Welcome to CIS 210

• What is Computer Science?

• Computational Problem Solving

• What can you expect from CIS 210?

• Prior programming experience/Quiz

What is Computer Science?

... the study of algorithms (MR p. 2)

... the study of problems, the [computational problem-solving process], and [programming] the solutions [algorithms] to those problems. (MR p. 330)
Computational problem solving is an approach to problem solving that is inspired and constrained by the possibilities and limitations of computers and computing.

Enlisting a computer as a problem-solving partner requires addressing the limitations of computers 0, 1.

We need tools, abilities, approaches for communicating with the computer for computational problem solving.

science, math, design, engineering

A computational process is a sequence of well-defined operations that leads from an initial starting point to a computer-generated desired final outcome.

FOR EXAMPLE: Fizzbuzz

0. form into groups of 3-5 students
1. the first person says the number 1
2. go around the group, with each person saying the next number in turn
3. though if the number is divisible by 3, say “fizz”, and if the number is divisible by 5, say “buzz”, and if the number is divisible by 3 and 5, say “fizzbuzz”
4. if an error is made, start again
5. stop when you reach 100
FOR EXAMPLE: Fizzbuzz

0. form into groups of 3-5 students
1. the first person says the number 1
2. go around the group, with each person saying the next number in turn
3. though if the number is divisible by 3, say "fizz",
   and if the number is divisible by 5, say "buzz",
   and if the number is divisible by 3 and 5, say "fizzbuzz"
4. if an error is made, start again
5. stop when you reach 100

A sequence of well-defined operations is called an algorithm.

ALGORITHMS — solutions to problems —

have been around for a long time
use an existing one
adapt (revise, refactor) an existing one
develop a new one
can be carried out (implemented) by a person or a computer

Computational Problem Solving

TASK/PROBLEM ➔ Computational Thinking ➔
SOLUTION/ALGORITHM

SOLUTION/ALGORITHM ➔ Design/Coding ➔
COMPUTER PROGRAM

COMPUTER PROGRAM ➔ Testing/Debugging ➔
HIGH QUALITY COMPUTER PROGRAM

SOLUTION/ALGORITHM ➔ Design/Coding ➔
COMPUTER PROGRAM

COMPUTER PROGRAM ➔ Testing/Debugging ➔
HIGH QUALITY COMPUTER PROGRAM

HIGH QUALITY COMPUTER PROGRAM ➔ Execute ➔
AUTOMATIC, FAST, RELIABLE, REUSABLE SOLUTION
What is Computer Science?
Support for/computational problem solving

Theory
• automata theory
• algorithms & data structures
• complexity
• programming languages

Systems
• computer organization
• operating systems
• networks/high performance computing/security

Software Development/Engineering
• programming tools and techniques
• programming large, complex systems

Applied Computer Science
• data analytics
• computing + X (e.g., biology, linguistics, law, economics, etc.)

What can you expect from CIS 210, cont’d.
✓ Weekly readings, projects, exercises to support learning of computational problem solving and other computer science topics

Supported by
✓ Lectures — large group meetings
✓ Labs — small group meetings w/ computers
✓ Lab help hours/office hours — beyond class and lab
✓ Assessments, e.g., solutions, code feedback, exams

Welcome to CIS 210
• class/lab – 4
• text 1st reading/practice – 1
• review reading/class notes/project solution – 1
• projects – design/implement/test&debug – 6 (3 per project) [lab help hours]
Hofstadter's Law: It always takes longer than you expect, even when you take into account Hofstadter's Law.

Douglas Hofstadter, Gödel, Escher, Bach: An Eternal Golden Braid

CIS 210 / Welcome

210, 211, 212 Computer Science I, II, III

prereqs:  
- Math 112
- prior programming experience

familiar with programming fundamentals – variables, expressions, basic data types (e.g., numeric, boolean, string, collections), conditionals, loops, user-defined functions, parameter passing, ...

familiar with process of programming – time, bugs, technical glitches, multiple drafts ...

def testfunc(timeLeft, time):
  # prior programming experience
  # What does this function do?
  #
  while timeLeft > 0:
    print(timeLeft)
    timeLeft = timeLeft - 1
  if time == 'am':
    print('Good morning, CIS 210!')
  return None

What will be the result when the following code is executed?

```python
>>> testfunc(10, 'am')
```

# [calling a function/argument passing]
Welcome to CIS 210

✔ What is Computer Science?
✔ Computational Problem Solving
✔ What can you expect from CIS 210?
✔ Prior programming experience

Your questions

CIS 210 Computer Science I

Welcome to CIS 210

• CIS 210/Questions
• Hello, Python

A computer program implements an algorithm on a computer.

A computer program is (therefore) a set of instructions written in a language the computer can understand.

WHAT (KIND OF) LANGUAGE IS THAT?
A program is a set of instructions written in a language the computer can understand.

**What kind of language is that?**

0s and 1s?
Natural language?

<table>
<thead>
<tr>
<th>A</th>
<th>J</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>K</td>
<td>T</td>
</tr>
<tr>
<td>C</td>
<td>L</td>
<td>U</td>
</tr>
<tr>
<td>D</td>
<td>M</td>
<td>V</td>
</tr>
<tr>
<td>E</td>
<td>N</td>
<td>W</td>
</tr>
<tr>
<td>F</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td>G</td>
<td>P</td>
<td>Y</td>
</tr>
<tr>
<td>H</td>
<td>Q</td>
<td>Z</td>
</tr>
<tr>
<td>I</td>
<td>R</td>
<td></td>
</tr>
</tbody>
</table>

Ok Google ...
Hey Siri ...

