LECTURE 1

INTRO & NUMBER SYSTEMS

MCS 260 Fall 2021

Emily Dumas

REMINDER: MASKS REQUIRED

😷 ✓

😐 ✗

MCS 260: INTRO TO COMPUTER SCIENCE

- Professor: Emily Dumas ⟨ddumas@uic.edu⟩
- TA: Johnny Joyce ⟨jjoyce22@uic.edu⟩
- TA: Kylash Viswanathan ⟨kviswa5@uic.edu⟩

IMMEDIATE ACTION ITEMS

- Read the syllabus on the Blackboard course site.
- Check the blackboard course site regularly.

TYPES OF WORK

	Frequency	Graded?	Collaborate?
Worksheets	Weekly	No	Yes!
Homework	Weekly	Yes	No
Projects	4 times	Yes	No

Notice that all graded work is to be done *individually*.

PYTHON

Python is a computer programming language.

- #2 most popular programming language in TIOBE
- Extensive use at Google, Dropbox, Instagram, Netflix, ...
- #1 most popular (by far) in a 2018 survey of data science / machine learning professionals (source)

Learning Python (version 3.6 or higher) is a key focus of MCS 260.

Most of our discussion of general computer science concepts will be based on the way they are seen and used in Python.

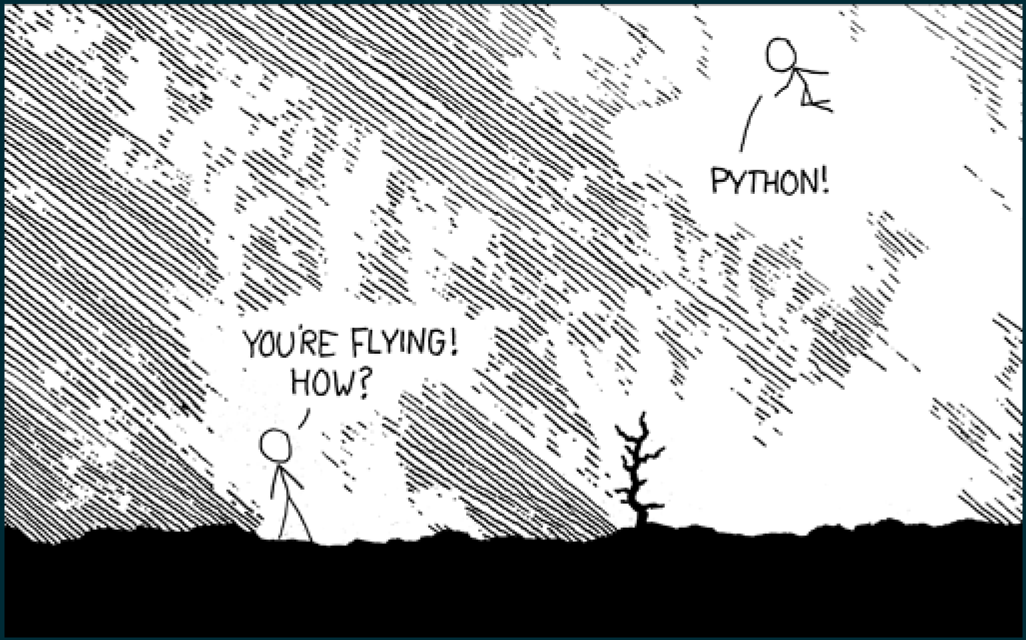
PYTHON VERSIONS

In this course we only use Python 3.

The transition from Python 2 to Python 3 was a major milestone, with incompatible changes.

Python 2 support ended in January 1, 2020.

LIVE DEMO TIME



Excerpt of xkcd by Randall Munroe CC-BY-NC-2.5

NUMBER SYSTEMS

Humans usually use the **decimal** number system, also known as **base** 10.

In this system there is a $10^0 = 1$s place, a $10^1 = 10$s place, a $10^2 = 100$s place, etc.

There are 10 digits with values $0, 1, \ldots, 9$.

In decimal, 312 means:

$$\color{red}{3}\color{green}{1}\color{orange}{2} = \color{red}{3} \times 10^2 + \color{green}{1} \times 10^1 + \color{orange}{2} \times 10^0$$

For any whole number $b > 1$ there is a number system called **base** b where the place values are b^0, b^1, b^2, etc.

In base b there are b digits with values $0, 1, \ldots, b-1$.

In mathematics, it is common to use a subscript to indicate the base.

So 201_5 means the base 5 number with digits $2, 0, 1$.

201_5 is equal to the decimal number 51:

$$\begin{aligned} 201_5 &= 2 \times 5^2 + 0 \times 5^1 + 1 \times 5^0 \\ &= 2 \times 25 + 1 \times 1 = \boxed{51} \end{aligned}$$

In computer science, three non-decimal number systems are often encountered.

- **Binary**, or base 2.
- **Hexadecimal**, or base 16.
- **Octal**, or base 8. (Least common.)

BINARY

The digits are 0 and 1. A binary digit is called a **bit**.

