Moore's Law

Computers are built from *transistor circuits* that appear on a fingernail-sized device known as an *integrated circuit* (IC for short).

An IC is also called a *chip*, due to being sliced from a larger chunk of silicon.

Since the invention of the IC around 1960, IC's have doubled in *circuit capacity* about every 1.5 or 2 years, a trend known as **Moore's Law**.

Gordon Moore, a co-founder of Intel, made that prediction in 1965.

Doublings grow surprisingly rapidly:
2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, …

Moore’s Law: clock frequency

A key improvement resulting from Moore's Law is that over the years the same-size chip becomes higher speed.

Speedup occurs in part due to higher clock frequency:

- smaller transistors can change between 0 and 1 more quickly
- smaller transistors means circuits are more compact so an IC's clock signal need travel less distance

Typical clock frequencies in the 1980's were about 10 MHz.

Typical frequencies in the 2010's are over 1 GHz, meaning over 100 times faster.

Hardware trends: server farms

More and more computers are being built to act as servers to serve information over the web.

Companies like Google, Microsoft, Facebook, and Amazon have giant collections of servers.

A *server farm* is a collection of thousands of servers, typically located in large buildings specially equipped to provide electrical power, cooling, physical security, and accessibility for repairs.

Some of these buildings are as large as a football field, and thus *server farms* are often built in remote areas where land is cheap.

Hardware trends: server farms

A *server farm* may use large amounts of electricity to power the servers and to cool the servers.

Imagine 2,000 servers, each requiring 500 watts!
That will require 2,000 * 500 = 1 megawatt of power.

Thus, server farms are often built near cheap electric sources, such as near a river having a hydroelectric power plant.

Hardware trends: multicore and graphics

A computer chip in the 1980s through early 2000s typically held just one CPU.

Today the main chip commonly holds two, four, eight, or more CPUs.

A *dual-core chip* has two CPUs on the chip, each CPU known as a *core*, having appeared first around 2005.

A *quad-core chip* has four CPUs on the chip.

8-core and *16-core chips* appeared around 2015, initially used mostly in servers, then making their way into desktop and laptop computers.

Hardware trends: multicore and graphics

Generating graphics or video on a screen, such as for a video game or for a movie, requires special computations that *must occur quickly* for good viewing quality.

A *graphics processing unit* or GPU is a special-purpose CPU chip whose hardware is *specially designed for fast graphics and video processing*.

Many computers, laptop, tablets, and smartphones today come with built-in graphics processors.

Hardware trends: embedded computers

Computer chips are small and thus can be used to improve electronic device functionality.

An *embedded computer* is a computer embedded inside some electronic device, like inside a car.

In the 1970s, devices like washing machines and coffee makers had computers embedded.

In the 1980s and 1990s, hundreds of electronic devices became heavily computerized, like televisions, copy machines, cars, jet aircraft, cameras, and more.

Today about 25% of the *cost of a car* is for its computers, and those computers involve large programs with many millions of instructions.

Programming: Machine Language

CPU (*Central Processing Unit*) understands only 0's and 1's.

Machine language instructions are composed of 0's and 1's

Let's see participation activity 2.8.2 in ZyBooks

Programming: Assembly language

Assembly language is a textual human-understandable representation of a machine language's 0's and 1's, as in:

Add M[5] M[6] M[7].

A program called an *assembler* automatically converts an assembly language program into machine language.

Let’s see participation activity 2.9.1 in ZyBooks

Programming: High-level language

A *high-level language* is a programming language having higher-level instructions than assembly language, enabling greater programmer productivity.

