**Week 2 Teacher's Guide: "Learning Your Instruments"**

**Programming Fundamentals - JavaScript & Python**

**Week:** February 24-28, 2025  
**Total Hours:** 30 hours (Mon-Thu: 6.5h/day, Fri: 4h)  
**Theme:** Building programming foundation in two languages simultaneously

**Week Overview & Teaching Philosophy**

**The "Two Instruments" Approach**

Week 2 is where students **finally start coding** - and they've earned it after mastering professional tools in Week 1. The unique approach of this bootcamp is teaching JavaScript and Python **side-by-side from day one**.

Use the musician analogy:

* "A professional session musician can play multiple instruments. They understand music theory, not just one specific guitar."
* "Similarly, professional developers understand programming concepts - not just one language's syntax."

**Key Teaching Principles for Week 2:**

1. **Concepts first, syntax second** - Focus on "what" and "why" before "how"
2. **Compare and contrast constantly** - "Here's how JS does it... now here's Python"
3. **Normalize confusion** - "Yes, it's a lot. That's normal. You're learning TWO languages."
4. **Emphasize transferable skills** - "Once you understand loops, the syntax is just details"
5. **Make it practical** - Real problems from day one, not toy examples

**Week 2 Success Metrics:**

* Students can write basic programs in both languages
* Students understand when to use each language
* Students can debug their own code systematically
* Students embrace the learning process (not perfection)
* Students help each other learn
* Professional habits from Week 1 continue

**Teaching Challenge:**

Students will feel overwhelmed. This is **two languages in one week**. Your job is to:

* Validate their feelings ("Yes, this is hard")
* Show the patterns ("See? Same concept, different syntax")
* Celebrate small wins ("You just wrote Python!")
* Build confidence ("You can do hard things")

**Daily Breakdown**

**DAY 1: Monday, February 24, 2025**

**"Setting Up Your Instruments" (6.5 hours)**

**Schedule:**

* 08:30-09:00: Daily Standup & Week 2 Kickoff
* 09:00-10:30: Programming Paradigms & REPL Exploration
* 10:30-10:45: Break
* 10:45-12:00: Node.js & Python Environment Setup
* 12:00-12:30: Lunch Break
* 12:30-14:00: First Programs in Both Languages
* 14:00-14:15: Break
* 14:15-15:15: REPL Experimentation & Interactive Coding
* 15:15-15:30: Daily Standup & Reflection

**Session 1: Daily Standup & Week 2 Kickoff (30 min)**

**Standup Questions:**

1. What did you work on over the weekend?
2. What's one Git thing you feel confident about?
3. Any blockers?

**Week 2 Introduction (15 min):**

"Last week was your professional foundation. This week, you become programmers.

But here's what makes this program different: most bootcamps teach you ONE language. You're learning TWO. Why?

Because companies don't hire 'JavaScript developers' or 'Python developers' - they hire DEVELOPERS who can solve problems. The language is just a tool.

By the end of this week, you'll:

* Write programs in JavaScript AND Python
* Understand what makes each language special
* Know which tool to reach for
* Think like a programmer, not just a coder

Yes, it's ambitious. Yes, you can do it. Let's go."

**Session 2: Programming Paradigms & REPL Exploration (1.5 hours)**

**Learning Objectives:**

* Understand what programming paradigms are
* Learn what JavaScript and Python excel at
* Explore REPL as learning tool
* Write first "Hello World" in both languages

**What is Programming? (20 min)**

Start with the big picture:

"Programming is giving instructions to a computer. But computers are very literal - they do EXACTLY what you tell them, not what you MEANT.

Think of it like GPS directions:

* 'Turn right' - Computer does it
* 'Turn kinda right' - Computer crashes
* 'Turn right at the nice house' - Computer doesn't know what 'nice' means

Programming languages are how we give these precise instructions."

**The Two Languages (20 min)**

**JavaScript:**

Born: 1995

Created by: Brendan Eich (in 10 days!)

Originally for: Making websites interactive

Now used for: Everything - frontend, backend, mobile apps

Known for: Flexibility, async operations, web dominance

Runs on: Browsers (Chrome, Firefox) and Node.js (server)

Philosophy: "There's many ways to do it"

**Python:**

Born: 1991

Created by: Guido van Rossum

Originally for: Scripting and automation

Now used for: Data science, AI, backend, automation, everything

Known for: Readable code, great for beginners, powerful libraries

Runs on: Python interpreter on any computer

Philosophy: "There should be one obvious way to do it"

**Key Differences:**

JavaScript Python

-------------------------------------------

Uses {} Uses indentation

Semicolons optional No semicolons

let/const/var Just assignment

CamelCase common snake\_case common

Web-native General-purpose

Async by design Sync by default

**Show Real Code Examples (20 min):**

**JavaScript:**

// JavaScript

function greet(name) {

return `Hello, ${name}!`;

}

console.log(greet("World"));

**Python:**

# Python

def greet(name):

return f"Hello, {name}!"

print(greet("World"))

**Point out:**

* Functions declared differently
* String formatting different
* Indentation matters in Python, optional in JS
* But the LOGIC is the same!

**REPL Introduction (30 min)**

"REPL = Read-Eval-Print-Loop. It's your programming playground."

