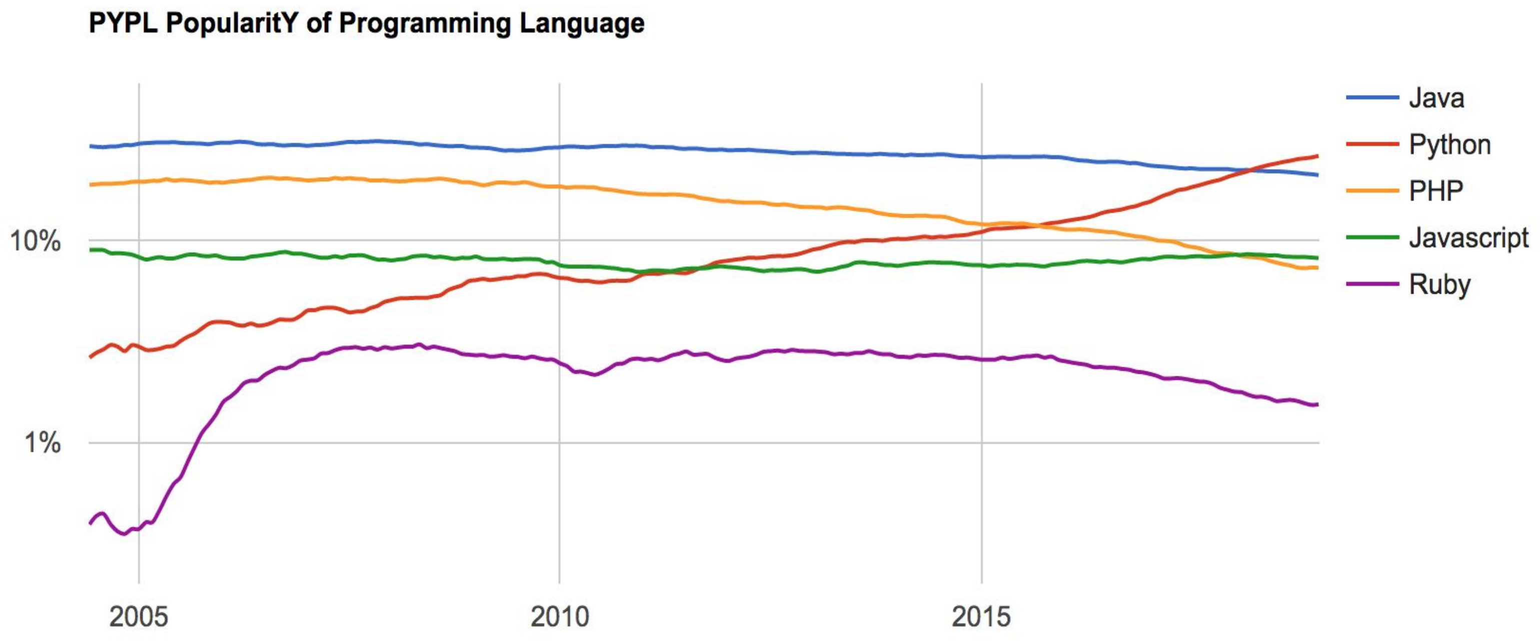
Learning Programming Languages--(Faster)!

Nnenna Ndukwe, Software Engineer
@OReillyMedia

Why Is It Important to Be a Quick Learner?

- The market is always evolving
- Your adaptability can give you a very specific type of edge
  - Of course, risk is always involved when comparing specialization vs. generalist
  - Builds trust with companies you work for and teams you work on
- You will figure out what you like and ***don't*** like sooner rather than later!
  - THINK: career projection, becoming more opinionated about your craft, specialization interests, etc.

