

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

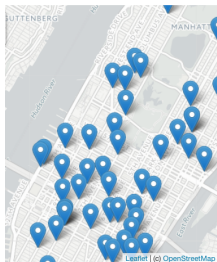
Syllabus

CSci 127: Introduction to Computer Science

Catalog Description: 3 hours, 3 credits: This course presents an overview of computer science (CS) with an emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners. Other topics include: organization of hardware, software, and how information is structured on contemporary computing devices. This course is pre-requisite to several introductory core courses in the CS Major. The course is also required for the CS minor. MATH 12500 or higher is strongly recommended as a co-req for intended Majors.

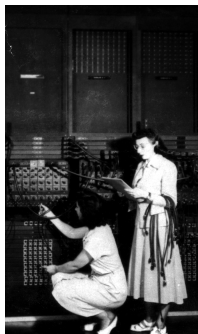
(Show syllabus webpage)

Syllabus: Topics



- **This course assumes no previous programming experience.**
- “... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners...”
- Organized like a fugue, with variations on this theme:
 - ▶ Introduce coding constructs in Python,
 - ▶ Apply those ideas to different problems (e.g. analyzing & mapping data),
 - ▶ See constructs again:
 - ★ for logical circuits,
 - ★ for Unix command line interface,
 - ★ for the markup language for github,
 - ★ for the simplified machine language, &
 - ★ for C++.

Class Structure



First “computers”
ENIAC, 1945.

Lecture:

- Tu, Th 1:30PM-3:04PM 510 Hunter North
- Lecture Slips: only help your grade, final exam replaces incomplete or missing slips
- Mix of explanation, challenges, & group work.
- Hard to ask questions in a large lecture, ask UTAs & instructors during group work.

Recitation Section:

- ① Quiz: final exam replaces low/missing quizzes.
- ② Brief overview of lab & programs for the week.
- ③ One-on-one help with instructors & UTAs.

Software Platforms:

- Blackboard: visit ICIT for access issues.
- Gradescope: email invite sent Sunday.

Introductions: Your Turn



- Introduce yourself to two classmates (that you have not met before).
- Write down names & interesting fact on lecture slip.

Today's Topics



- Introduction to Python
- Definite Loops (for-loops)
- Turtle Graphics
- Algorithms

Introduction to Python



- We will be writing programs– commands to the computer to do something.
- A **programming language** is a stylized way of writing those commands.
- If you can write a logical argument or persuasive essay, you can write a program.
- Our first language, Python, is popular for its ease-of-use, flexibility, and extendibility.
- The first lab goes into step-by-step details of getting Python running.
- We'll look at the design and basic structure (no worries if you haven't tried it yet in lab).

First Program: Hello, World!



Demo in pythonTutor

First Program: Hello, World!

```
#Name: Thomas Hunter
```

← *These lines are comments*

```
#Date: September 1, 2017
```

← *(for us, not computer to read)*

```
#This program prints: Hello, World!
```

← *(this one also)*

```
print("Hello, World!")
```

← *Prints the string "Hello, World!" to the screen*

- Output to the screen is: Hello, World!
- Can replace Hello, World! with another string to be printed.

Variations on Hello, World!

```
#Name: L-M Miranda
#Date: Hunter College HS '98
#This program prints intro lyrics

print('Get your education,')
print("don't forget from whence you came, and")
print("The world's gonna know your name.")
```

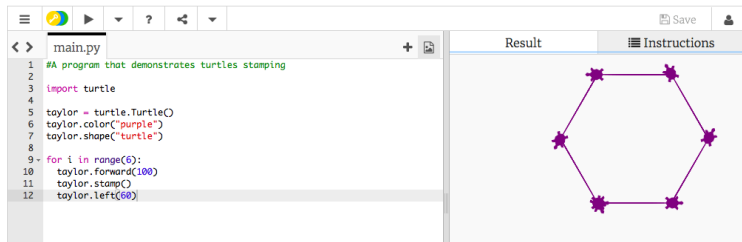
- Each print statement writes its output on a new line.
- Results in three lines of output.
- Can use single or double quotes, just need to match.

