CIS 122 - Introduction to Programming and Problem Solving - Spring 2020

Learn to **solve problems** by running a computer program you have written.

Note what **data** and "recipes" or **algorithms** to use to solve a well-defined problem.

Learn to **check** whether your solution is valid.

You will learn the **Python 3** progamming language.

Cmputer programs communicate with::

- A **machine** (a computer)
  - — pay attention to detail because a **computer follows your orders** but has **no understanding** — **none** — about what it is doing.

- A **person** — describe the **purpose** and techniques and data your program uses.

Python is now the most widely used programming language for machine learning and data science, and finds widespread use in science.

Corona Virus

I plan on an entire term taking place **online**. If we are able to go back to standard classes, we will still have the materials online. Stay patient and resilient, we will work out snags as they pop up.

In the event of a campus emergency that disrupts academic activities, course requirements, deadlines, and grading percentages are subject to change. Information about changes in this course will be communicated as soon as possible by email, and on Canvas. If we are not able to meet face-to-face, students should immediately log onto Canvas and read any announcements and/or access alternative assignments. Students are also encouraged to continue the readings and other assignments as outlined on this syllabus or subsequent syllabi.

Instructor
Dave Wilkins **davew@cs.uoregon**.edu or **dwilkins@uoregon.edu**
Class (Live presentations over the internet) 9 -9:50 a.m. MWF
"Office hours" equivalent MWF 2 to 6 PM (Email and Canvas conversations)

Teaching Assistants (Graduate Educators)
Sam Schwartz **sam@cs.uoregon.edu**
Hours (not yet available)

Toby Harvey **tharvey@cs.uoregon.edu**
Hours (not yet available)

Learning Assistants (undergraduates with solid Python experience)
Amy Reichhold **areichh2@uoregon.edu**
River Veek **riverv@uoregon.edu**
Patrick Thompson **prthomasma@gmail.com**

Textbook

Think Python: How to Think Like a Computer Scientist, **2nd Edition**, **Allen B. Downey**,O'Reilly Media, 2nd edition December 28, 2015. IBSN-13: 978-1491939369, ISBN-10: 1491939362

Download Think Python 2e (2nd edition) for **free** from the author's GreenTea website:
http://greenteapress.com/wp/think-python-2e or search for **green tree press think python-2e**
Allen Downey states "Think Python 2e is a Free Book. It is available under the Creative Commons Attribution-NonCommercial 3.0 Unported License, which means that you are free to copy, distribute, and modify it, as long as you attribute the work and don't use it for commercial purposes."

Also available on **Canvas**

Course Topics

Install Python 3.x See the notes in Canvas on what to do.
Do this before class begins.

The basics in 5 weeks

The first 5 weeks cover most of the basics of programming
Know everything we work on well and you are good for a grade in the C range.

Week 1
Introduction to programming a computer; why Python;
Your program must communicate with a machine and with other humans. Save and Run your program.
IDLE, # comments, **variables**, types of data, **print( )** to display data, **errors**, errors, errors
round decimal numbers (called **float** type)

Week 2
if and **else** allow a computer to change what it is doing and do something different.
indented block of statements – group statements
debug strategies
input( ) always returns strings; getting numbers input int( ) and float( )
some new **errors**
comparisons **== > >= < <= !=**
Check to see if a value is **in** a static list

Week 3
for to repeat some actions a number of times; **for** often creates a new variable
debug repeated actions
2 variations:

```
1) # use a number to show repetitons
   for some_variable in range(3):
2) how_many = 5
   for some_variable in range(how_many):
```

some new **errors**

Week 4
functions
Use a function ("**call**" the function) to compute some value

define a function to work on the given data and return an answer to the caller.
debug a function you wrote
To tackle a big, tough problem, break it into several smaller, simpler sub–problems.
Function definitions let you write your programs out of a series of functions.

Week 5
Midterm

Putting all the pieces together
- **functions**
  - functions built in to Python
  - functions you write
- **for** (to repeat actions)
- **if** (to choose between two courses of action)
- **variables** (to store values, to update stored values)
- **debug** techniques
- display results using **print( )**

Week 6
Repeat using **while**
Repeat using **for** item **in** some_list**:**
random numbers for simulated coin flips and dice rolls
Lists: starting from empty lists, appending new data to end of list

Week 7
Functions that just do something instead of returning an answer
A model of calling data, and what data a function can use
Best practice with functions
Drawing using **turtle graphics functions**

Choosing from a series of possiblitlies with **if elif elif ... else**
and using **and or**

Week 8
Read a text file to bring some data into memory in your program
Store the data in lists, retrieve data by an index number (similar to using an apartment number)
Working with parts of a list, called a **slice** of a list; **change** items in a list.

Week 9
Dictionaries of key : value pairs provide a very flexible and very very fast access to data
Use dictionaries to keep counts of items read from a file
Use of boolean (True / False) variables with while and if statements

Week 10
Deal with **errors** in a program – how to **continue running** the program **after an error**
Review entire term
Tackle a challenge such as: Count word usage in an entire book or Shakespeare play; create working calendar functions (example: what day of the week is Sep 17, 2024?); simulate an online vendor.

Final Exam

Covers all topics from entire term

Special accommodations needed?

Our goal is student mastery of programming.

We respect and will accommodate special student needs.

Give priority to your own good physical and mental health. Email your instructor or TA if you are unable to take a test or complete an assignment.

Please let us know if we introduce an idea without enough examples to let you understand what is going on.

Online Etiquette

Follow the Golden Rule:

treat others as you would like to be treated

Do's and **Don'ts**

Do submit work **ontime**, even if it's not working right.

Exception: You are ill and cannot work on your project.

Do seek help from Teaching Assistant or Learning Assistants or Instrutor

Don't copy someone else's work.

Do start your program with one of these comments

Example assuming your name is Taylor Smith

```
# Solo work  by Taylor Smith
    or (example when you collaborated with your friend Jay Ling)
# Taylor Smith worked with Jay Ling
```

Grades

A Solid mastery of all syllabus topics *A+ rarely given, only for clearly exceptional work*

B Good mastery of most syllabus topics
Thorough mastery of 1st 5 weeks topics

C Thorough mastery of 1st 5 weeks topics

Reworks available to allow you time to work with basic topics you may have had trouble with.