CSC 2301

• Welcome to Class CSC 2301!

• “Introduction to Computer Programming for Non-Majors”

<table>
<thead>
<tr>
<th>Who are you?</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>What is your Major or Major you want?</td>
<td></td>
</tr>
<tr>
<td>What year are you in?</td>
<td></td>
</tr>
<tr>
<td><strong>Target of this course?</strong></td>
<td></td>
</tr>
</tbody>
</table>
CSC 2301

• Welcome to Class CSC 2301!

• “Introduction to Computer Programming for Non-Majors”
• CSC 2301 Syllabus
• Schedule
• Course Website:
  – Desire2learn

**Why Python?**
• Python programming language
• Many applications in diversified fields
• Python is: fun-to-learn and easy-to-use
• Book to use:
  • “Python Programming: An Introduction to Computer Science”
• Objectives
• To understand the respective roles of hardware and software in a computing system.

• To learn what computer scientists study and the techniques that they use.

• To understand the basic design of a modern computer.
• Objectives (cont.)

• To understand the form and function of computer programming languages.

• To begin using the Python programming language.

• To learn about chaotic models and their implications for computing.
Chapter 1: Computers and Programs

✓ Computers
✓ Computer Science
✓ Computer Program
✓ Programming Languages
✓ Magic of Python
✓ Inside Python
The Universal Machine

- A modern computer can be defined as: “a machine that **stores** and **manipulates** information under the control of a changeable program.”

- Two key elements:
  - Computers are devices for **manipulating information**.
  - Computers operate **under the control of a changeable program**.
• Chapter 1: Computers and Programs

✓ Computers
✓ Computer Science
✓ Computer Program
✓ Programming Languages

✓ Magic of Python
✓ Inside Python
What is Computer Science?

• It is not the study of computers!

• “Computers are to computer science what telescopes are to astronomy.”
  – E. Dijkstra

• The question becomes, “What processes can be described?”

• This question is really, “What can be computed?”
What is Computer Science?

• Design

– One way to show a particular problem can be solved is to actually design a solution.

– This is done by developing an algorithm, a step-by-step process for achieving the desired result.
What is Computer Science?

• Analysis

  – **Analysis** is the process of examining algorithms and problems mathematically.

  – Some seemingly simple problems are not solvable by any algorithm. These problems are said to be *unsolvable*.

  – Problems can be *intractable* if they would take too long or take too much memory to be of practical value.
What is Computer Science?

• **Experimentation**

  – Some problems are too complex for analysis.

  – Implement a system and then study its behaviour.
Basic Model of a Computer

- Input
- Output
- Control Unit
- Arithmetic/Logic Unit (ALU)
- Memory
- CPU
Hardware Basics

• The *central processing unit (CPU)* is the “brain” of a computer.

  – The CPU carries out all the basic operations on the data.

  – Examples: simple arithmetic operations, testing to see if two numbers are equal.
Hardware Basics

• Memory stores programs and data.
  – CPU can only directly access information stored in main memory (RAM or Random Access Memory).
  – Main memory is fast, but volatile, i.e. when the power is interrupted, the contents of memory are lost.
  – Secondary memory provides more permanent storage: magnetic (hard drive, floppy), optical (CD, DVD)
Hardware Basics

• **Input devices**
  – Information is passed to the computer through keyboards, mice, etc.

• **Output devices**
  – Processed information is presented to the user through the monitor, printer, etc.
Hardware Basics

- **Fetch-Execute Cycle**
  - First instruction retrieved from memory
  - Decode the instruction to see what it represents
  - Appropriate action carried out.
  - Next instruction fetched, decoded, and executed.
  - Lather, rinse, repeat!
Computer program
• What is a *computer program*?

  – A detailed, step-by-step set of instructions *telling a computer what to do*.

  – If we change the program, the computer performs a different set of actions or a different task.

  – The *machine stays the same, but the program changes*!
Program Power

• *Software* (programs) rule the *hardware* (the physical machine).

• The process of creating this software is called *programming*.

• Why learn to program?
  