Worldwide, Python is the most popular language, Python grew the most in the last 5 years (16.2%) and PHP lost the most (-5.6%)



Source: http://pypl.github.io/PYPL.html

Review the Fundamentals of Programming

- ➔ It’s less about the language and ***more*** about fundamental programming concepts!
- ➔ Understanding the fundamentals of Computer Science (or programming in general) will help you contextualize a language’s conventions when first diving in to learn

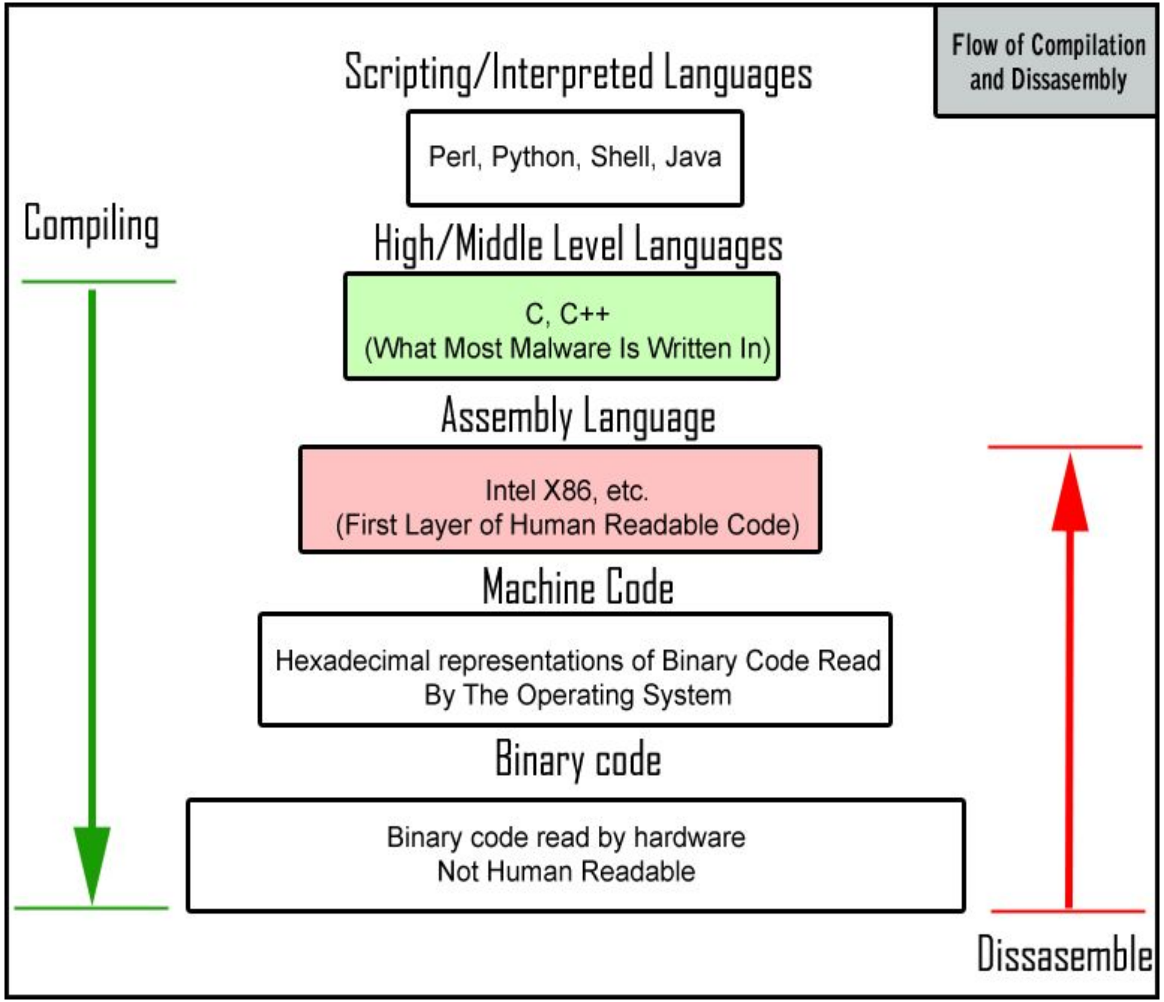
Understanding the Scope: Classify The Language(s)

When first learning a new programming language, begin to compartmentalize the language based upon its characteristics!

Great Source for Language Type Overview: http://cs.lmu.edu/~ray/notes/pltypes/

Why?