The place values are $1, 2, 4, 8, 16$, etc.

Example: 1001_2 means

$$1 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1 = 9$$

In Python, binary numbers are indicated by preceding the digits with `0b`.

So the previous example would be written `0b1001`.

We can convert to binary using integer division and remainder.

Integer division
$x//2$ means x divided by 2, discarding the remainer.
e.g. $7//2 = 3,\ \ 6//2 = 3$.

Remainder
$x\%2$ means the remainder when x is divded by 2.
$7\%2 = 1, 6\%2 = 0$.

To convert a number to binary, just keep track of the remainders when you repeatedly integer-divide by 2.

x	$x//2$	$x\%2$
312	156	0
156	78	0
78	39	0
39	19	1
19	9	1
9	4	1
4	2	0
2	1	0
1	0	1

So $312 = \texttt{0b100111000}$, i.e.
$312 = 256 + 32 + 16 + 8.$

Binary is not ideal for human consumption because of its low information density.

$$\text{e.g. } 9754 = \texttt{0b10011000011010}.$$

Hexadecimal addresses this, giving a more condensed way of expressing a sequence of bits.

HEXADECIMAL

Hexadecimal or **hex** is a condensed representation of binary, with one symbol for each 4-bit block.

Each 4-bit block is just a number between $\mathtt{0b0000} = 0$ and $\mathtt{0b1111} = 15$. We use **hex digits** $0 \ldots 9, A \ldots F$:

Digit	0	1	2	3	4	5	6	7
Value	0	1	2	3	4	5	6	7
Bit block	0000	0001	0010	0011	0100	0101	0110	0111

Digit	8	9	A	B	C	D	E	F
Value	8	9	10	11	12	13	14	15
Bit block	1000	1001	1010	1011	1100	1101	1110	1111

HEXADECIMAL

Hexadecimal or **hex** is a condensed representation of binary, with one symbol for each 4-bit block.

Each 4-bit block is just a number between $\texttt{0b0000} = 0$ and $\texttt{0b1111} = 15$. We use **hex digits** $0 \ldots 9, A \ldots F$:

Digit	0	1	2	3	4	5	6	7
Value	0	1	2	3	4	5	6	7
Bit block	0000	0001	0010	0011	0100	0101	0110	0111

Digit	8	9	A	B	C	D	E	F
Value	8	9	10	11	12	13	14	15
Bit block	1000	1001	1010	1011	1100	1101	1110	1111

In Python notation, **hexadecimal numbers begin with `0x`**, followed by the digits.

So `0x3e` means

$$\begin{array}{cc} 3 & \text{e} \\ \hline 0011 & 1110 \end{array} \longrightarrow \quad \texttt{0b00111110} = 62$$

Hexadecimal is also base 16. Another way to see `0x3e`:

$$\begin{aligned} \texttt{0x3e} &= \texttt{3} \times 16^1 + \texttt{e} \times 16^0 \\ &= 3 \times 16 + 14 \times 1 = 62 \end{aligned}$$

Aside: In decimal we sometimes separate groups of digits with punctuation for easier reading.

e.g. in the USA one million is often written "$1,000,000$".

In Python notation the underscore "_" can be used as a separator.

$$\begin{aligned} \texttt{0b1111\_0100\_0010\_0100\_0000} &= \texttt{0xf4240} \\ &= \texttt{1\_000\_000} \end{aligned}$$

When converting binary to hex, the number of bits may not be a multiple of 4 at first. In this case we need to add some zeros on the left:

$$\begin{aligned} \texttt{0b10101} &= \texttt{0b00010101} \\ &= \texttt{0b00010101} \\ &= \texttt{0x15} \end{aligned}$$

(As in decimal, adding zeros on the left doesn't change the value.)

To convert a decimal number to hex, one way is to convert to binary and group bits.

An alternative is to repeatedly integer-divide by 16 and use the remainders:

x	$x//16$	$x\%16$
62	3	14
3	0	3

Therefore $62 = \texttt{0x3e}$

OCTAL

Octal or **base** 8 is similar but we divide a binary number into blocks of 3 bits, to using $0, \ldots, 7$ to represent blocks of 3 bits.

In Python notation, **octal numbers begin with** $\texttt{0o}$ followed by the digits.

(That's numeral zero followed by lower case letter o.)

Example: $\texttt{0o775} = \texttt{0b111\_111\_101} = 509$

Octal is most commonly seen when setting file permissions on unix/Linux, where 9 bits are naturally divided into 3 groups of 3.

e.g.

```
chmod 600 secrets.dat
```

REFERENCES

- The first steps in working with Python are covered in Section 1.2 of *Downey*.
- Binary and hexadecimal are covered in Section 1.1 of *Brookshear & Brylow*.

ACKNOWLEDGEMENTS

- Some of today's lecture was based on teaching materials developed for MCS 260 by Jan Verschelde.

REVISION HISTORY

- 2021-08-24 Fix colors on slide about converting 63 to hex.
- 2021-08-23 Update to reflect TA schedule change
- 2021-08-22 Initial publication