Intro to Computer Science & Computer Programming

CS195 - Lecture 1
Instructor: Dr. V

Rules for Class Interaction

1. 🕒 Please be on time.
2. 🗣️ Jump in with comments/questions.
3. 📥 Pay attention to your email.
4. 📧 Email me if you have any questions.
5. 💻 Bring your laptop to every class, unless explicitly told otherwise.
6. 😷 PLEASE wear a mask if you have
   - sore throat or cough
   - sneezing or runny nose

   (ask me for a mask if you do not have one)

Personal laptops

- You must have a personal laptop for CS courses
  - though there may be a way to use a phone/tablet instead
- If you cannot afford a laptop, you can borrow one from the library
- If you need laptop recommendations (specific to your budget), just ask
- If you have a laptop, but it's old/slow, I may be able to help you set it up for max functionality

About the Instructor

Dr. Veksler
WER218; vveksler@caldwell.edu

Engineer, Research Scientist, Professor

- worked in industry, government, academia, self-employed
- web development, AI and robotics, HCI, cyber-security, data analytics, research on human behavior

I'd love to get to know each of you, but...

- I am bad at remembering names, so please just keep reminding me what your name is
- If I mess up your name, just correct me as many times as it takes
- Keep your name-plate in front of you

CS195

Blackboard

- https://caldwell.blackboard.com/

Grading (5-15-40-20-20)

- attendance/participation
- quizzes
- assignments
- midterm
- final

Code Evaluation

- Submitted code will be evaluated for
  - Correctness
    - Does it work?
    - Does it do what is intended?
    - Are there bugs?
  - Readability
    - Make sure your code indented properly
    - Name variables/functions/classes in a meaningful way
    - Add comments and signature descriptions
    - Keep your code clean

Late submissions

- late assignments will lose 10pts (out of 100) for each week they are late, but will not go lower than 70 (out of 100)
  - 1-7 days late: highest grade = 90
  - 8-14 days late: highest grade = 80
  - >14 days late: highest grade = 70

Cheating

- don't do it. not worth.
- what is considered cheating in this class?
  - it's complicated (if you're not sure, just ask)
    - you're encouraged to look up answers online, but…
    - on quizzes/exams
      - don't look at classmates' answers
      - may have other requirement
    - on assignments
      - don't submit someone else's code
      - if a part of your code was copied, give credit to the source
      - i <u>will</u> have your code checked for plagiarism

Class Structure

- 14 weeks of classes
- each week:
  - lecture
  - quiz based on lecture
  - practicum
    - technical tips, interactive time to work on your assignments, ask questions, and get feedback

Inclement weather or other class cancellations

- If the school is open, I will be here
- Use your best judgment
- Check school site
- Sign up for notifications
- Check your email

Questions? Comments? Concerns?

Computer Programming (aka coding)

- Who is it for?
  - Anyone can benefit from basic coding skills
- Career focused on coding (e.g., software engineering)
  - If you enjoy solving puzzles, you will love programming

- A lot of CS focuses on coding, but
  - coding is not just for CS or related disciplines
  - coding is not the entirety of CS

What is your coding experience level?

0. Zero coding experience
1. Some drag-and-drop coding, like Scratch
2. Been exposed to text-based coding; i can guess what this does:

```
s='lo wor'
print('hel'+s+'ld')
```

3. Some text-based coding; i can guess what this does:

```
for i in range(10):
    print(i)
    if i==5: break
```

4. A lot of coding; i can guess what this does:

```
for x,y in enumerate(map(lambda x:x*2,range(5))):
    print(f'{x} => {y}')
```

