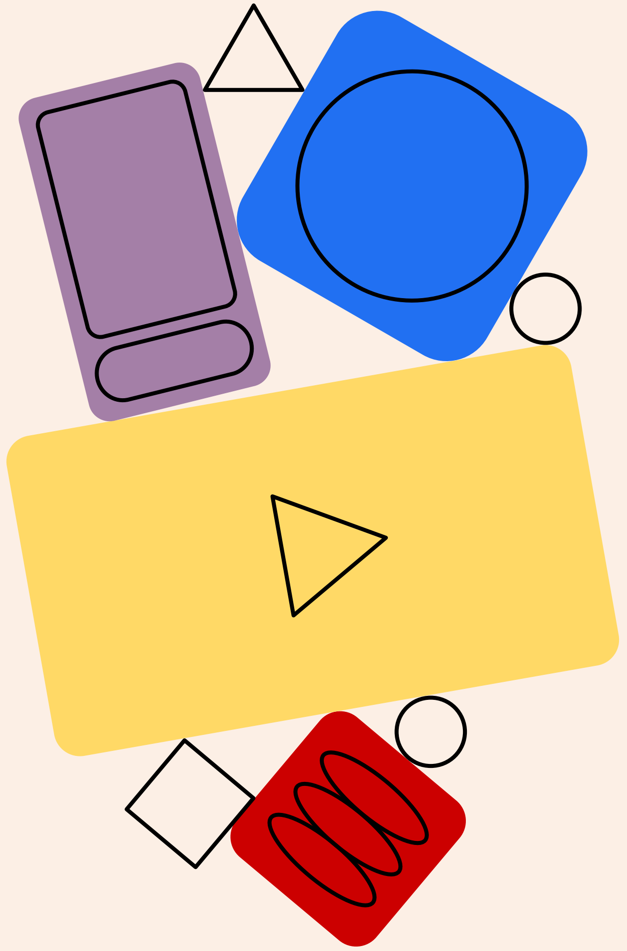
Week 4: Numbers and Math

BHCC CSC 125-03 Python Programming



Today's Agena

- Week 3 Review - String Operations
- Week 4 Introduction: Math Operators, Assignment, Precedence, and Boolean expressions

Homework Review

String Methods for Text Processing

- Essential methods: .upper(), .lower(), .strip(), .replace(), .title()
- Method chaining to apply multiple operations in sequence
- Cleaning and formatting user input data

String Indexing and Slicing

- Accessing individual characters with [index] notation (starting from 0)
- Negative indexing to count from the end of strings
- Slicing with [start:end] to extract substrings
- Common patterns like extracting usernames from email addresses

F-String Formatting

- Modern syntax f"text with {variable}" for dynamic output
- Embedding expressions and method calls inside f-strings
- Replacing older concatenation methods with cleaner formatting

Understanding Error Messages

- IndexError when accessing characters that don't exist
- AttributeError from misspelled method names (case sensitivity)
- NameError from variable typos or undefined variables
- Reading error messages systematically instead of panicking

Debugging Strategies

- Using print() statements to check variable contents and types
- Validating string length before indexing operations
- Converting data types before performing operations
- Testing with simple examples when code isn't working

Week 4 - Numbers and Math

Arithmetic Operators for Calculations

- Seven operators: +, -, *, /, //, %, ** with specific use cases
- Floor division (//) vs regular division (/) - whole numbers vs decimals
- Modulo (%) for remainders, even/odd checking, and time conversions
- Exponentiation (**) for power calculations

Operator Precedence Rules

- PEMDAS-like order: parentheses, exponents, multiplication/division, addition/subtraction
- Using parentheses to override precedence and clarify intentions
- Common precedence mistakes in complex mathematical expressions

Comparison Operators for Logic

- Six comparison operators: ==, !=, <, >, <=, >= returning True/False
- Chained comparisons unique to Python (e.g., 18 <= age <= 65)
- Comparing different data types and Converting types

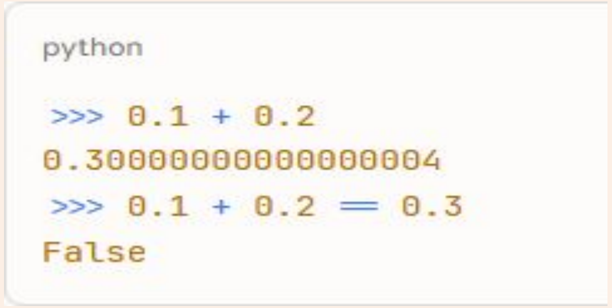
Assignment Operators for Efficiency

- Shorthand operators: +=, -=, *=, /= as alternatives to longer syntax
- Common patterns like counters (count += 1) and accumulators (total += value)
- Understanding equivalence (x += 5 same as x = x + 5)