**JavaScript REPL (Node.js):**

# Start Node.js REPL

node

# Now you're in Node

> 2 + 2

4

> "Hello" + " " + "World"

'Hello World'

> let x = 10

undefined

> x \* 2

20

> .exit // or Ctrl+C twice

**Python REPL:**

# Start Python REPL

python3

# Now you're in Python

>>> 2 + 2

4

>>> "Hello" + " " + "World"

'Hello World'

>>> x = 10

>>> x \* 2

20

>>> exit() # or Ctrl+D

**Live Demo:**

* Open both REPLs side by side
* Try same operations in both
* Show similarities and differences
* Let students follow along

**Resources:**

* JavaScript MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript
* Python Official Tutorial: https://docs.python.org/3/tutorial/
* Node.js Docs: https://nodejs.org/en/docs/
* Python vs JavaScript: https://realpython.com/python-vs-javascript/

**Break (15 min)**

**Session 3: Node.js & Python Environment Setup (1.25 hours)**

**Learning Objectives:**

* Install and verify Node.js
* Install and verify Python
* Understand npm and pip
* Run first script files

**Node.js Installation (30 min)**

**Check if already installed:**

node --version

npm --version

**Installation:**

* **Mac:** brew install node or download from nodejs.org
* **Windows:** Download from nodejs.org
* **Linux:** sudo apt install nodejs npm

**Verify installation:**

node --version # Should see v18+ or v20+

npm --version # Should see 9+ or 10+

**What is Node.js?** "JavaScript was born in browsers. Node.js lets JavaScript run on your computer/server. It's how we build servers, tools, APIs with JavaScript."

**What is npm?** "npm = Node Package Manager. It's like an app store for JavaScript code. Need to work with dates? There's a package. Need to build a server? There's a package."

**Python Installation (30 min)**

**Check if already installed:**

python3 --version

pip3 --version

**Installation:**

* **Mac:** Usually pre-installed, or brew install python3
* **Windows:** Download from python.org
* **Linux:** sudo apt install python3 python3-pip

**Verify installation:**

python3 --version # Should see 3.9+ or 3.10+

pip3 --version # Should see pip for Python 3

**What is pip?** "pip = Python Package Installer. Like npm for Python. Need to work with data? pip install pandas. Need to build websites? pip install flask."

**Create First Script Files (25 min)**

**JavaScript:**

cd ~/bootcamp/week2

mkdir day1

cd day1

touch hello.js

**Edit hello.js:**

// hello.js

console.log("Hello from JavaScript!");

let name = "Developer";

console.log("Welcome, " + name);

// Basic math

let x = 10;

let y = 5;

console.log("10 + 5 =", x + y);

**Run it:**

node hello.js

**Python:**

# Same directory

touch hello.py

**Edit hello.py:**

# hello.py

print("Hello from Python!")

name = "Developer"

print("Welcome, " + name)

# Basic math

x = 10

y = 5

print("10 + 5 =", x + y)

**Run it:**

python3 hello.py

**Students Exercise:** "Create your own hello scripts. Make them print:

1. Your name
2. Your favorite number multiplied by 2
3. A greeting message"

**Lunch Break (30 min)**

**Session 4: First Real Programs (1.5 hours)**

**Learning Objectives:**

* Write multi-line programs
* Use comments effectively
* Understand program execution flow
* Practice both languages

**Comments & Documentation (15 min)**

**JavaScript:**

// Single line comment

/\*

Multi-line comment

Can span multiple lines

\*/

/\*\*

\* Documentation comment

\* Used for functions/classes

\*/

**Python:**

# Single line comment

"""

Multi-line comment

(Actually a multi-line string,

but used for comments)

"""

def function():

"""Documentation string (docstring)

Used for functions/classes"""

pass

**Why comments matter:**

* Future you won't remember
* Team members need to understand
* Complex logic needs explanation
* BUT: Good code should be self-documenting

**First Real Program: Calculator (45 min)**

**JavaScript version:**

// calculator.js

console.log("=== Simple Calculator ===");

// Input numbers (hardcoded for now)

let num1 = 10;

let num2 = 5;

// Perform operations

let sum = num1 + num2;

let difference = num1 - num2;

let product = num1 \* num2;

let quotient = num1 / num2;

// Display results

console.log(`${num1} + ${num2} = ${sum}`);

console.log(`${num1} - ${num2} = ${difference}`);

console.log(`${num1} \* ${num2} = ${product}`);

console.log(`${num1} / ${num2} = ${quotient}`);

console.log("\nProgram complete!");

**Python version:**

# calculator.py

print("=== Simple Calculator ===")

# Input numbers (hardcoded for now)

num1 = 10

num2 = 5

# Perform operations

sum\_result = num1 + num2

difference = num1 - num2

product = num1 \* num2

quotient = num1 / num2

# Display results

print(f"{num1} + {num2} = {sum\_result}")

print(f"{num1} - {num2} = {difference}")

print(f"{num1} \* {num2} = {product}")

print(f"{num1} / {num2} = {quotient}")

print("\nProgram complete!")