- Help you to understand what it can be used for
- Helps you to understand ***how*** it is used (implementation examples)
- Helps interpret your compilation and runtime errors if you use it to build something
- The better you can define the language, the better you'll understand it!


Flow of Compilation and Dissasembly
Scripting/Interpreted Languages
Perl, Python, Shell, Java
Compiling
High/Middle Level Languages
C, C++
(What Most Malware Is Written In)
Assembly Language
Intel X86, etc.
(First Layer of Human Readable Code)
Machine Code
Hexadecimal representations of Binary Code Read By The Operating System
Binary code
Binary code read by hardware
Not Human Readable
Dissasemble

Levels

- Mentally, ***GROUP*** the languages by LEVELS
- Have examples of other languages that are in the same level
- Learn small differences between the different language levels
- Use visuals to help you remember

Levels of Programming Languages

High-level program

```
class Triangle {
  ...
  float surface()
    return b*h/2;
  }
```

Low-level program

```
LOAD r1,b
LOAD r2,h
MUL r1,r2
DIV r1,#2
RET
```

Executable Machine code

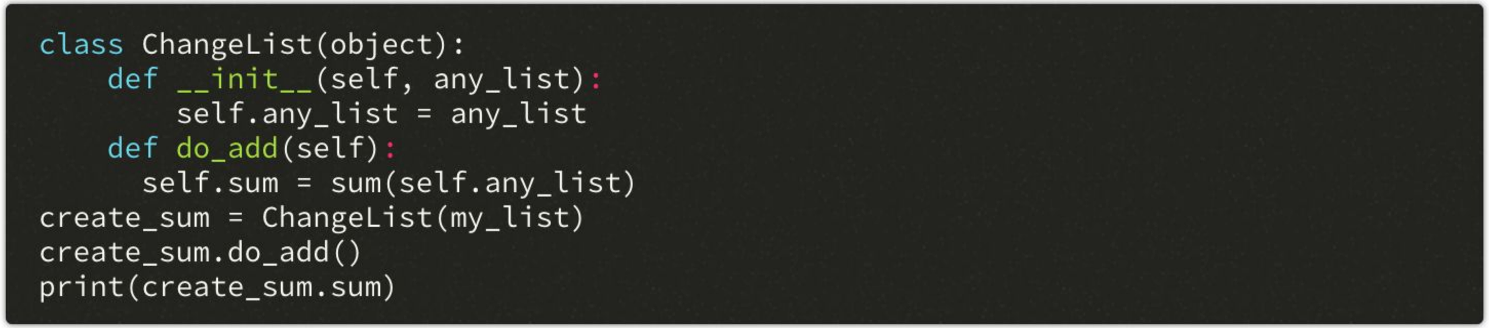
```
0001001001000101
0010010011101100
10101101001...
```

Understanding the Scope: Classify the Language(s)