Boolean Expressions and Decision Making

- Storing comparison results in boolean variables
- Combining arithmetic and comparison operations
- Foundation concepts for upcoming conditional logic (if statements)

The 0.1 + 0.2 Problem



python

```
>>> 0.1 + 0.2
0.30000000000000004
>>> 0.1 + 0.2 == 0.3
False
```

- In most programming languages (including Python), `0.1 + 0.2` doesn't equal `0.3`
- Computers store numbers in binary (base 2), but decimal fractions like 0.1 can't be represented exactly in binary - just like 1/3 can't be written exactly as a decimal (0.333...). The computer stores a very close approximation, and tiny errors accumulate.

Real Impact: Banking software, scientific calculations, and any precise mathematical work must account for this limitation

Patriot Missile Miss (1991)

The Situation: During the Gulf War, a U.S. Patriot missile system was supposed to intercept an incoming Iraqi Scud missile

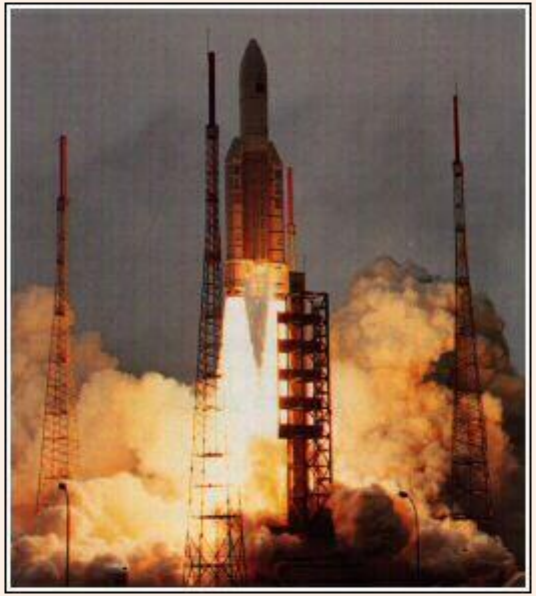
The Math Problem:

- The system's internal clock counted time in tenths of seconds
- But it stored this as a fraction: 1/10
- In binary, 1/10 becomes 0.00011001100110011... (repeating forever)
- The computer chopped off the repeating part, creating a tiny error

The Deadly Result:

- After running for 100 hours, this tiny error added up to 0.34 seconds
- In 0.34 seconds, a Scud missile travels over 500 meters
- The Patriot looked in the wrong place and missed completely

Ariane 5 Rocket Explosion (1996)



The Disaster: European Space Agency's $500 million rocket exploded 40 seconds after launch. Called the 7 billion dollar overflow

The Computer Math Error:

- A horizontal velocity calculation produced a number too large to fit in a 16-bit integer (maximum value: 32,767). The software tried to convert a 64-bit floating point number to a 16-bit integer, causing an "integer overflow."
- The guidance system crashed, the rocket lost control, and the automatic self-destruct activated to prevent ground damage.

The Bug: The code worked fine for the slower Ariane 4 rocket, but Ariane 5 was faster and generated larger velocity numbers that broke the math assumptions.

Both examples show how computer math limitations can have consequences ranging from confusing (decimal arithmetic) to catastrophic (rocket explosions).

Kaggle Introduction

Kaggle is the world's largest data science community and competition platform, used by millions of data scientists and machine learning engineers at companies like Google, Netflix, and Microsoft. We'll use Kaggle's free cloud-based notebook environment that runs Python directly in your web browser with no software installation required, eliminating technical barriers so you can focus on learning programming instead of troubleshooting setup issues.

Download the homework

Go to the course moodle and find the homework for the week: https://online.bhcc.edu/course/view.php?id=43643