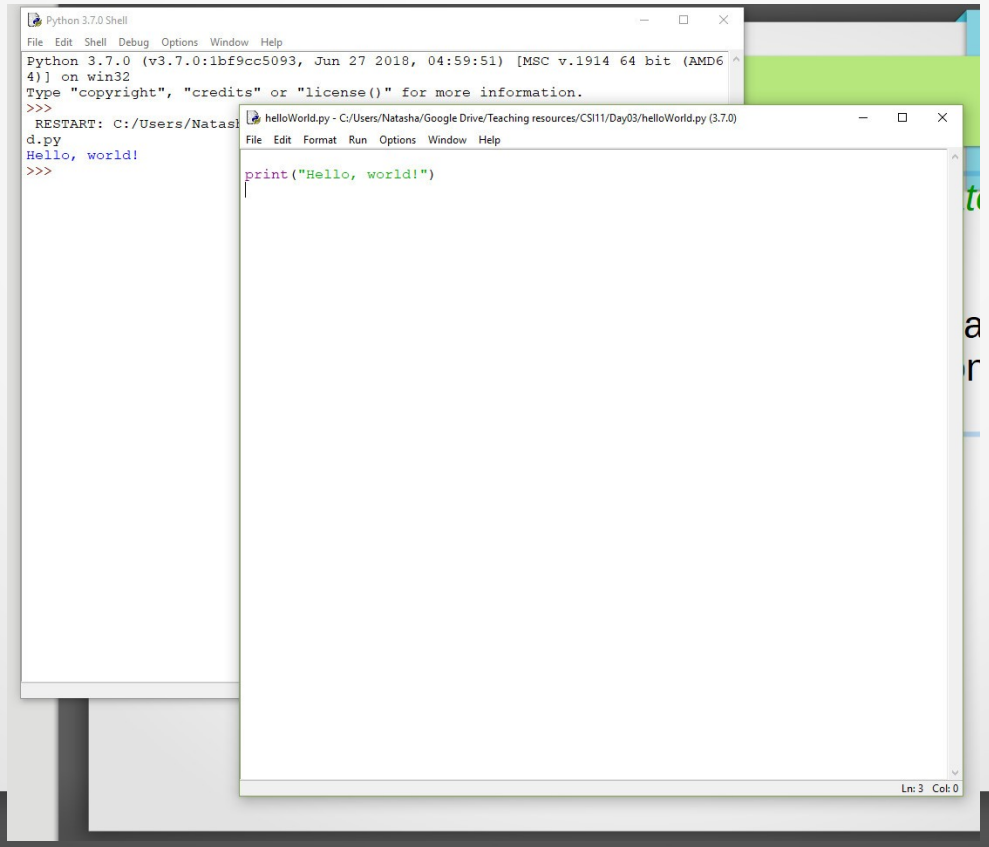
The first mainstream high-level language was Fortran, from IBM in 1957, short for *Formula Translation*.

Compilers and *interpreters* convert a high-level language to assembly/machine language.

Development Environment

Code development is usually done with an *Integrated Development Environment*, or *IDE*.

There are various IDEs, we will be using the official Python IDE that is distributed with the installation of Python, called *IDLE*.