**Students build these:**

* Type both versions
* Run both
* Compare output
* Modify numbers and rerun

**Second Program: Name Formatter (30 min)**

**JavaScript:**

// nameFormatter.js

let firstName = "Ada";

let lastName = "Lovelace";

let age = 36;

// String concatenation

let fullName1 = firstName + " " + lastName;

console.log("Full name (concatenation):", fullName1);

// Template literals

let fullName2 = `${firstName} ${lastName}`;

console.log("Full name (template):", fullName2);

// Build a sentence

let introduction = `Hello, my name is ${firstName} ${lastName} and I am ${age} years old.`;

console.log(introduction);

// String methods

console.log("Uppercase:", fullName2.toUpperCase());

console.log("Lowercase:", fullName2.toLowerCase());

console.log("Length:", fullName2.length);

**Python:**

# name\_formatter.py

first\_name = "Ada"

last\_name = "Lovelace"

age = 36

# String concatenation

full\_name1 = first\_name + " " + last\_name

print("Full name (concatenation):", full\_name1)

# F-strings

full\_name2 = f"{first\_name} {last\_name}"

print("Full name (f-string):", full\_name2)

# Build a sentence

introduction = f"Hello, my name is {first\_name} {last\_name} and I am {age} years old."

print(introduction)

# String methods

print("Uppercase:", full\_name2.upper())

print("Lowercase:", full\_name2.lower())

print("Length:", len(full\_name2))

**Point out differences:**

* Python uses underscores in names (snake\_case)
* JavaScript uses camelCase
* Python: .upper() vs JavaScript: .toUpperCase()
* Python: len() function vs JavaScript: .length property

**Break (15 min)**

**Session 5: REPL Experimentation (1 hour)**

**Learning Objectives:**

* Use REPL for quick testing
* Experiment without fear
* Discover language features
* Debug errors in safe environment

**REPL Best Practices (15 min)**

"The REPL is your sandbox. Use it to:

* Test ideas quickly
* Learn new syntax
* Debug confusing code
* Explore language features"

**Guided REPL Experiments (30 min)**

**JavaScript Experiments:**

// In Node REPL

> typeof 42

'number'

> typeof "hello"

'string'

> typeof true

'boolean'

> "hello".length

5

> "hello".toUpperCase()

'HELLO'

> [1, 2, 3].length

3

> Math.random()

0.234234... // random number

> Math.round(3.7)

4

> Math.floor(3.7)

3

> "test" + 123

'test123' // JavaScript converts number to string

> 5 / 2

2.5

> 5 % 2

1 // modulo (remainder)

**Python Experiments:**

# In Python REPL

>>> type(42)

<class 'int'>

>>> type("hello")

<class 'str'>

>>> type(True)

<class 'bool'>

>>> "hello".\_\_len\_\_()

5

>>> len("hello")

5

>>> "hello".upper()

'HELLO'

>>> len([1, 2, 3])

3

>>> import random

>>> random.random()

0.345345... # random number

>>> round(3.7)

4

>>> import math

>>> math.floor(3.7)

3

>>> "test" + str(123)

'test123' # Must convert explicitly

>>> 5 / 2

2.5

>>> 5 // 2

2 # integer division

>>> 5 % 2

1 # modulo

**Student Experiments (15 min):**

Give students challenges to solve in REPL:

1. Find the length of your name
2. Convert your name to all caps
3. Calculate 17 \* 23
4. Calculate 100 / 7 (note the difference!)
5. Figure out what % does (modulo operator)
6. Try adding a string and a number (see what happens!)

**Error Exploration:**

// JavaScript

> undefinedVariable

ReferenceError: undefinedVariable is not defined

> "hello" + undefined

'helloundefined' // JavaScript quirk!

> 5 + "5"

'55' // Type coercion

# Python

>>> undefined\_variable

NameError: name 'undefined\_variable' is not defined

>>> "hello" + 5

TypeError: can only concatenate str to str

>>> 5 + int("5")

10 # Explicit conversion

**Key Learning:** "Errors are GOOD. They tell you what's wrong. Don't fear errors - read them."

**Session 6: Daily Standup & Reflection (15 min)**

**Standup Questions:**

1. What's one thing you learned about programming today?
2. Which language feels more comfortable so far? (It's okay to not know!)
3. Any blockers?

**Reflection:** Students update their journal:

cd ~/bootcamp/journal

git pull # Get latest

# Add today's entry

git add .

git commit -m "Add Day 1 Week 2 learnings"

git push

**Prompt:** "Write about your first day of programming. What surprised you?"

**Day 1 Homework Assignment**

**Assignment: "First Programs & REPL Mastery"**

**Due:** Before class tomorrow

**Part 1: Environment Verification (15 min)**

# Verify everything works

node --version

python3 --version

npm --version

pip3 --version

# Create project structure

mkdir ~/bootcamp/week2/homework

cd ~/bootcamp/week2/homework

mkdir day1

cd day1

**Part 2: Four Required Programs**

**Program 1: Personal Info Card (Both Languages)**

JavaScript (personal\_info.js):

// Your personal information

let fullName = "Your Name";

let age = 25;

let city = "Your City";

let favoriteLanguage = "JavaScript or Python?";

// Display formatted

console.log("=== Personal Info Card ===");

console.log(`Name: ${fullName}`);

console.log(`Age: ${age}`);

console.log(`Location: ${city}`);

console.log(`Favorite Programming Language: ${favoriteLanguage}`);

console.log(`\nIn 5 years, I'll be ${age + 5} years old!`);

Python (personal\_info.py):

# Your personal information

full\_name = "Your Name"

age = 25

city = "Your City"

favorite\_language = "JavaScript or Python?"

# Display formatted

print("=== Personal Info Card ===")

print(f"Name: {full\_name}")

print(f"Age: {age}")

print(f"Location: {city}")

print(f"Favorite Programming Language: {favorite\_language}")

print(f"\nIn 5 years, I'll be {age + 5} years old!")