Types of Programming Styles

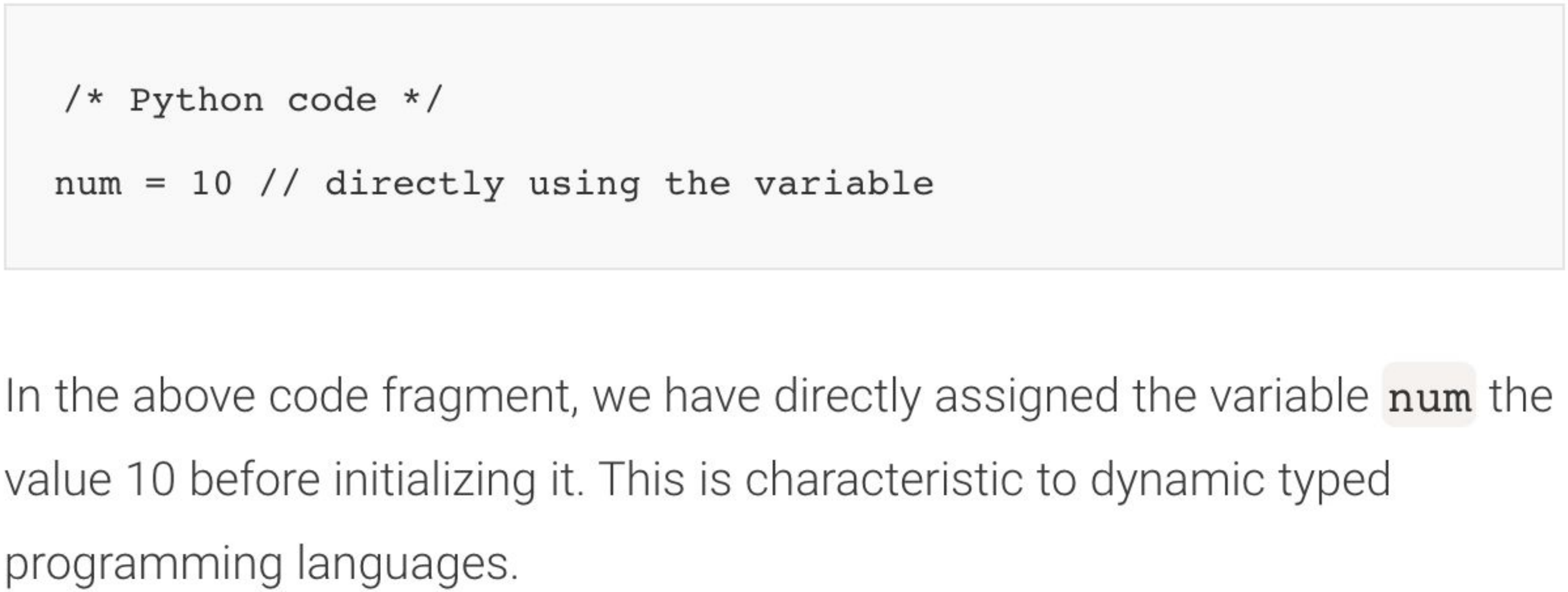
- Object-Oriented Programming
- Functional Programming
- Procedural Programming
- Imperative Programming

etc.



```
class ChangeList(object):
    def __init__(self, any_list):
        self.any_list = any_list
    def do_add(self):
      self.sum = sum(self.any_list)
create_sum = ChangeList(my_list)
create_sum.do_add()
print(create_sum.sum)
```

Type Checking: Is it Dynamically Typed?



```
 /* Python code */
num = 10 // directly using the variable
```

In the above code fragment, we have directly assigned the variable `num` the value 10 before initializing it. This is characteristic to dynamic typed programming languages.

Type Checking: Is it Statically Typed?

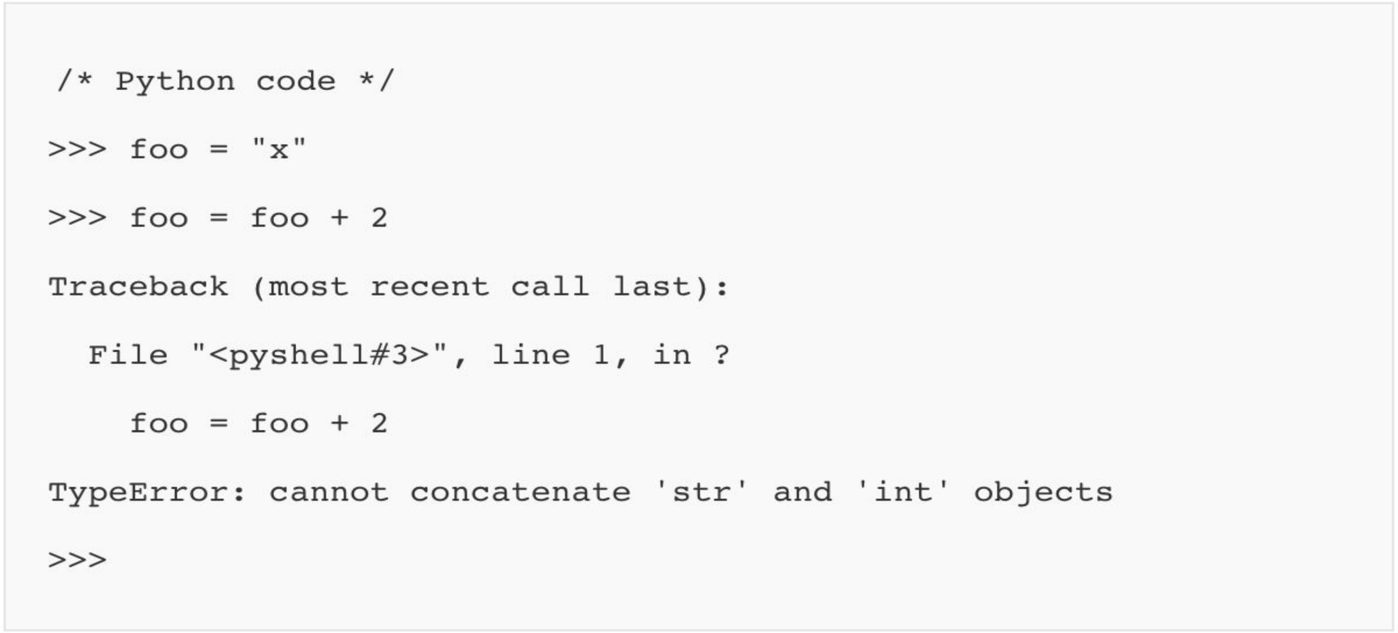
Static typing does not imply that you have to declare all the variables first, before you use them; variables maybe be initialized anywhere, but developers have to do so before they use those variables anywhere. Consider the following example:

```
/* C code */
static int num, sum; // explicit declaration
num = 5; // now use the variables
sum = 10;
sum = sum + num;
```