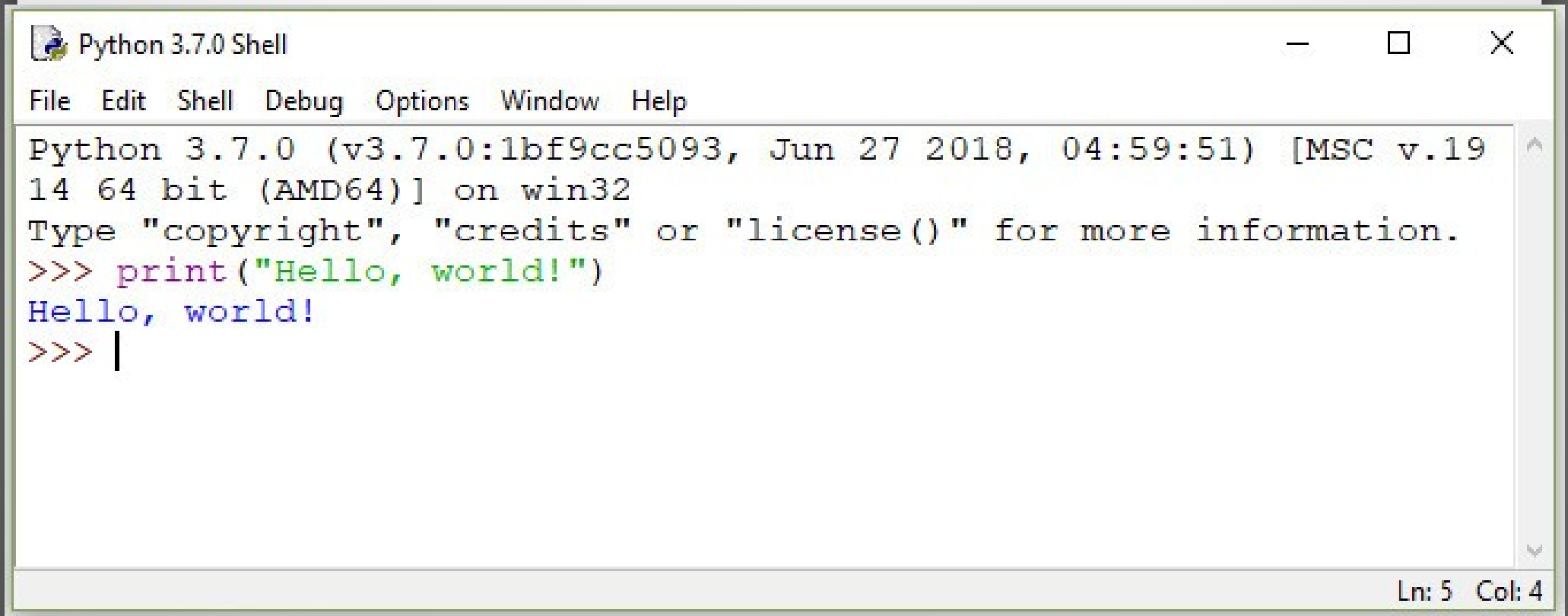


Development Environment

- Demonstrate IDLE
- Discuss Python Interpreter
- Discuss Python Shell (line prompt, …)
- Discuss File Editor (pyton files have extension **.py**)

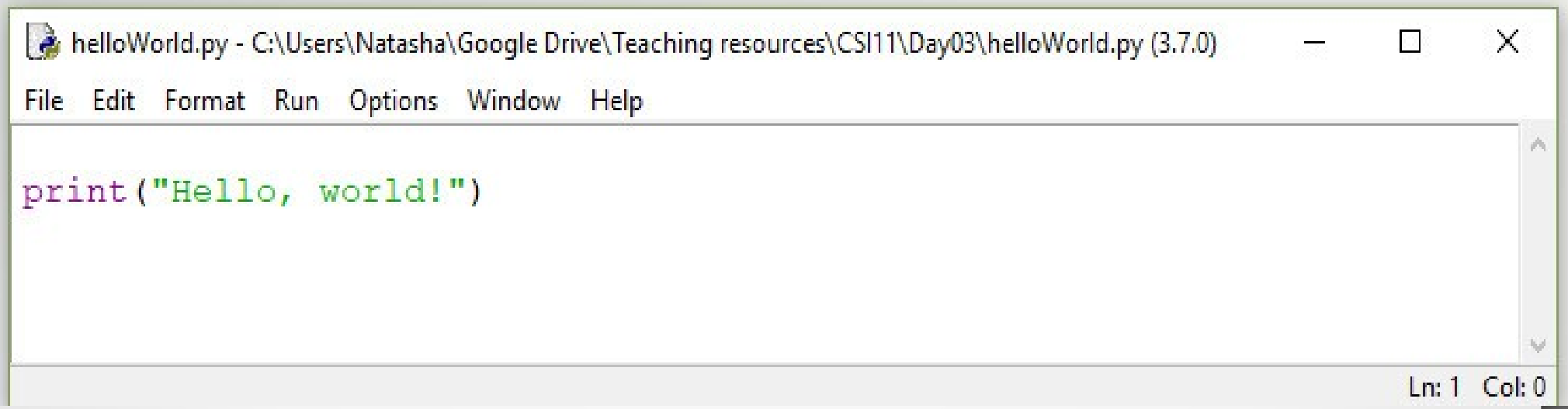
My first program

In Python shell:



My first program

In IDLE's file editor:



Save the program (File → Save) as a file named helloWorld.py

Then press F5 or go to Run → Run Module

Then check what you see in Python shell...

My second program

Create a new file (File → New File) and type in the following:

```
# this is my second program!

print("   *")
print(" * | *")
print("  \\*/")
print("*-***-*")
print("  /*\\")
print(" * | *")
print("   *")
print("Do you like my snowflake?")
```

Comment:
␣ denotes one space (*whitespace*)

Save the program (File → Save) as mySecondProgram.py
Then press F5 or go to Run → Run Module
Then check what you see in Python shell...

My second program

```
# this is my second program!

print("   *")
print(" * | *")
print("  \\*/")
print("*-***-*")
print("  /*\\")
print(" * | *")
print("   *")
print("Do you like my snowflake?")
```

comment
(starts with #)

`print()` method displays
variables or expression values

text enclosed in
quotes is known as
a *string literal*

Each print statement will output on a *new line*,
unless directed otherwise by a previous print statement

In-class practice:

See the handout.
Let’s work on it!