**Program 2: Math Explorer (Both Languages)**

Calculate and display:

* Your birth year (current year - age)
* How many days old you are (approximately)
* Your age in months
* Your age in hours (approximately)

**Program 3: String Manipulator (Both Languages)**

Take a sentence and:

* Display it in uppercase
* Display it in lowercase
* Display its length
* Display it repeated 3 times
* Display just the first word

**Program 4: REPL Discovery**

Document 10 interesting things you discovered in the REPL. Format:

# REPL Discoveries

## 1. [What you tried]

\*\*JavaScript:\*\*

[code and result]

\*\*Python:\*\*

[code and result]

\*\*What I learned:\*\*

[explanation]

**Part 3: Comparative Analysis (30 min)**

Write a document: "JavaScript vs Python - First Impressions"

Answer:

* Which felt more natural to you? Why?
* What's one thing you like about JavaScript?
* What's one thing you like about Python?
* What's confusing about JavaScript?
* What's confusing about Python?
* Which would you choose for a website? Why?

**Part 4: Git Workflow**

# Initialize git in homework directory

git init

git add .

git commit -m "Complete Day 1 homework: First programs"

# Create GitHub repo and push

# (Follow Week 1 process)

**Submission:**

* Push to GitHub
* Share repository link in class Slack/platform
* Be ready to demo one program tomorrow

**Reading (Optional but Recommended):**

* JavaScript Basics: https://javascript.info/first-steps
* Python Basics: https://realpython.com/python-first-steps/
* Why Learn Multiple Languages: https://www.freecodecamp.org/news/why-you-should-learn-multiple-programming-languages/

**Day 1 Teaching Notes & Tips**

**Time Management:**

* Programming takes longer than expected
* Build in buffer time for technical issues
* Some students will fly ahead, others will struggle
* Have extension exercises ready for fast students

**Common Day 1 Issues:**

1. **Installation Problems:**
   * Have Node.js and Python installers ready on USB
   * Pre-test installations on different OS
   * Pair students with same OS for troubleshooting
2. **"Which language should I focus on?"**
   * Answer: "Both equally"
   * Explain: The concepts matter most
   * Reassure: By end of week, both will click
3. **Syntax Confusion:**
   * Normal! They're learning two
   * Keep side-by-side examples visible
   * Make a cheat sheet for the wall
4. **Feeling Overwhelmed:**
   * Validate: "Yes, this is a lot"
   * Normalize: "Everyone feels this way"
   * Perspective: "Week 1 felt overwhelming too, right? Now look at you!"

**Creating Learning Culture:**

* Celebrate first programs: "You're a programmer now!"
* Share screens: Let students show their work
* Normalize errors: "Errors mean you're trying new things"
* Pair programming: Match strong/struggling students

**Assessment Signals:**

* Who asks good questions?
* Who helps others?
* Who's experimenting vs just copying?
* Who's frustrated vs curious?

**DAY 2: Tuesday, February 25, 2025**

**"Data Types & Variables - The Building Blocks" (6.5 hours)**

**Schedule:**

* 08:30-09:00: Daily Standup & Day 1 Showcase
* 09:00-10:30: Variables & Data Types Deep Dive
* 10:30-10:45: Break
* 10:45-12:00: Type Conversion & Operators
* 12:00-12:30: Lunch Break
* 12:30-14:00: Working with Numbers
* 14:00-14:15: Break
* 14:15-15:15: Working with Strings
* 15:15-15:30: Daily Standup & Reflection

**Session 1: Daily Standup & Day 1 Showcase (30 min)**

**Standup (15 min):**

1. What program did you complete last night?
2. What's one difference between JS and Python you noticed?
3. Any blockers?

**Showcase (15 min):**

* 3-4 volunteers demo one homework program
* Class gives positive feedback
* Instructor highlights good practices

**Session 2: Variables & Data Types Deep Dive (1.5 hours)**

**Learning Objectives:**

* Master variable declaration in both languages
* Understand primitive data types
* Learn type checking and conversion
* Understand mutability vs immutability

**Variables: The Container Analogy (15 min)**

"Variables are labeled containers that hold data."

Draw on whiteboard:

[box labeled "age"] contains: 25

[box labeled "name"] contains: "Ada"

[box labeled "isStudent"] contains: true

"In programming:

* You give containers names (variables)
* You put data in them (assignment)
* You can look inside them (access)
* You can change what's inside (reassignment)"

**JavaScript Variables (25 min)**

**Three ways to declare:**

// let - can be reassigned

let age = 25;

age = 26; // OK

// const - cannot be reassigned

const birthYear = 1998;

birthYear = 1999; // ERROR!

// var - old way (avoid)

var oldStyle = "don't use this";

**When to use which:**

// Use const by default

const PI = 3.14159;

const userName = "Ada";

// Use let when you need to reassign

let score = 0;

score = score + 10; // OK

let currentLevel = 1;

currentLevel = 2; // OK

**Best Practice:** "Start with const. Change to let only if you need to reassign."

**Python Variables (20 min)**

# Python has just one way: assignment

age = 25

age = 26 # OK - can reassign

name = "Ada"

name = "Grace" # OK

# No "const" in Python (by convention, CAPITALS = constant)

PI = 3.14159

# Programmer knows not to change this (not enforced)

**Data Types (30 min)**

**JavaScript Types:**

// Number (no distinction between int/float)

let integer = 42;

let decimal = 3.14;

let negative = -10;

console.log(typeof integer); // "number"

// String

let single = 'Hello';

let double = "World";

let template = `Hello ${name}`;

console.log(typeof single); // "string"

// Boolean

let isTrue = true;

let isFalse = false;

console.log(typeof isTrue); // "boolean"

// Undefined

let notAssigned;

console.log(notAssigned); // undefined

console.log(typeof notAssigned); // "undefined"

// Null

let empty = null;

console.log(typeof empty); // "object" (JavaScript quirk!)