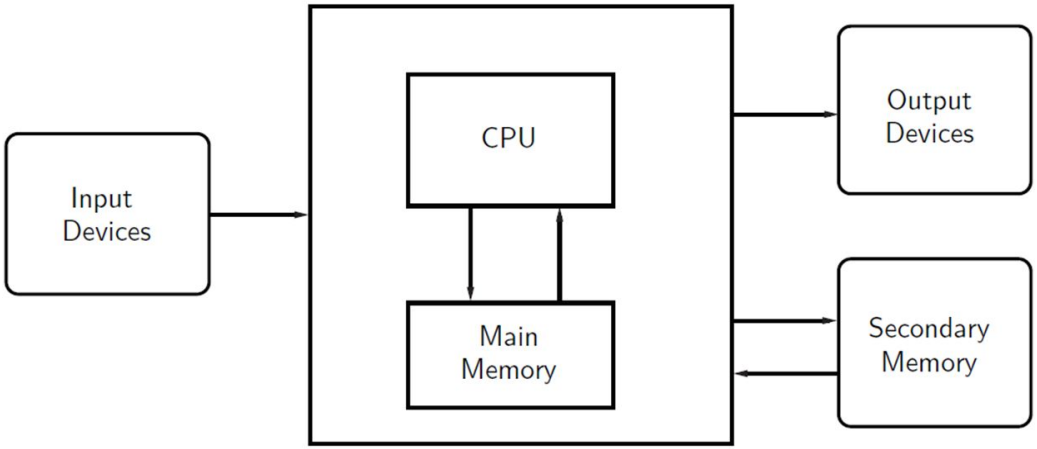
Your prior coding experience

- If you have zero or little experience
  - don't be intimidated – you can do this, it just takes time
  - you will NOT be left behind
  - **ask lots of questions!**
  - you will need to put in a lot more time into your assignments
- You have some coding experience
  - **ask lots of questions!**
  - challenge yourself
  - help your classmates
- You have lots of coding experience
  - I will give you alternate assignments
  - you still have to take the exams, but you may be excused from the quizzes on a case-by-case basis
  - help your classmates

Let's start at the beginning...

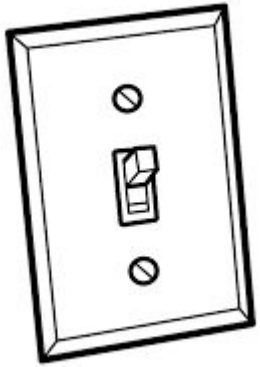
Computer hardware

- main memory (RAM; Random Access Memory)
  - fast
  - lost when machine is turned off
  - expensive (so you probably only have 4GB-32GB)
- secondary memory
  - permanent storage
  - usually SSD or HD



How does your computer store information?

- the smallest unit of information in your computer's memory is called a **bit**
- it is a switch
  - it can be switched to ON position, or
  - it can be switched to OFF position
- one bit can represent any binary information
  - ON or OFF; True or False; 1 or 0
- all info on your machine is just 1's and 0's



...00101010010101010100101111010110111...

Other numbers

- if a computer can only store 1's and 0's, how does it represent a 2? or 3? or any other number?
  - well, arabic numerals only offer digits 0..9, and yet we can represent number >=10 just fine
    - all we do is add more digits to represent 10s, 100s, 1000s, 10000s…
      - i.e., 10^1, 10^2, 10^3, 10^4…
    - so we do the same in binary – add more digits to represent 2s, 4s, 8s, 16s…
      - i.e., 2^1, 2^2, 2^3, 2^4…)

Decimal vs Binary

0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	?

120	1111000
121	1111001
122	1111010
123	1111011
124	1111100
125	1111101
126	1111110
127	1111111
128	?

There are 10 kinds of people in the world: those who understand binary code, and those who don't.

Bits and Bytes

- Most often we measure memory size in bytes, not bits
- **Each byte is 8 bits**
- Most modern computers are byte-addressable
- Each address identifies a single byte of storage

```
1B (1 byte) = 8b (8 bits)
1KB (1 kilobyte) = 1000B (1000 bytes)
1MB (1 megabyte) = 1000KB (1000000 bytes)

1KiB (1 kibibyte) = 1024B
1MiB (1 mebibyte) = 1024KiB
```

Bits and Bytes

- what's the largest positive integer you can represent in a single byte of memory?

- what's the largest positive integer you can represent using two bytes of memory?