A program is a set of instructions written in a language the computer can understand.

**What kind of language is that?**

0s and 1s?
Natural language?

*Programming language*
Python is a language (virtual machine) AND Python is a program (translator/interpreter)

**Computer levels of abstraction**

- Office suites
- Web browser
- Python
- Other applications
- Operating system (OS X, Windows, etc.)
- Hard drive
- Monitor

**Python (or any computer programming language) provides a notation (syntax, semantics) for writing out a computation as a series of steps.**

- Formal
- Precise
- Unambiguous
- Readable

Theoretical underpinnings of computer science: any “Turing complete” language has the same functionality as any other language.

Different programming languages -> easier or harder to accomplish different tasks.
Why Python?

- Python is widely used in many fields
- Accessible to entry level programmers and also for experts – like chess or tennis
- Interactive (interpreted), high-level, syntax-lite language - concentrate on problem-solving rather than the language itself
- Lots of built in functionality and support libraries (“batteries included”)
- General purpose, multiple paradigm language and syntax support straightforward transition to C, C++, Java
- Popular, well-supported, good documentation and development environments.

Python/IDLE

- Downloads along with Python
- Simple integrated development environment (IDE)
- Shell for exploring Python and testing bits of code
- Editor supports Python program development

Python Quick Overview

- Python language
- Python interpreter (program)

What sorts of Python language input does the Python interpreter recognize/”understand”?

- Python keywords, primitive elements, identifiers
What sorts of Python language input does the Python interpreter recognize/“understand”? 

→ Python keywords, primitive elements, identifiers 

**keywords**
read input data (input)
perform operations (evaluate)
store/retrieve intermediate values (variables)
decide how to proceed (control)
display result (output)
return result when the computation is finished

```python
def testfunc(timeLeft, time):
    # [input data is provided when function is called (executed)]
    What does this function do?
    while timeLeft > 0:  # [flow of control, Boolean operation]
        print(timeLeft)  # [display result]
        timeLeft = timeLeft - 1  # [variable/memory store]
    if time == 'am':  # [flow of control, Boolean operation]
        print('Good morning, CIS 210!')
        return None  # [return result of executing testfunc]
>> testfunc(10, 'am')
```

**keywords** define the syntax and structure of the Python language, for example

– special values (None)
– special operations (return)
– flow of control (if, while)
– special instructions (def)

**identifiers** are the names given to Python elements like variables and functions (time, timeLeft, print)

Quick Python Overview

3 important questions you should ask about any programming language:

• what are the primitive elements?
• how can we combine elements?
• how can we create our own elements?
many Python primitive elements are available at start up

```python
>>> 4
4
>>> len
<built-in function len>
>>> 'hello'
'hello'
>>> others are available in the Python standard libraries
```

### 2. combining primitive elements

```python
>>> 4 + 3
7
>>> len('hello')
5
>>> a = 4
>>> if a > 2:
    print('goodbye')
goodbye
>>> 
```

### 3. how can we extend the language – create our own elements?

- naming values (variables/assignment)
- user-defined functions

```python
>>> a = 4
>>> a
4
```

### 3. how can we extend the language – create our own elements?

- naming values (variables/assignment)
- user-defined functions

Python functions name an operation, e.g., built-in function `abs`

```python
>>> abs
<built-in function abs>
>>> abs()
```
Python Functions

Executing, or calling, a built-in function:

```python
>>> abs()
TypeError: abs() takes exactly one argument (0 given)
```

```python
>>> abs(7)
7
```

The general form of a function call is:

```python
<function name>(<arguments>)
```

```python
>>> abs(-7) >>> abs(99)
7 99
```

```python
>>> pow(3, 2) >>> pow(2, 5)
9 32
```

Python Functions

Executing, or calling, a built-in function:

```python
>>> abs()
TypeError: abs() takes exactly one argument (0 given)
```

```python
>>> abs(7) >>> abs(-7)
7 7
```

how can we extend the language – create our own elements?

✓ naming values (variables/assignment)

– user-defined functions

For example

```python
def twice(): # header incl. parameters
    """""" # docstring
    result = 3 * 2 # code
    return result # value to return
```
For example

def twice():
    # header incl. parameters
    # docstring
    result = 3 * 2
    return result

Defining a function is like defining a variable name –
the function name refers to the operations specified
in the function definition

Calling a user-defined function is the same as calling a Python built in function

```python
twice()
```

```python
3 * 2
```

Does this code do what it is supposed to do?

```python
def et(i,n):
    sd = 10000
    et = (i-sd-(4050*n))*.2
    return et

>>> et(35000, 2)
```

Generate an estimate for federal income tax based on reported income and number of exemptions. The standard deduction ($10,000) and standard exemption ($4,050) are always used and the tax rate is assumed to be 20%. Return the estimated tax.

```python
def et(i,n):
    sd = 10000
    et = (i-sd-(4050*n))*.2
    return et

>>> et(35000, 2)  # work an example or two offline
```

20,000 gross income
20,000 – 10,000 = 10,000 subtract standard deduction
10,000 – (4,050 * 2) = 1900 subtract exemptions
1,900 taxable income
1,900 *.2 is tax ($380)

→ write an algorithm
def et(i, n):
    sd = 10000
    et = (i-sd-(4050*n))*.2
    return et

>>> et(35000, 2)