**Python Types:**

# Integer

integer = 42

print(type(integer)) # <class 'int'>

# Float

decimal = 3.14

print(type(decimal)) # <class 'float'>

# String

single = 'Hello'

double = "World"

fstring = f"Hello {name}"

print(type(single)) # <class 'str'>

# Boolean (capitalized!)

is\_true = True

is\_false = False

print(type(is\_true)) # <class 'bool'>

# None (Python's null/undefined)

empty = None

print(type(empty)) # <class 'NoneType'>

**Side-by-Side Comparison:**

JavaScript Python

--------------------------------

let/const just =

number int/float

string str

boolean bool

undefined/null None

typeof type()

**Break (15 min)**

**Session 3: Type Conversion & Operators (1.25 hours)**

**Learning Objectives:**

* Convert between types safely
* Understand implicit vs explicit conversion
* Master arithmetic operators
* Learn comparison operators

**Type Conversion (30 min)**

**JavaScript:**

// Number to String

let num = 42;

let str1 = String(num); // "42"

let str2 = num.toString(); // "42"

let str3 = "" + num; // "42" (implicit)

// String to Number

let str = "42";

let num1 = Number(str); // 42

let num2 = parseInt(str); // 42 (integer)

let num3 = parseFloat("3.14"); // 3.14 (decimal)

let num4 = +str; // 42 (implicit)

// String to Boolean

let bool1 = Boolean("hello"); // true (non-empty string)

let bool2 = Boolean(""); // false (empty string)

// Common mistakes

console.log("5" + 5); // "55" (string concat!)

console.log("5" - 5); // 0 (math!)

console.log(Number("hello")); // NaN (Not a Number)

**Python:**

# Number to String

num = 42

str1 = str(num) # "42"

# String to Number

string = "42"

num1 = int(string) # 42 (integer)

num2 = float("3.14") # 3.14 (decimal)

# String to Boolean

bool1 = bool("hello") # True (non-empty)

bool2 = bool("") # False (empty)

# Common mistakes

# print("5" + 5) # TypeError! Must be explicit

print("5" + str(5)) # "55"

print(int("5") + 5) # 10

# print(int("hello")) # ValueError!

**Key Difference:**

* JavaScript: Loose/implicit conversion (can cause bugs!)
* Python: Strict/explicit conversion (safer, more verbose)

**Arithmetic Operators (25 min)**

// JavaScript

let a = 10;

let b = 3;

console.log(a + b); // 13 (addition)

console.log(a - b); // 7 (subtraction)

console.log(a \* b); // 30 (multiplication)

console.log(a / b); // 3.3333... (division, always float)

console.log(a % b); // 1 (modulo/remainder)

console.log(a \*\* b); // 1000 (exponent: 10^3)

// Increment/Decrement

let x = 5;

x++; // x is now 6

x--; // x is now 5

x += 10; // x is now 15

x \*= 2; // x is now 30

# Python

a = 10

b = 3

print(a + b) # 13 (addition)

print(a - b) # 7 (subtraction)

print(a \* b) # 30 (multiplication)

print(a / b) # 3.3333... (division, always float)

print(a // b) # 3 (floor division, integer result)

print(a % b) # 1 (modulo/remainder)

print(a \*\* b) # 1000 (exponent: 10^3)

# No ++ or -- in Python!

x = 5

x += 1 # x is now 6

x -= 1 # x is now 5

x += 10 # x is now 15

x \*= 2 # x is now 30

**Comparison Operators (30 min)**

// JavaScript

console.log(5 == "5"); // true (loose equality, converts types)

console.log(5 === "5"); // false (strict equality, checks type too)

console.log(5 != "5"); // false

console.log(5 !== "5"); // true

console.log(10 > 5); // true

console.log(10 < 5); // false

console.log(10 >= 10); // true

console.log(10 <= 9); // false

// ALWAYS use === and !== (strict)

# Python

print(5 == "5") # False (always strict!)

print(5 != "5") # True

print(10 > 5) # True

print(10 < 5) # False

print(10 >= 10) # True

print(10 <= 9) # False

# Python has additional: is, is not

x = None

print(x is None) # True

print(x is not None) # False

**Lunch Break (30 min)**

**Session 4: Working with Numbers (1.5 hours)**

**Learning Objectives:**

* Use Math methods effectively
* Handle edge cases (division by zero, NaN)
* Format numbers for display
* Build practical number applications

**Math Operations (30 min)**

**JavaScript Math Object:**

// Random numbers

console.log(Math.random()); // 0-1 random

console.log(Math.random() \* 100); // 0-100 random

console.log(Math.floor(Math.random() \* 100)); // 0-99 integer

// Rounding

console.log(Math.round(3.7)); // 4

console.log(Math.round(3.2)); // 3

console.log(Math.floor(3.9)); // 3 (always down)

console.log(Math.ceil(3.1)); // 4 (always up)

// Min/Max

console.log(Math.min(5, 10, 3, 8)); // 3

console.log(Math.max(5, 10, 3, 8)); // 10

// Power and roots

console.log(Math.pow(2, 3)); // 8 (2^3)

console.log(Math.sqrt(16)); // 4

// Absolute value

console.log(Math.abs(-5)); // 5

// Constants

console.log(Math.PI); // 3.14159...

console.log(Math.E); // 2.718...