Type Checking: Is it Weakly Typed?

```
 /* PHP code */

<?php

$foo = "x";

$foo = $foo + 2; // not an error

echo $foo;

?>
```

Type Checking: Is it Strongly Typed?



```
/* Python code */
>>> foo = "x"
>>> foo = foo + 2
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in ?
    foo = foo + 2
TypeError: cannot concatenate 'str' and 'int' objects
>>>
```

Frustrations with Learning to Code

The three most commonly reported age-related frustrations were:

1. 14% were frustrated by perceived cognitive impairments, such as memory loss and difficulty in concentrating.
2. 11% were frustrated by lack of free time since they often had other duties, such as being a spousal caretaker.
3. 10% were frustrated by lack of human contact with tutors or peers, since they must learn online and do not have convenient access to in-person classroom environments.

Source: Communications of the ACM Magazine, August 2017

https://dl-acm-org.ezproxy.bu.edu/citation.cfm?doid=3127343.3105421

Solutions to Some Learning Frustrations

Diversify your learning habits...

- Attend **events** around the **language** you want to learn
  - This keeps you conversing about the language and you will learn from others about their opinions and practices
  - You'll hear talks and sit in on workshops that could keep you up to date on the latest tools built on top of the language and/or framework
  - You'll improve your ability to communicate more complex technical problems