  – Fundamental part of computer science
  – Having an understanding of *programming* helps you have an understanding of *the strengths and limitations of computers*.
Program Power

- Helps you become a more intelligent user of computers
- It can be fun!
- Form of expression
- Helps the development of problem solving skills, especially in analyzing complex systems by reducing them to interactions between simpler systems.

Programmer
• Chapter 1: Computers and Programs

✓ Computers
✓ Computer Science
✓ Computer Program
✓ Programming Languages

✓ Magic of Python
✓ Inside Python
Programming Languages

- **Natural language** has ambiguity and imprecision problems when used to describe complex algorithms.

  - Programs expressed in an unambiguous, precise way using *programming languages*.

  - Every structure in programming language has a precise form, called its *syntax*.

  - Every structure in programming language has a precise meaning, called its *semantics*. 
Programming Languages

• Programming language like a code for writing the instructions the computer will follow.

  – Programmers will often refer to their program as *computer code*.

  – Process of writing an algorithm in a programming language often called *coding*.
Programming Languages

- **High-level** computer languages
  - Designed to be used and understood by humans

- **Low-level** language
  - Computer hardware can only understand a very low level language known as *machine language*, represented in binary (1’s and 0’s)
Programming Languages

• High-level language
  \[ c = a + b \]

• This needs to be translated into machine language that the computer can execute.

• **Compilers** convert programs written in a high-level language into the machine language of some computer.

• Recommend:
Programming Languages

- *Interpreters* simulate a computer that understands a high-level language.

- The source program is not translated into machine language all at once.

- An *interpreter* analyzes and executes the source code instruction by instruction.
Programming Languages

• **Compiling vs. Interpreting**

  – Once program is compiled, it can be executed over and over without the source code or compiler. If it is interpreted, the source code and interpreter are needed each time the program runs.

  – Compiled programs generally run faster since the translation of the source code happens only once.
Programming Languages

- **Interpreted languages** are part of a more flexible programming environment since they can be developed and run **interactively**

- **Interpreted programs** are more portable, meaning the executable code produced from a compiler for a Pentium won’t run on a Mac, without recompiling. If a suitable interpreter already exists, the interpreted code can be run with no modifications.
Chapter 1: Computers and Programs

- Computers
- Computer Science
- Programming Languages

- Magic of Python
- Inside Python
The Magic of Python

When you start Python, you will see something like:

Python 3.4.2 (r312:79149, Mar 21 2010, 00:41:52) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.

>>> 

Google Python ➔ download ➔ Python 3.4 ➔ install ➔ using IDLE (Python GUI)

https://www.python.org/
The Magic of Python

• The “>>>” is a Python prompt indicating that Python is ready for us to give it a command. These commands are called statements.

• >>> print("Hello, world")
  Hello, world
>>> print(2+3)
  5
>>> print("2+3=", 2+3)
  2+3= 5
>>>
The Magic of Python

- Usually we want to execute several statements together that solve a common problem. One way to do this is to use a function.

```python
>>> def hello():
    print("Hello")
    print("Computers are Fun")

>>>
```
The Magic of Python

• >>> def hello():
   print("Hello")
   print("Computers are Fun")

>>> 

• The first line tells Python we are defining a new function called hello.
• The following lines are indented to show that they are part of the hello function.
• The blank line (hit enter twice) lets Python know the definition is finished.
The Magic of Python

- >>> def hello():
      print("Hello")
      print("Computers are Fun")

>>>  

- Notice that nothing has happened yet! We’ve defined the function, but we haven’t told Python to perform the function!

- A function is *invoked* by typing its name.

- >>> hello()
Hello
Computers are Fun
>>>  

- Tips: using Alt+p to get the last command
The Magic of Python

• What’s the deal with the ()’s?

• Commands can have changeable parts called **parameters** that are placed between the ()’s.

• >>> def greet(person):
    print("Hello", person)
    print("How are you?")

    >>>
The Magic of Python

• >>> greet("Terry")
  Hello Terry
  How are you?
>>> greet("Paula")
Hello Paula
How are you?
>>>  

• When we use parameters, we can customize the output of our function.
The Magic of Python

• When we exit the Python prompt, the functions we’ve **defined** cease to exist!