**Python Math Module:**

import math

import random

# Random numbers

print(random.random()) # 0-1 random

print(random.random() \* 100) # 0-100 random

print(random.randint(0, 99)) # 0-99 integer

# Rounding

print(round(3.7)) # 4

print(round(3.2)) # 3

print(math.floor(3.9)) # 3 (always down)

print(math.ceil(3.1)) # 4 (always up)

# Min/Max

print(min(5, 10, 3, 8)) # 3

print(max(5, 10, 3, 8)) # 10

# Power and roots

print(math.pow(2, 3)) # 8.0 (2^3)

print(math.sqrt(16)) # 4.0

# Absolute value

print(abs(-5)) # 5

# Constants

print(math.pi) # 3.14159...

print(math.e) # 2.718...

**Number Formatting (20 min)**

**JavaScript:**

let num = 123.456789;

// Decimal places

console.log(num.toFixed(2)); // "123.46" (string!)

console.log(parseFloat(num.toFixed(2))); // 123.46 (number)

// Localization

let price = 1234.56;

console.log(price.toLocaleString('en-US', {

style: 'currency',

currency: 'USD'

})); // "$1,234.56"

console.log(price.toLocaleString('de-AT', {

style: 'currency',

currency: 'EUR'

})); // "1.234,56 €"

**Python:**

num = 123.456789

# Decimal places

print(f"{num:.2f}") # "123.46"

print(round(num, 2)) # 123.46

# Formatting with commas

price = 1234.56

print(f"${price:,.2f}") # "$1,234.56"

print(f"€{price:\_.2f}") # "€1\_234.56"

**Exercise: Build a Tip Calculator (40 min)**

Students build in both languages:

**JavaScript:**

// tip\_calculator.js

console.log("=== Tip Calculator ===");

const billAmount = 50.00;

const tipPercentage = 18;

const tipAmount = billAmount \* (tipPercentage / 100);

const totalAmount = billAmount + tipAmount;

console.log(`Bill Amount: $${billAmount.toFixed(2)}`);

console.log(`Tip (${tipPercentage}%): $${tipAmount.toFixed(2)}`);

console.log(`Total: $${totalAmount.toFixed(2)}`);

// Split bill

const numberOfPeople = 4;

const perPerson = totalAmount / numberOfPeople;

console.log(`\nPer person (${numberOfPeople} people): $${perPerson.toFixed(2)}`);

**Python:**

# tip\_calculator.py

print("=== Tip Calculator ===")

bill\_amount = 50.00

tip\_percentage = 18

tip\_amount = bill\_amount \* (tip\_percentage / 100)

total\_amount = bill\_amount + tip\_amount

print(f"Bill Amount: ${bill\_amount:.2f}")

print(f"Tip ({tip\_percentage}%): ${tip\_amount:.2f}")

print(f"Total: ${total\_amount:.2f}")

# Split bill

number\_of\_people = 4

per\_person = total\_amount / number\_of\_people

print(f"\nPer person ({number\_of\_people} people): ${per\_person:.2f}")

**Break (15 min)**

**Session 5: Working with Strings (1 hour)**

**Learning Objectives:**

* Master string manipulation
* Learn string methods in both languages
* Build text-processing applications
* Understand string immutability

**String Basics (20 min)**

**JavaScript:**

// Creation

let str1 = 'Single quotes';

let str2 = "Double quotes";

let str3 = `Template literal with ${str1}`;

// Concatenation

let firstName = "Ada";

let lastName = "Lovelace";

let fullName = firstName + " " + lastName;

let fullName2 = `${firstName} ${lastName}`; // Better!

// Length

console.log(fullName.length); // 12

// Accessing characters

console.log(fullName[0]); // 'A'

console.log(fullName.charAt(0)); // 'A'

console.log(fullName[fullName.length - 1]); // 'e'

// Strings are immutable

let text = "hello";

text[0] = "H"; // Doesn't work!

text = "H" + text.slice(1); // "Hello" (create new string)

**Python:**

# Creation

str1 = 'Single quotes'

str2 = "Double quotes"

str3 = f'F-string with {str1}'

# Concatenation

first\_name = "Ada"

last\_name = "Lovelace"

full\_name = first\_name + " " + last\_name

full\_name2 = f"{first\_name} {last\_name}" # Better!

# Length

print(len(full\_name)) # 12

# Accessing characters

print(full\_name[0]) # 'A'

print(full\_name[-1]) # 'e' (negative indexing!)

# Strings are immutable

text = "hello"

# text[0] = "H" # TypeError!

text = "H" + text[1:] # "Hello" (create new string)

**String Methods (30 min)**

**JavaScript:**

let text = " Hello, World! ";

// Case conversion

console.log(text.toUpperCase()); // " HELLO, WORLD! "

console.log(text.toLowerCase()); // " hello, world! "

// Trimming

console.log(text.trim()); // "Hello, World!"

console.log(text.trimStart()); // "Hello, World! "

console.log(text.trimEnd()); // " Hello, World!"