Solutions to Some Learning Frustrations

Listen to **podcasts** around the language(s) you want to learn

- Easy to listen to actively & passively when you're on the go

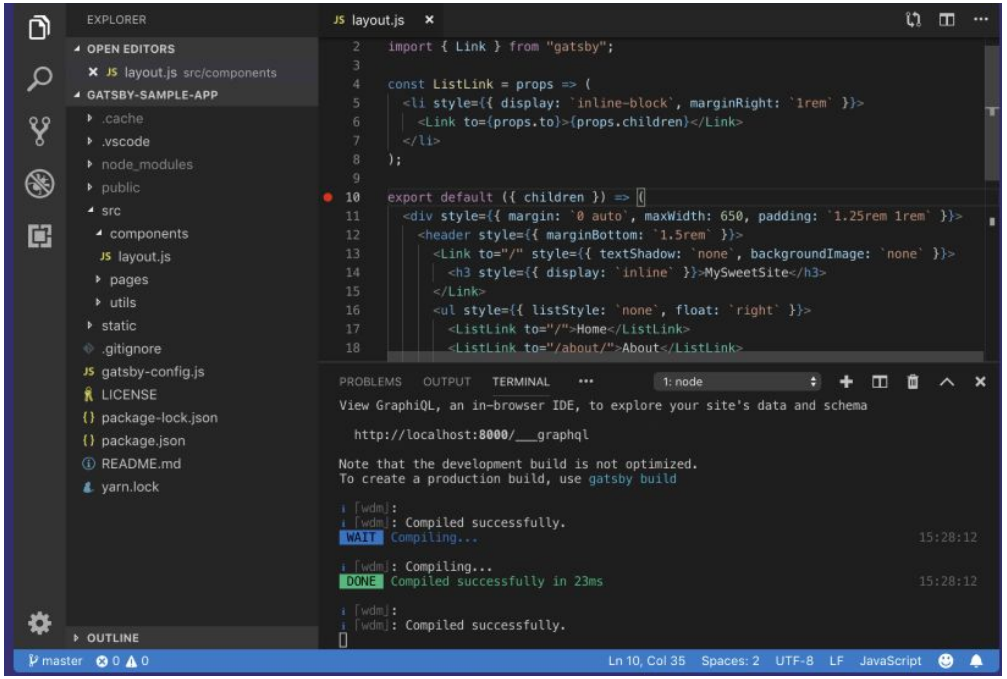
SOFTWARE ENGINEERING DAILY

SE

Tips & Tricks

Practice Searching & Reading Code in Your Text Editor!

- The faster you become at *searching* and *interpreting* brand new code, the faster you will be able to understand what's happening, explain it, develop and opinion on it (if necessary), and contribute to it!



```
import { Link } from "gatsby";

const ListLink = props => (
  <li style={{ display: `inline-block`, marginRight: `1rem` }}>
    <Link to={props.to}>{props.children}</Link>
  </li>
);

export default ({ children }) => (
  <div style={{ margin: `0 auto`, maxWidth: 650, padding: `1.25rem 1rem` }}>
    <header style={{ marginBottom: `1.5rem` }}>
      <Link to="/" style={{ textShadow: `none`, backgroundImage: `none` }}>
        <h3 style={{ display: `inline` }}>MySweetSite</h3>
      </Link>
      <ul style={{ listStyle: `none`, float: `right` }}>
        <ListLink to="/">Home</ListLink>
        <ListLink to="/about/">About</ListLink>
```

```
View GraphiQL, an in-browser IDE, to explore your site's data and schema

  http://localhost:8000/___graphql

Note that the development build is not optimized.
To create a production build, use gatsby build

i [wdm]:
i [wdm]: Compiled successfully.
WAIT  Compiling...                                                    15:28:12

i [wdm]: Compiling...
DONE  Compiled successfully in 23ms                                   15:28:12

i [wdm]:
i [wdm]: Compiled successfully.
```