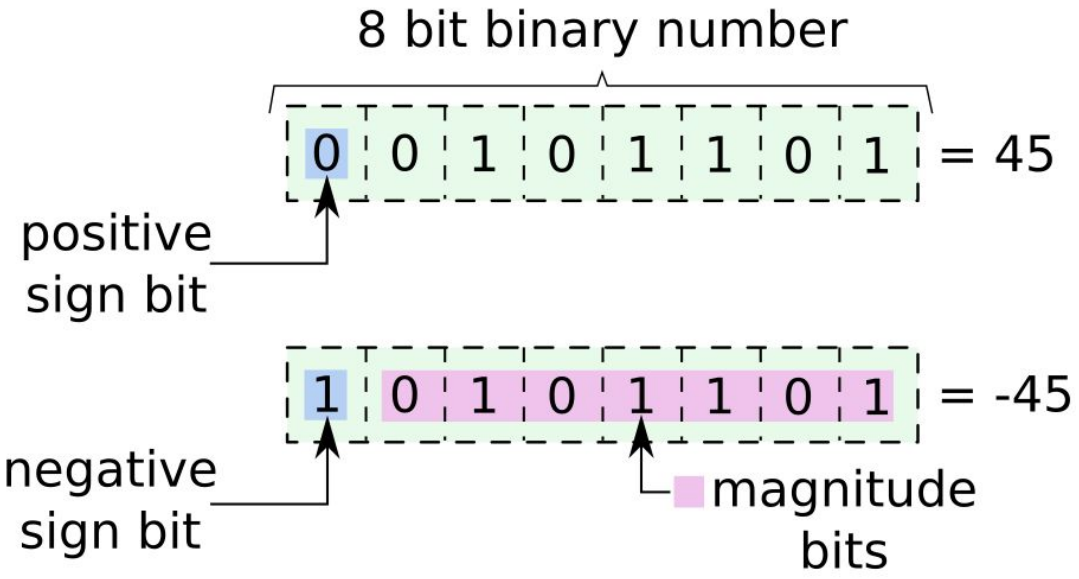
What about negative numbers?

- You can use 1 byte to
  - store any positive integer 0-255

  - What if you wanted to represent integers that can be either positive or negative?

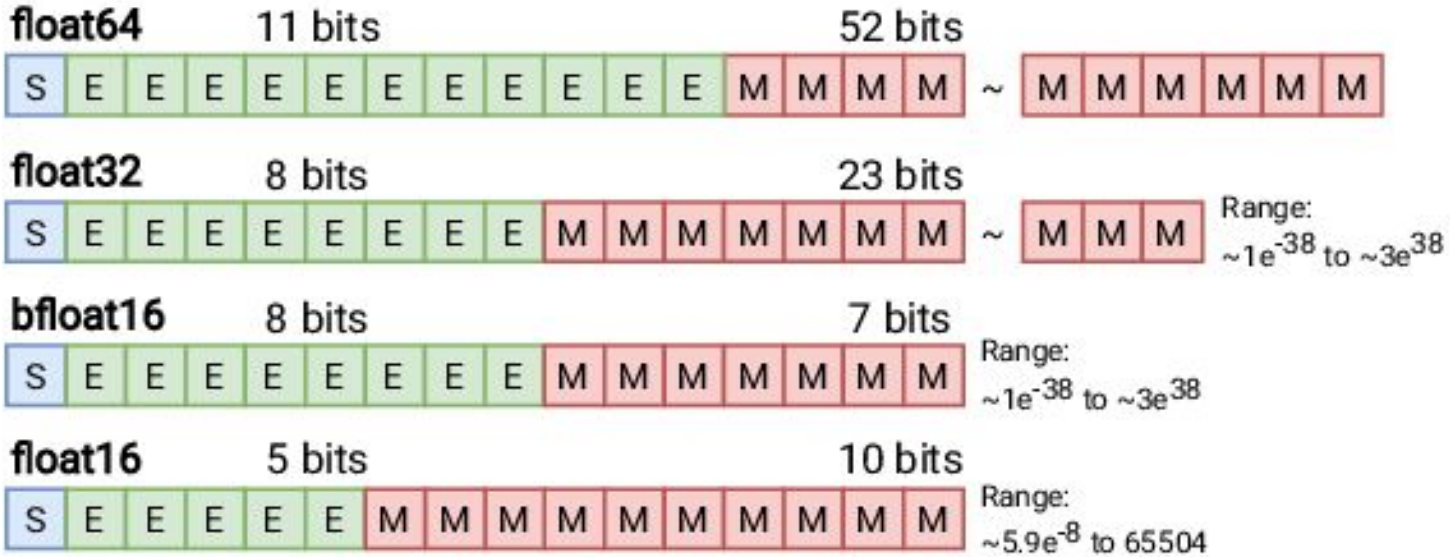
Signed vs Unsigned Integers

- You can use 1 bit to signify whether the number is negative, and other 7 bits for 128 possible patterns



What about floating point numbers?