// Searching

console.log(text.includes("World")); // true

console.log(text.startsWith(" H")); // true

console.log(text.endsWith("! ")); // true

console.log(text.indexOf("World")); // 9

console.log(text.indexOf("xyz")); // -1 (not found)

// Replacing

console.log(text.replace("World", "JavaScript")); // " Hello, JavaScript! "

console.log(text.replaceAll("l", "L")); // " HeLLo, WorLd! "

// Splitting

let words = "apple,banana,orange".split(",");

console.log(words); // ['apple', 'banana', 'orange']

// Joining

let joined = words.join(" - ");

console.log(joined); // "apple - banana - orange"

// Repeating

console.log("Ha".repeat(3)); // "HaHaHa"

// Extracting

console.log(text.slice(2, 7)); // "Hello"

console.log(text.substring(2, 7)); // "Hello"

**Python:**

text = " Hello, World! "

# Case conversion

print(text.upper()) # " HELLO, WORLD! "

print(text.lower()) # " hello, world! "

print(text.title()) # " Hello, World! "

print(text.capitalize()) # " hello, world! "

# Trimming

print(text.strip()) # "Hello, World!"

print(text.lstrip()) # "Hello, World! "

print(text.rstrip()) # " Hello, World!"

# Searching

print("World" in text) # True

print(text.startswith(" H")) # True

print(text.endswith("! ")) # True

print(text.find("World")) # 9

print(text.find("xyz")) # -1 (not found)

# Replacing

print(text.replace("World", "Python")) # " Hello, Python! "

# Splitting

words = "apple,banana,orange".split(",")

print(words) # ['apple', 'banana', 'orange']

# Joining

joined = " - ".join(words)

print(joined) # "apple - banana - orange"

# Repeating

print("Ha" \* 3) # "HaHaHa"

# Extracting (slicing)

print(text[2:7]) # "Hello"

print(text[:5]) # " Hel"

print(text[7:]) # ", World! "

**Exercise: Text Analyzer (10 min)**

Build a program that analyzes a sentence:

* Count total characters
* Count characters without spaces
* Count words
* Display first word
* Display last word
* Convert to title case
* Replace a word

**Session 6: Daily Standup & Reflection (15 min)**

**Standup:**

1. What's your favorite string or number method?
2. What's still confusing about variables/types?
3. Blockers?

**Reflection:** Journal about today's learning

**Day 2 Homework Assignment**

**Assignment: "Data Types Mastery"**

**Program 1: Personal Budget Calculator**

* Income amount
* List expenses (rent, food, transport, entertainment)
* Calculate total expenses
* Calculate remaining money
* Calculate percentage of income spent
* Display everything formatted nicely

**Program 2: String Formatter**

* Take your full name
* Display it in 5 different formats
* Extract initials
* Count vowels
* Reverse the name
* Create username (lowercase, no spaces)

**Program 3: Number Games**

* Generate random number 1-100
* Calculate if it's even or odd (using modulo)
* Calculate its square
* Calculate its square root
* Round it to nearest 10

**Program 4: Type Conversion Challenge** Try these and document what happens:

// JavaScript

"5" + 5

"5" - 5

true + 1

false + 1

"hello" \* 2

# Python

# Each of these - what works? What errors?

"5" + 5

"5" + str(5)

int("5") + 5

True + 1

False + 1

**Documentation:** Create a comparison chart:

* When to use const vs let (JS) vs just assignment (Python)
* Number types in JS vs Python
* String methods comparison
* Type conversion differences

**DAY 3: Wednesday, February 26, 2025**

**"Control Flow - Making Decisions" (6.5 hours)**

**Schedule:**

* 08:30-09:00: Daily Standup
* 09:00-10:30: Conditional Statements (if/else)
* 10:30-10:45: Break
* 10:45-12:00: Logical Operators & Complex Conditions
* 12:00-12:30: Lunch Break
* 12:30-14:00: Loops Part 1 (for loops)
* 14:00-14:15: Break
* 14:15-15:15: Loops Part 2 (while loops)
* 15:15-15:30: Daily Standup & Reflection

**Key Topics Summary:**

**Session 1: Conditional Statements**

* if/else/else if (JS) vs if/elif/else (Python)
* Boolean expressions
* Truthy/falsy values
* Nested conditions
* Ternary operators

**Session 2: Logical Operators**

* AND, OR, NOT
* Truth tables
* Complex conditions
* Short-circuit evaluation

**Session 3: For Loops**

* Classic for loop (JS) vs range-based (Python)
* Iterating over arrays/lists
* Loop patterns
* Common use cases

**Session 4: While Loops**

* while loop syntax
* break and continue
* When to use for vs while
* Avoiding infinite loops

**Day 3 Homework:**

* FizzBuzz challenge
* Number guessing game
* Pattern printer
* Menu system

**DAY 4: Thursday, February 27, 2025**

**"Functions & Scope" (6.5 hours)**

**Schedule:**

* 08:30-09:00: Daily Standup
* 09:00-10:30: Function Fundamentals
* 10:30-10:45: Break
* 10:45-12:00: Parameters, Arguments & Return Values
* 12:00-12:30: Lunch Break
* 12:30-14:00: Scope (Global vs Local)
* 14:00-14:15: Break
* 14:15-15:15: Arrow Functions (JS) & Lambda (Python)
* 15:15-15:30: Daily Standup & Reflection

**Key Topics Summary:**

**Session 1: Function Fundamentals**

* What are functions and why use them
* Function declaration syntax (JS vs Python)
* Calling functions
* DRY principle (Don't Repeat Yourself)