Import to Kaggle

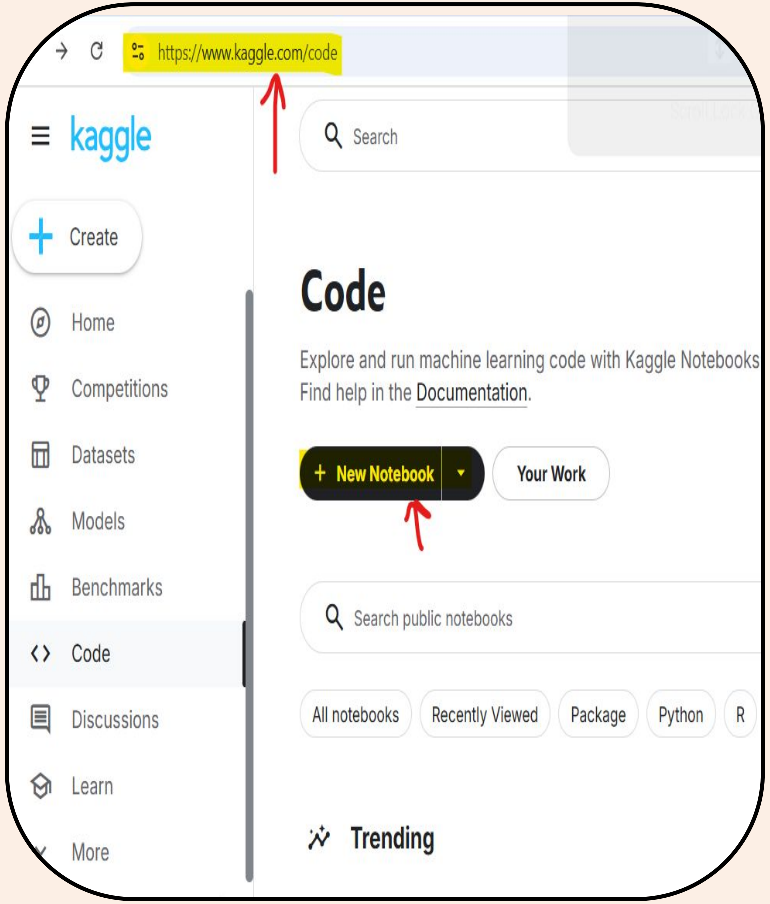
Go to https://www.kaggle.com/code and select ***new notebook***, then file > import notebook

Edit and run the provided cells

You will not need to add new cells, cells can be run by hitting the play icon or using shift + enter

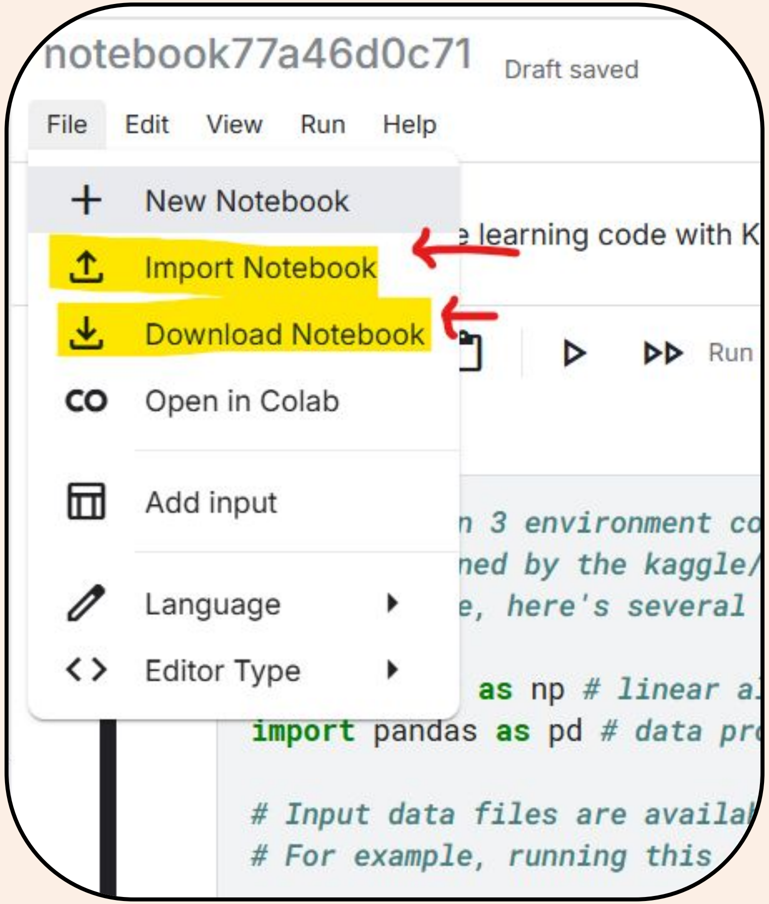
Download and submit to moodle

Make sure the cells are run and the output is showing. Go to file > download notebook then submit to the moodle



Code

Explore and run machine learning code with Kaggle Notebooks
Find help in the Documentation.



```
n 3 environment co
ned by the kaggle/
e, here's several
 as np # linear a
import pandas as pd # data pr

# Input data files are availa
# For example, running this
```