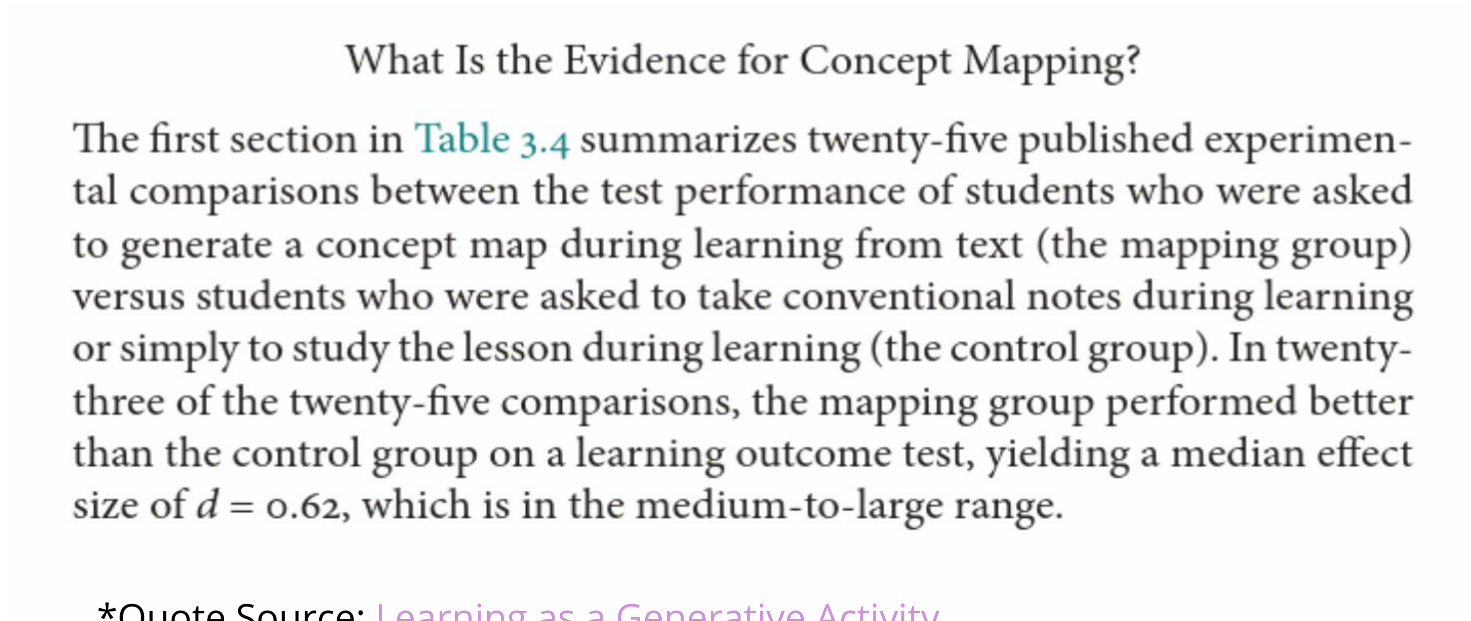
Learning Method: Summarizing

“In 26 out of 30 experimental learning comparisons, students who were asked to generate summaries during learning performed better than a control group that was not asked to generate summaries on a subsequent test of the material…“

“Students learn more deeply if they are able to effectively select main ideas, organize them, and then restate them in their own words.”

******TRY THIS:*** Stop at reasonable checkpoints when going through some tutorials or documentation for a language or framework and create a new file that immediately puts the concept you’re learning to practice. Comment well to explain to yourself the purpose of what you’ve just learned.

Learning Method: Mapping



What Is the Evidence for Concept Mapping?

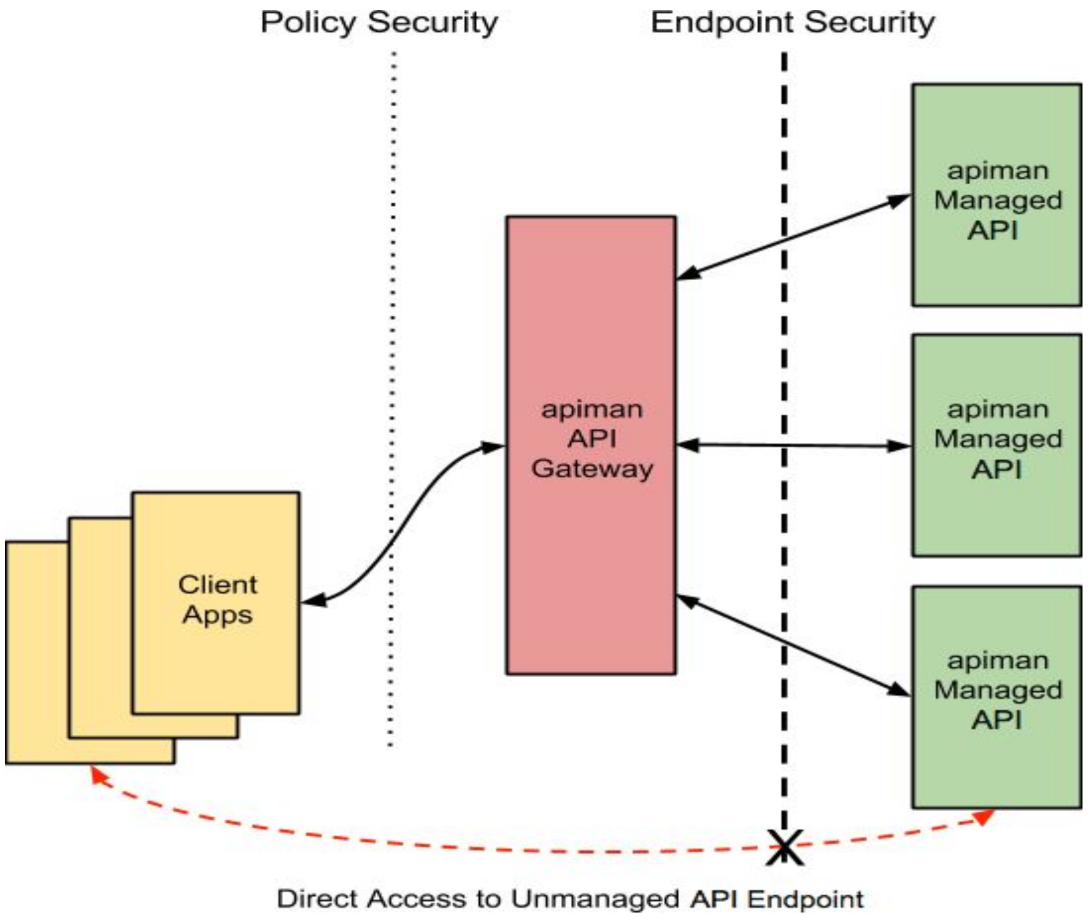
The first section in Table 3.4 summarizes twenty-five published experimental comparisons between the test performance of students who were asked to generate a concept map during learning from text (the mapping group) versus students who were asked to take conventional notes during learning or simply to study the lesson during learning (the control group). In twenty-three of the twenty-five comparisons, the mapping group performed better than the control group on a learning outcome test, yielding a median effect size of $d = 0.62$, which is in the medium-to-large range.