**Session 2: Parameters & Arguments**

* Positional parameters
* Default parameters
* Keyword arguments (Python)
* Return values
* Multiple return values

**Session 3: Scope**

* Global scope
* Local scope
* Variable shadowing
* Scope chain
* Best practices

**Session 4: Advanced Function Patterns**

* Arrow functions (JS)
* Lambda functions (Python)
* Functions as values
* Callback patterns

**Day 4 Homework:**

* Build utility function library
* Temperature converter
* Password validator
* Calculator with functions

**DAY 5: Friday, February 28, 2025**

**"Testing & Week Review" (4 hours)**

**Schedule:**

* 08:30-09:00: Daily Standup & Week Reflection
* 09:00-10:00: Introduction to Testing
* 10:00-10:15: Break
* 10:15-11:30: Week 2 Integration Challenge
* 11:30-12:00: Sprint Review & Presentations
* 12:00-12:30: Sprint Retrospective

**Key Topics Summary:**

**Session 1: Testing Mindset**

* Why test code?
* Manual vs automated testing
* Writing simple tests
* Test-driven thinking

**Session 2: Week Integration**

* Complete dual-language challenge
* Solve 10 problems in both JS and Python
* Document differences
* Peer code review

**Session 3: Sprint Review & Retrospective**

* Present solutions
* Demonstrate learning
* What went well
* What to improve
* Prepare for Week 3

**Week 2 Resources Library**

**JavaScript Learning**

* JavaScript.info: https://javascript.info/
* MDN JavaScript Guide: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
* Eloquent JavaScript: https://eloquentjavascript.net/

**Python Learning**

* Python Official Tutorial: https://docs.python.org/3/tutorial/
* Real Python: https://realpython.com/
* Python for Beginners: https://www.python.org/about/gettingstarted/

**Practice Problems**

* Codewars: https://www.codewars.com/
* LeetCode (Easy): https://leetcode.com/
* HackerRank: https://www.hackerrank.com/
* Exercism: https://exercism.org/

**Testing**

* Jest Documentation: https://jestjs.io/
* Pytest Documentation: https://docs.pytest.org/

**Week 2 Assessment Rubric**

**Programming Fundamentals (40%)**

* Can write basic programs in both languages
* Understands variables, types, operators
* Uses control flow appropriately
* Writes functions effectively

**Code Quality (20%)**

* Clean, readable code
* Appropriate comments
* Good naming conventions
* Consistent style

**Dual-Language Competency (25%)**

* Completes assignments in both languages
* Understands differences
* Makes intelligent comparisons
* Can justify language choices

**Professional Practices (15%)**

* Git commits for all work
* Helps classmates
* Participates in code review
* Attends standup/retrospective
* Completes homework on time

**Common Week 2 Issues & Solutions**

**Issue: "Which language should I focus on?"**

**Solution:**

* Answer: Both equally
* The concepts are what matter
* Syntax is just details
* Knowing both makes you more valuable

**Issue: Syntax confusion between languages**

**Solution:**

* Create side-by-side reference sheet
* Practice translating between languages
* Use REPL to test syntax
* Compare code snippets constantly

**Issue: "I don't understand functions"**

**Solution:**

* Use real-world analogies (recipes, machines)
* Start with simple examples
* Build complexity gradually
* Practice writing many small functions

**Issue: Feeling overwhelmed**

**Solution:**

* Normal for Week 2!
* Validate feelings
* Show progress: "Look what you built!"
* Break into smaller pieces
* Encourage peer support

**Issue: Off-by-one errors in loops**

**Solution:**

* Very common!
* Teach debugging technique
* Use print/console.log to trace
* Understand range differences (0-based vs 1-based)

**Week 2 Success Indicators**

By end of Week 2, students should:

* Write basic programs in both languages without reference
* Understand when to use each control structure
* Create and use functions
* Debug their own code systematically
* Read and understand code written by others
* Explain code decisions
* Use Git for all work
* Help classmates
* Feel competent (not expert, but competent)
* Be excited to learn more

**Most Important:** Students should think "I'm a programmer now" not "I'm learning to code."

**Teaching Philosophy Recap**

**The Dual-Language Advantage**

"You're not learning JavaScript and Python. You're learning PROGRAMMING using JavaScript and Python."

This distinction is crucial. By Week 2 end, students should:

* See patterns across languages
* Understand concepts transcend syntax
* Make informed language choices
* Feel confident in their ability to learn new languages

**Week 2 as Transformation**

Week 1: Tool users  
Week 2: Programmers  
Week 3+: Professional developers

Week 2 is where the transformation happens. It's hard, it's overwhelming, and it's supposed to be. Your job is to make them feel capable even when they're confused.

**Practice Over Perfection**

By end of Week 2, students will have written:

* 20+ complete programs
* 100+ lines of code in each language
* Dozens of functions
* Multiple program types

That's the goal. Not perfect code, but practiced code. Not bug-free, but debugged. Not polished, but progressing.

**Remember:** Week 2 is where students transform from "tool users" to "programmers." It's the hardest week of the program. They'll feel overwhelmed. That's expected. Your job is to normalize the struggle while celebrating the progress.

"Yes, this is hard. Look how much you've learned in just two weeks. You started with nothing. Now you can make a computer do what you want in TWO languages. That's incredible."

*Digital Campus Vorarlberg | Feldkirch, Austria*  
*Next Cohort: February 17, 2025*