• Programs are usually composed of functions, **modules**, or **scripts** that are saved on disk so that they can be used again and again.

• A **module file** is a text file created in **text editing software** (saved as “plain text”) that contains function definitions.

• A **programming environment** is designed to help programmers write programs and usually includes automatic indenting, highlighting, etc.
The Magic of Python

# File: chaos.py
# A simple program illustrating chaotic behavior

def main():
    print("This program illustrates a chaotic function")
    x = eval(input("Enter a number between 0 and 1: "))
    for i in range(10):
        x = 3.9 * x * (1 - x)
    print(x)

main()

• We’ll use `filename.py` when we save our work to indicate it’s a Python program.
• In this code we’re defining a new function called `main`.
• The `main()` at the end tells Python to run the code.
This program illustrates a chaotic function
Enter a number between 0 and 1: .5
0.975
0.0950625
0.335499922266
0.869464925259
0.442633109113
0.962165255337
0.141972779362
0.4750843862
0.972578927537
0.104009713267

>>>
• Chapter 1: Computers and Programs

✓ Computers
✓ Computer Science
✓ Programming Language

✓ Magic of Python
✓ Inside Python
Inside a Python Program

# File: chaos.py
# A simple program illustrating chaotic behaviour

• Lines that start with # are called comments
• Python skips text from # to end of line
• Intended for human readers and ignored by Python
def main():

• Beginning of the definition of a function called main

• Since our program has only this one module, it could have been written without the main function.

• The use of main is customary, however.
Inside a Python Program
print("This program illustrates a chaotic function")

• This line causes Python to print a message introducing the program.
Inside a Python Program

```python
x = eval(input("Enter a number between 0 and 1: "))
```

- **x** is an example of a *variable*

- A variable is used to assign a name to a value so that we can refer to it later.

- The quoted information is displayed, and the number typed in response is stored in **x**.
Inside a Python Program

for i in range(10):

• For is a *loop* construct

• A loop tells Python to *repeat the same thing over and over*.

• In this example, the following code will be repeated 10 times.
Inside a Python Program

```
x = 3.9 * x * (1 - x)
print(x)
```

- These lines are the *body* of the loop.
- The body of the loop is what gets repeated each time through the loop.
- The body of the loop is identified through indentation.
- The effect of the loop is the same as repeating this two lines 10 times!
Inside a Python Program

for i in range(10):
    x = 3.9 * x * (1 - x)
    print(x)

• These are equivalent!
Inside a Python Program

\[ x = 3.9 \times x \times (1 - x) \]

• This is called an \textit{assignment} statement

• The part on the right-hand side (RHS) of the “\textasciitilde=” is a mathematical expression.

• \texttt{*} is used to indicate multiplication

• Once the value on the RHS is computed, it is stored back into (\textit{assigned}) into \texttt{x}
Inside a Python Program

main()

• This last line tells Python to execute the code in the function main
Chaos and Computers

• The `chaos.py` program:

```python
def main():
    print("This program illustrates a chaotic function")
    x = eval(input("Enter a number between 0 and 1: "))
    for i in range(10):
        x = 3.9 * x * (1 - x)
        print(x)
main()
```

• For any given input, returns 10 seemingly random numbers between 0 and 1

• It appears that the value of x is chaotic
Chaos and Computers

• The function computed by program has the general form \( k(x)(1-x) \) where \( k \) is 3.9

• This type of function is known as a logistic function.

• Very small differences in initial value can have large differences in the output.
Chaos and Computers

- Input: 0.25
  0.73125
  0.76644140625
  0.698135010439
  0.82189581879
  0.570894019197
  0.955398748364
  0.166186721954
  0.540417912062
  0.9686289303
  0.118509010176

- Input: 0.26
  0.75036
  0.73054749456
  0.767706625733
  0.6954993339
  0.825942040734
  0.560670965721
  0.960644232282
  0.147446875935
  0.490254549376
  0.974629602149
Reminder!

- Install Python
- Test Python codes
Questions

Thank You!