- `S - Sign`
- `E - Exponent`
- `M - Mantissa`



What about text?

- How would you represent letters using binary?

- You want to save the text "Hello" on computer
  - your only storage option is on/off switches (bits)
  - how would you store this text?

ASCII (American Standard Code for Information Interchange)

Letter	ASCII Code	Binary	Letter	ASCII Code	Binary
a	097	01100001	A	065	01000001
b	098	01100010	B	066	01000010
c	099	01100011	C	067	01000011
d	100	01100100	D	068	01000100
e	101	01100101	E	069	01000101
f	102	01100110	F	070	01000110
g	103	01100111	G	071	01000111
h	104	01101000	H	072	01001000
i	105	01101001	I	073	01001001
j	106	01101010	J	074	01001010
k	107	01101011	K	075	01001011
l	108	01101100	L	076	01001100

How does your computer store text?

- ASCII
  - Each text character is represented by a number between 0 and 127
  - Each character requires 7 bits
- But we need way more than 128 unique characters for
  - non-English characters βγξΨ̃ḦĊϕ□
  - symbols ®Ⅲ⅝∑▽∀≡⊨⊞⋮↘↩
  - emojis 🙄💚💙🕐🔺👱🧵🎄🎲🔐
- Unicode format (UTF-8, UTF-16, UTF-32)
  - allows for more bits per character

How would your machine represent an image?

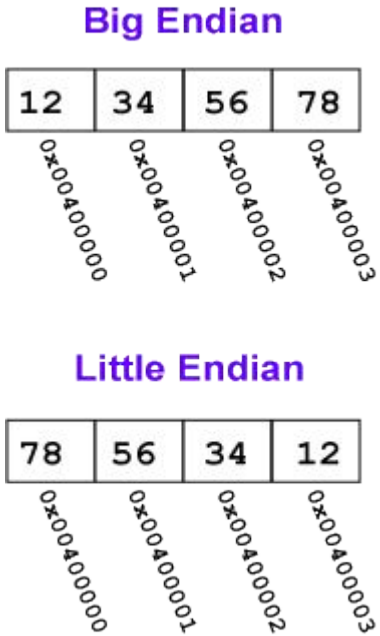
- each pixel in an image is represented by a mix of reds, greens, and blues
  - an 100x100 image where there may be 256 shades of red, 256 shades of green, and 256 shades of blue
    - 3 bytes per pixel
      - 1 byte for red, 1 byte for green, 1 byte for blue
    - a total of 100x100x3 bytes needed to store image

How does your computer know?

- How does your computer know whether the next 3 bytes represent a number, a unicode character, 3 ASCII characters, a pixel, sound frequencies, executable code, or something else entirely?

Big Endian vs Little Endian

- there's no reason for data to flow left-to-right on a computer
- Big Endian systems read data left-to-right
- Intel processors (among others) are Little Endian
  - Little Endian systems are weird (to humans)
  - the least significant byte is written first

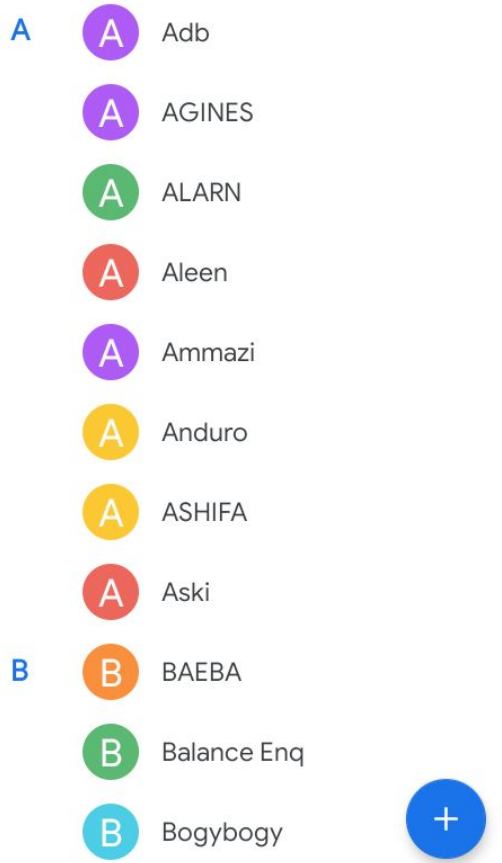


Algorithms & Pseudocode

Let's consider a sample search

- Task:
  - Find a person in a contact list to call them

1. open contact list
2. magic happens
3. call the person



Searching a contact list...

```
1.  go to first contact
2.  if the name matches:
3.     call the person; quit
4.  else:
5.     go to next contact
6.     repeat step 2
```