*Quote Source: Learning as a Generative Activity

Awesome example of using visuals to convey information:
https://boyan.io/how-webassembly-influences-existing-javascript-frameworks/

An example of mapping out an API endpoint diagram

***TRY THIS*:** Stop at reasonable checkpoints when going through some tutorials or documentation for a language or framework and try drawing/mapping out a diagram of the concept or process that was explained. Use arrows, shapes, and labels that symbolize major parts of the concept or process to convey it as best you can. This stimulates your cognitive skills and helps you retain & summarize the information better!



Helpful Quick Learning Sources

1. Strong/Weak & Dynamic/Static Type Programming Explanation
   https://www.sitepoint.com/typing-versus-dynamic-typing/
2. Good Programming Conventions to Remember:
   http://courses.cs.vt.edu/cs2604/fall02/standards.html
3. Diagram of Types of Programming:
   https://www.info.ucl.ac.be/~pvr/paradigmsDIAGRAMeng108.pdf
4. Programming Styles (Examples with Python):
   https://blog.newrelic.com/engineering/python-programming-styles/
5. Comparing and Describing the Major Programming Paradigms
   http://www.eecs.ucf.edu/~leavens/ComS541Fall97/hw-pages/paradigms/major.html

Platforms to Create Web Development Projects On

Create projects in their online virtual environments so you won't have to worry about installing things onto your own machine. These platforms handle all of that. Save and edit your projects, tell others about your projects, and more!

1. Codepen.io
2. Repl.it
3. CodeSandbox.io

Helpful Books to Read or Listen To

1. Pragmatic Thinking & Learning
2. The Self-Taught Programmer
   a. This book was how I was able to hit the ground running with learning how to code, which ultimately helped me secure my first Software Engineering Internship!
3. The Pragmatic Programmer
4. The Complete Software Developer's Career Guide
5. Learning as a Generative Activity : Eight Learning Strategies that Promote Understanding

Coding Podcasts to Listen to ...

Search for these podcast series to download on your phone and listen to no matter where you are or are headed to.

1. CodeNewbie
2. Software Engineering Daily
3. SE Radio

Continue Learning....

Twitter: https://www.twitter.com/nnennahacks

LinkedIn: https://linkedin.com/in/nnenna-ndukwe/

Medium: https://medium.com/@nnennahacks

O’Reilly Media Learning Platform Free Trial Link

http://oreilly.com/getsafari?code=ITL19