Outline

1. Computer Architecture
   - hardware components
   - programming environments

2. Getting Started with Python
   - installing Python
   - executing Python code

3. Number Systems
   - decimal and binary notations
   - running Sage

4. Summary + Assignments

MCS 260 Lecture 2
Introduction to Computer Science
Jan Verschelde, 13 January 2016
A computer system consists of

1. **Hardware**: physical components of computer
   - computer: processor, memory, bus, ...
   - peripherals: printer, screen, keyboard, mouse, ...

2. **Software**: programs executed by computer
   - basic software like the Operating System (OS)
     - either Unix (e.g.: Solaris, GNU-Linux, Mac OS X)
     - or Windows (the OS of Microsoft)
   - application software such as IDLE, Sage, ...
   - application software needs operating system to run
1 Computer Architecture
   • hardware components
   • programming environments

2 Getting Started with Python
   • installing Python
   • executing Python code

3 Number Systems
   • decimal and binary notations
   • running Sage

4 Summary + Assignments
Hardware Components

**processor** (or CPU: Central Processing Unit) does the computing and coordinates data transfer

**memory** (or RAM: Random Access Memory) is used to store data and programs,
- of limited capacity, and
- volatile (lost if power off).

**storage** persistently stores large quantities of data and programs,
- slower access to storage than to memory,
- but larger than RAM

**peripherals** are used to communicate with computer

**system bus** connects CPU, RAM, storage, and peripherals
1. **Computer Architecture**
   - hardware components
   - programming environments

2. **Getting Started with Python**
   - installing Python
   - executing Python code

3. **Number Systems**
   - decimal and binary notations
   - running Sage

4. **Summary + Assignments**
Programming Environments
what it takes to run programs

**editor**: is used to write source code

**compiler**: translates source code into an object, an executable program — if code is bug free

**interpreter**: executes high level code directly

**linker**: combines several objects into one single executable program

**debugger**: helps user to locate bugs, allowing a stepwise execution of the program

use an IDE: Integrated Development Environment
Python’s IDE is called IDLE
Executing Programs
how programs are executed

- high level programming languages are oriented towards the convenience of the programmer
- an assembler language offers symbols to the basic instructions for writing machine code

The Python Virtual Machine:

![Diagram showing the Python Virtual Machine process]

The Python interpreter creates bytecode that is then executed by the Python Virtual Machine at runtime.
Alan Mathison Turing
computer pioneer, 1912-1954

- Introduced in his 1936 paper ‘on computable numbers’ the universal computing machine, now known as the Turing machine.

- Created the Turing test in 1950, can a computer imitate intelligence?

- The A.M. Turing award is the ACM’s most prestigious award, the ‘Nobel Prize’ of computing.

Image taken from www.alanturing.net.
1. Computer Architecture
   - hardware components
   - programming environments

2. Getting Started with Python
   - installing Python
   - executing Python code

3. Number Systems
   - decimal and binary notations
   - running Sage

4. Summary + Assignments
Installing Python

on your desktop or laptop

Free to download from www.python.org.

Unix  most Linux distributions have Python installed, or else
      contact your system administrator.

Mac OS X  like with unix you can download the source
           or run Python 3.5 for Macintosh OS X.

Windows  run the Python 3.5 windows installer.

Computers in ACCC labs, in particular 2263 SEL,
have Python installed.
1. **Computer Architecture**
   - hardware components
   - programming environments

2. **Getting Started with Python**
   - installing Python
   - executing Python code

3. **Number Systems**
   - decimal and binary notations
   - running Sage

4. **Summary + Assignments**
Executing Python code

program prints "hello world!" Ways to run python programs:

1. In a Python session, type commands at the prompt:
   ```python
   >>> print('hello world!')
   ```

2. Running programs at the command prompt:
   1. Save Python commands in a file, e.g.: `hello.py`.
   2. Type `python hello.py` at the command prompt.

3. On windows, double click a file with .py extension.

4. In IDLE, go to the run menu in the editor.
Interactive Python code

Let us write a program that asks for our name, as input. And then, as output, writes hello followed by our name.

input The `input()` function accepts input as text:

```python
>>> NAME = input('Who\'s there ? ')
```

Displays *Who’s there ?* on screen and assigns what the user types in to the variable `NAME`.

output With `NAME` in its argument, the `print` command displays the value of `NAME`:

```python
>>> print('hello', NAME , '!
```

*develop Python code interactively at the prompt*
our first interactive program

The file `hello_there.py` contains

```python
# L-2 MCS 260 an interactive program
NAME = input('Who\'s there ?
print('hello', NAME , '!')
```

The # signs the start of a comment, the line following # is ignored by the interpreter.