Searching a contact list...

```
1.  go to first contact
2.  if the name matches:
3.     call the person; quit
4.  else, if we are not at the end of the contact list:
5.     go to next contact
6.     repeat step 2
7.  else:
8.     say "Contact not found"; quit
```

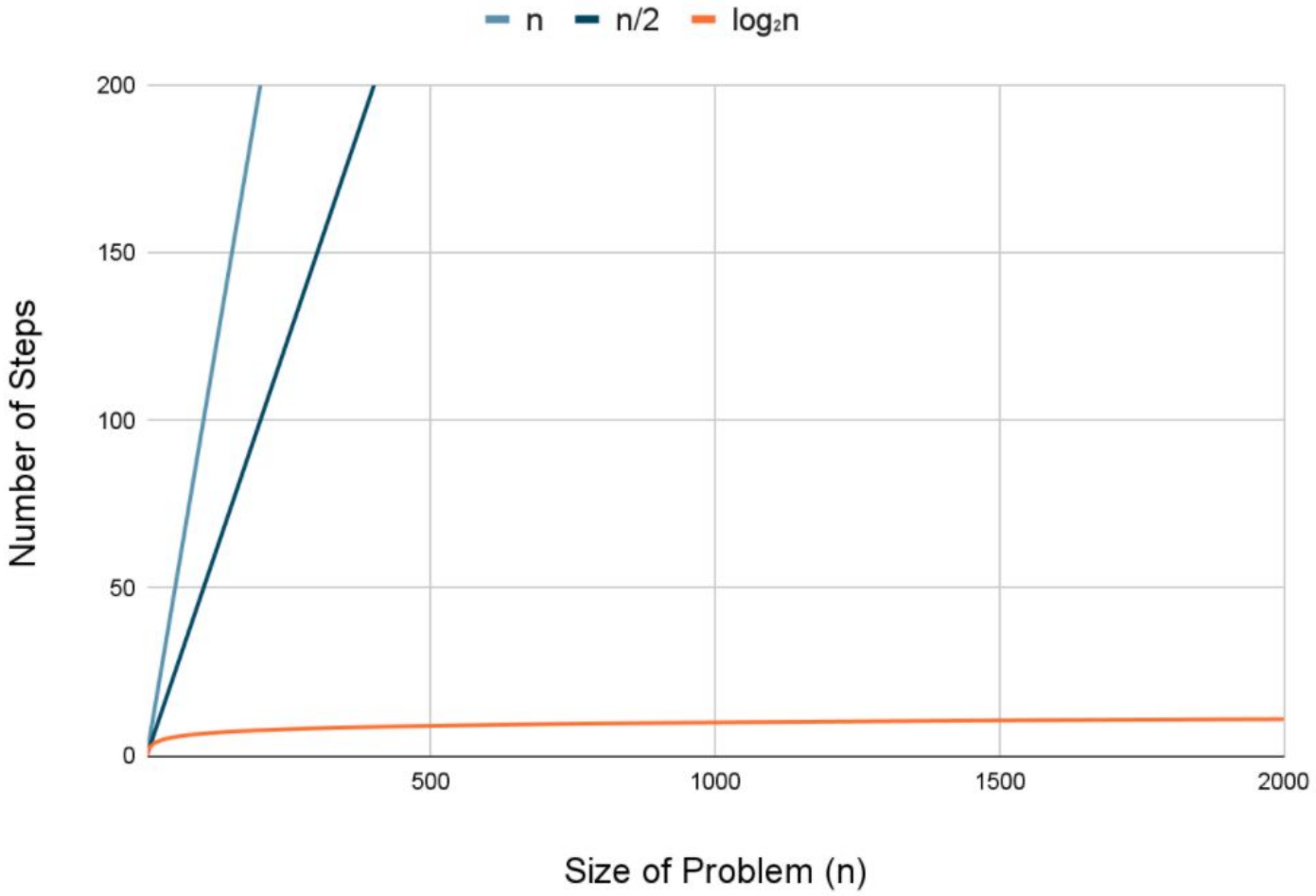
Can we make this algorithm faster?

