to a database
Sets	A collection of unordered unique and immutable objects

Examples Other Types of Data Structures

Array	A contiguous set of data elements that are of the same type
Linked List	A non-contiguous set of data elements each containing a link to the next element
Binary Tree	A data structure that can be connected to a maximum of two other data elements
Hash Table	A data structure that retrieves values using keys

Searching and Sorting

Big-O Notation

If you study data structures, one of the first things you will hear talked about is "Big-O". Basically this is just the speed and efficiency of each searching and sorting algorithm according to the amount of data you are adding. Not all data structures are the same. Some are best suited for their specific tasks, yet may suck at other tasks. If you are tasked with writing a program that deals with large amounts of data, you may want to research and understand which data structures best suit your need.

Common Examples of Big-O Notation

$O(1)$	Constant
$O(\log n)$	Logarithmic
$O(n)$	Linear
$O(n \log n)$	Log Linear
$O(n^2)$	Quadratic
$O(2^n)$	Exponential
$O(n!)$	Factorial

Bubble Sort Example

```
def bubble_sort(list):
    length = len(list) - 1
    for i in range(length):
        for j in range(length):
            if list[j] > list[j+1]:
                list[j], list[j+1] = list[j+1], list[j]
    return list

list = [9,4,5,3,2,6,7,4,3,2]

print(bubble_sort(list))
```

Building Better bubble Sort - Since we know after the first pass, the highest number is going to be to the furthest right-hand side, and on the second pass, we know that the second highest number is going to be next to the highest number on the right-hand side, we can see a pattern.

What can we do to the bubble sort to make this sorting algorithm faster?