Turtles Introduction



- A simple, whimsical graphics package for Python
- Dates back to Logos Turtles in the 1960s
- (Demo from webpage)
- (Fancier turtle demo)

Turtles Introduction



The screenshot shows a Python IDE with a code editor on the left and a result window on the right. The code editor contains the following Python code:

```
1 #A program that demonstrates turtles stamping
2
3 import turtle
4
5 taylor = turtle.Turtle()
6 taylor.color("purple")
7 taylor.shape("turtle")
8
9 for i in range(6):
10     taylor.forward(100)
11     taylor.stamp()
12     taylor.left(60)
```

The result window on the right shows a purple hexagon with a turtle-shaped stamp at each of its six vertices.

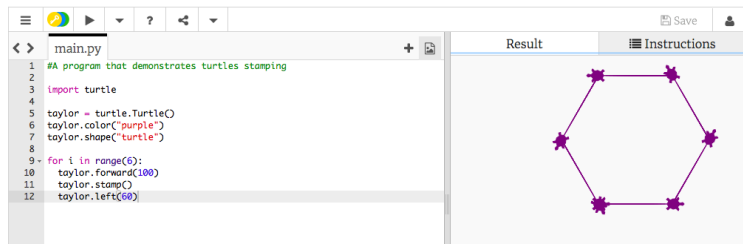
- Creates a turtle, called `taylor`
- Changes the color (to purple) and shape (to turtle-shaped)
- Repeats 6 times:
 - Move forward; stamp; and turn left 60 degrees

Group Work

Working in pairs or triples:

- ① Write a program that will draw a 10-sided polygon.
- ② Write a program that will repeat the line:
`I'm lookin' for a mind at work!`
three times.

Decagon Program



The screenshot shows a Python IDE with a code editor on the left and a result window on the right. The code editor contains the following Python code:

```
1 #A program that demonstrates turtles stamping
2
3 import turtle
4
5 taylor = turtle.Turtle()
6 taylor.color("purple")
7 taylor.shape("turtle")
8
9 for i in range(6):
10     taylor.forward(100)
11     taylor.stamp()
12     taylor.left(60)
```

The result window on the right shows a purple hexagon with a turtle stamp at each vertex.

- Start with the hexagon program.
- Has 10 sides (instead of 6), so change the `range(6)` to `range(10)`.
- Makes 10 turns (instead of 6), so change the `taylor.left(60)` to `taylor.left(360/10)`.

Work Program

- ② Write a program that will repeat the line:

```
I'm lookin' for a mind at work!
```

three times.

- Repeats three times, so, use `range(3)`:

```
for i in range(3):
```

- Instead of turtle commands, repeating a print statement.
- Completed program:

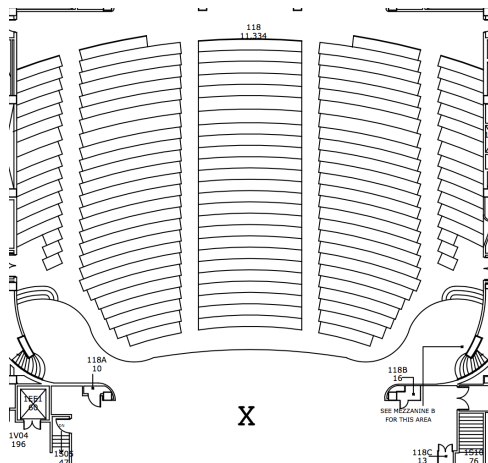
```
# Your name here!  
for i in range(3):  
    print("I'm lookin' for a mind at work!")
```

What is an Algorithm?

From our textbook:

- An **algorithm** is a process or set of rules to be followed to solve a problem.
- Programming is a skill that allows a computer scientist to take an algorithm and represent it in a notation (a program) that can be followed by a computer.

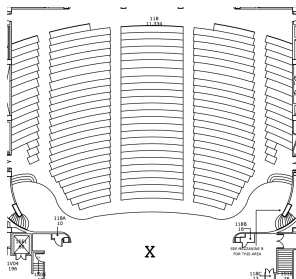
Group Work



Working in pairs or triples:

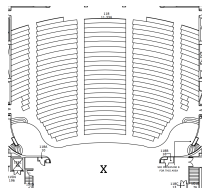
- ① On the floorplan, mark your current location.
- ② Write step-by-step directions to get to/from X.

Group Work



- Find another group, near you, that's going in the "opposite" way.
- Follow the directions to get to X.
- Follow the other set of directions from X back to your seat.
- Annotate any changes needed to the directions.

Recap



- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.
- In Python, we introduced:
 - ▶ `strings`, or sequences of characters,
 - ▶ `print()` statements,
 - ▶ `for`-loops with `range()` statements, &
 - ▶ `variables` containing turtles.