Searching a contact list...

1. go to middle of list
2. if the name matches:
3. call the person; quit
4. else, if the name comes after current contact:
5. throw out current contact and all those before it
6. repeat step 1
7. else, if the name comes before current contact:
8. throw out current contact and all those after it
9. repeat step 1

Searching a contact list...

1. go to middle of list
2. if the name matches:
3. call the person; quit
4. else, if name is after current contact & this isn't last contact:
5. throw out current contact and all those before it
6. repeat step 1
7. else, if name is before current contact & this isn't 1st contact:
8. throw out current contact and all those after it
9. repeat step 1
10. else:
11. say "Contact not found"
12. quit


n
n/2
log₂n
200
150
100
50
0
Number of Steps
500
1000
1500
2000
Size of Problem (n)

Programming Languages

There are many many programming languages

- general purpose vs domain-specific
- programming style
  - imperative, declarative, functional, object-oriented, hybrid
- compiled, interpreted
- ease of use
  - higher-level programming languages
    - untyped (e.g., python, javascript)
    - typed (e.g., C#, Kotlin, Swift, Objective-C, Go)
  - lower-level programming languages (e.g., assembly)

Python

- general purpose
- interpreted
- untyped
- hybrid programming style
- optimized for being
  - **easy to read**
  - **easy to write**
  - **fast development**
- **NOT** optimized for
  - runtime speed

Python

- Very popular
  - quick scripts
    - arithmetic
    - file and string manipulation
  - web apps, web services
  - statistics and data analytics
  - AI/ML and scientific computing
- Probably not the best for
  - native GUI application development
  - game development

Questions? Comments? Concerns?

Assignment 1

- install python 3
- install Visual Studio Code (VSCode)
- create folder "Assignment 1"
- create file hello.py
- in hello.py
  - add comments (your name, course number)
  - add code to print "hello world"
  - run hello.py
- submit hello.py file on blackboard

Introduction to Python

Let's try out python

- REPL
  - open terminal
  - type (and then hit enter)
    - python
    - print('hello world')
    - print(3 + 5)

Try out python

- .py file
  - create a folder "hello"
  - open folder in VSCode
  - create file "hello.py"
  - type in that file:
    - print('hello world')
  - save file
  - open terminal
  - in terminal type (and then hit enter)
    - python hello.py

Python syntax

- comments
  - a comment is text that Python will ignore
  - it is there for documentation purposes
- a comment in python begin with #

```
# this is a python comment
```

- in hello.py
  - add comment indicating your name
  - add comment indicating course number

Python syntax

- white-space usage
  - carriage returns at the end of each line signify end of operation
  - do not add tabs or spaces at start of lines unnecessarily – these are special in python (more on this later)

Python syntax

- function calls
  - a function call is instruction to do some action
  - function name is the verb
    - what action to do
  - function arguments (enclosed in parentheses immediately following the function name) are the nouns/adjectives
    - how & with what to do the action
  - Example: **print**("**hello**")
    - verb: **print**
    - noun: "**hello**" string
    - what does it do?
      - outputs the text **hello** to display

Python syntax

- What happens when your syntax is incorrect?
- Try it out:
  - print("hello") ← this is the correct way
  - print "hello"
  - print("hello")
  - Print("hello")
  - "print"("hello")

Python syntax

- What happens when your syntax is incorrect?
- Try it out:
  - print "hello"
    - Syntax Error
      - python function arguments should be in parentheses
  - print("hello")
    - Indentation Error
      - python indentation is special (more on this later)
  - Print("hello")
    - Name Error
      - python doesn't know what Print means (it's case-sensitive – print is not the same as Print)
  - "print"("hello")
    - Type Error
      - **"print"** is a string, **print** is a function

Visual Studio Code

- file management
- extensions
- autocomplete
- running code
- terminal
- output