At the command prompt, we type

```
python hello_there.py
```

On Windows, double clicking on the file with `.py` extension will execute the program.
Computer Architecture

first steps with Python

1. Computer Architecture
   - hardware components
   - programming environments

2. Getting Started with Python
   - installing Python
   - executing Python code

3. Number Systems
   - decimal and binary notations
   - running Sage

4. Summary + Assignments
Decimal Notation of Numbers

- The value of $284 = 2 \times 10^2 + 8 \times 10^1 + 4 \times 10^0$. The digits 2, 8, 4 are the digits of the number, 10 is the base. The position of each digit determines its contribution to the value of the number.

- For any base $B$, a number $n$ is denoted by $m$ coefficients $c_i$, $i = m, m - 1, \ldots, 1, 0$, $0 \leq c_i < B$:

  $$ n = c_m B^m + c_{m-1} B^{m-1} + \cdots + c_1 B^1 + c_0 B^0. $$

- From base five to decimal notation:

  $$ 2104_5 = 2 \times 5^3 + 1 \times 5^2 + 0 \times 5^1 + 4 \times 5^0 $$

  $$ = 250 + 25 + 0 + 4 $$

  $$ = 279_{10} $$
Binary Numbers

- The base is two, the coefficients are bits $\in \{0, 1\}$.

- The first 16 natural numbers — need 4 bits:
  
  $\begin{align*}
  0000 &= 0 & 0001 &= 1 & 0010 &= 2 & 0011 &= 3 \\
  0100 &= 4 & 0101 &= 5 & 0110 &= 6 & 0111 &= 7 \\
  1000 &= 8 & 1001 &= 9 & 1010 &= A & 1011 &= B \\
  1100 &= C & 1101 &= D & 1110 &= E & 1111 &= F \\
  \end{align*}$

- The hexadecimal ‘digits’ are 0,1,2,…,9,A,B,C,D,E,F.

- It is straightforward to convert binary into hexadecimal and hexadecimal into binary numbers.
Converting Numbers
from decimal to binary

Convert 123 into binary format:

<table>
<thead>
<tr>
<th>$n$</th>
<th>$n/2$</th>
<th>$n \mod 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>61</td>
<td>1</td>
</tr>
<tr>
<td>61</td>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
123 = 1 + 2 \times 61 = 1 + 2 \times (1 + 2 \times 30) \]
\[
= 1 + 2 \times (1 + 2 \times (0 + 2 \times 15)) \]
\[
= 1 + 2 \times (1 + 2 \times (0 + 2 \times (1 + 2 \times 7))) \]
\[
= \ldots \]

So $123 = 1111011 = 7B$. 
Flowchart
conversion algorithm

n = input()

r is remainder of n/2
print r; n := n/2

n == 0?

No

Yes stop
Computer Architecture

first steps with Python

1. Computer Architecture
   - hardware components
   - programming environments

2. Getting Started with Python
   - installing Python
   - executing Python code

3. Number Systems
   - decimal and binary notations
   - running Sage

4. Summary + Assignments
Running Sage

Ways to compute with Sage:

- a notebook blends commands with commentary
  - graphical user interface
  - runs in a browser

- command line use if no graphical output is needed
  and more dedicated to computationally intensive jobs

- language is Python, with some variation
  Python (version $\geq 3.0$):

  $\begin{align*}
  \text{>>> 68/25} & \quad \text{>>> 68/25} \\
  2.72 & \quad 2
  \end{align*}$

Sage:

```
sage: 68/25
68/25
```
Hexadecimal conversions in Python with the % operator:

```python
>>> "%X" % 123
'7B'
>>> '%x' % 123
'7b'
```

The same calculations are possible in Sage.

Sage has much more refined number systems:

1. integers and modular arithmetic
2. exact calculations with rational numbers
3. arbitrary precision float and complex
4. algebraic numbers
Summary

In this lecture, we

1. wrote our first *Hello world!* program; and
2. looked at the binary and hexadecimal number systems.

Recommended reading:

- Python tutorial on [http://docs.python.org/2/tutorial](http://docs.python.org/2/tutorial)
- sections 1.4 & 1.5 of *Computer Science. An Overview*
- chapter 1 of *Python Programming in Context*
Assignments

1. Explain the main difference between installing software \textit{from source} and installing a \textit{binary} of the software.

2. Write a script that asks the user for the first name and then in a second question for the last name. The script then prints a personalized greeting, e.g.: \textit{Hello John Doe!} if the user entered \textit{John} as answer to the first and \textit{Doe} as answer to the second question.

3. Given the base and a sequence of coefficients of a number, draw the flowchart of the algorithm to evaluate the number.

4. Write down pseudocode for the algorithm to compute the binary representation of a number.

5. Compute examples of general number conversions from any base to any other base.

6. What is the algorithm for such general conversions?