CIS 210 Computer Science I / Welcome

**CIS 210 Winter 2017**
Introduction to Computer Science

<table>
<thead>
<tr>
<th>Objectives</th>
<th>Welcome to CIS 210! Computational Thinking-task/problem-solving process, and programming the solutions to those problems. (MR p. 330)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIS 210 is a required course for CS majors. CS majors will usually take CIS 210 senior year with the three or four advanced courses.  Use CIS 210 as a mini pre-course for CS 4.</td>
<td></td>
</tr>
<tr>
<td>Students in other majors are also welcome in CIS 210, with or without the discrete mathematics course. Computational thinking has applications in many fields and computational problem-solving is a powerful tool for everyday life and even personal use.</td>
<td></td>
</tr>
<tr>
<td>Prerequisites</td>
<td>What to do after the class is over?</td>
</tr>
<tr>
<td>Instructor</td>
<td>How do I get started?</td>
</tr>
<tr>
<td>Assistant</td>
<td>Who is responsible for the course?</td>
</tr>
</tbody>
</table>

**CIS 210 / Welcome**

**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

TASK/PROBLEM $\rightarrow$ Computational Thinking $\leftrightarrow$
SOLUTION/ALGORITHM

SOLUTION/ALGORITHM $\leftrightarrow$ Design/Coding $\leftrightarrow$
COMPUTER PROGRAM

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.)
### CIS 210 / Welcome

**What can you expect from CIS 210**

- **CIS 210 Focus: Computational Problem Solving**
  - Depth and breadth intro to computer science
  - Design and implement algorithms
  - Develop excellent programming skills
  - Introduction to Computer Science topics

### CIS 210 Computer Science I

**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 and office hours
- Assessments, e.g., solutions, feedback, exams

### CIS 210 Computer Science I

**What can you expect from CIS 210**

- Lecture + Labs + Text/Readings + Lab Help Hours
  - computational problem solving (weekly projects)
  - and other programming practice, e.g.,
    - additional exercises from text
    - Turingscraft
    - online resources (see class website references page)
  - Lecture + Labs + Text/Readings + Lab Help Hours + Projects + Posted Solutions
  - ARE THE INPUTS FOR
  - midterm and final exams

### CIS 210 Computer Science I

**Welcome to CIS 210!**

- What is Computer Science?
- Why study computing/programming?
- What can you expect from CIS 210?
  - Prior programming experience

**Your questions**

- Getting started
CIS 210 / Welcome

210, 211, 212 Computer Science I, II, III
prereqs:  - Math 112
- prior programming experience

familiar with 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 ...

CIS 210 / Welcome

def testfunc(timeLeft, time):
    # prior programming experience
    """What does this function do?"
    """# [defining a function/parameters]
    while timeLeft > 0:
        print(timeLeft)
        timeLeft = timeLeft-1
        if time == 'am':
            print('Good morning, CIS 210!')
    return None
    # [functions return values]

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

>>> testfunc(5, 'am')

CIS 210 / Welcome

def testfunc(timeLeft, time):
    # prior programming experience
    """What does this function do?"
    """# [defining a function/parameters]
    while timeLeft > 0:
        print(timeLeft)
        timeLeft = timeLeft-1
        if time == 'am':
            print('Good morning, CIS 210!')
    return None
    # [functions return values]

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

>>> testfunc(5, 'am')

```python
CIS 210 / Welcome

def testfunc(timeLeft):
    """What does this function do?"
    while timeLeft > 0:
        print(timeLeft)
        timeLeft = timeLeft-1
    return None

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

```
def trackscore(score):
    '''convert score to grade point; return grade point'''
    gradepoint = 0
    if score >= 90:
        gradepoint = 4
    if score >= 80:
        gradepoint = 3
    if score >= 70:
        gradepoint = 2
    if score >= 60:
        gradepoint = 1
    return gradepoint

What is the result of executing the following code:
>>> gp = trackscore(90)
>>> gp
4

- Programming = Computational Thinking + Coding

An algorithm is a step by step procedure for solving a problem or accomplishing some end.

A computer program implements an algorithm on a computer.

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

Which language is that?

- Natural language?
- Flip switches?
- Programming language

Programming languages like Python are

- Formal
- Precise
- Unambiguous
- Readable

computer levels of abstraction

Why Python?

- Accessible to entry level programmers and also for experts – like chess or tennis
- Python is widely used in many fields
- Interactive, 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++, Java
- Popular, well-supported, good documentation and development environments.
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?

what are Python’s primitive elements?

That is,

what sorts of input does (the) Python (program) recognize/"understand"?

There is only one kind of primitive element in Python

OBJECTS

• what are Python’s primitive elements?

OBJECT
type
value
id (memory location)
PYTHON OBJECTS

type – range of values and operations

Example of a Python object:

```python
>>> 99
99
>>> type(99)
<class 'int'>
>>> id(99)
2276160
```
3 questions:
✓ what are the primitive elements?
   Objects, with types that determine range of values and operations for that object

• how can we combine elements?
  -- with expressions

expressions are combinations of values (operands) and operators, that can be evaluated and return a result

For example,
>>> 99 + 100  
199

>>> 99.9 + 100  
199.9

>>> 99 + 100  
199

>>> 99.9 + 100  
199.9

• how can we combine elements?
  -- expressions

For example,
>>> 99 + 100  
199

>>> 99.9 + 100  
199.9

expression are combinations of values (operands) and operators, that can be evaluated and return a result