

AI Assisted Coding

Lab Assignment 4.3

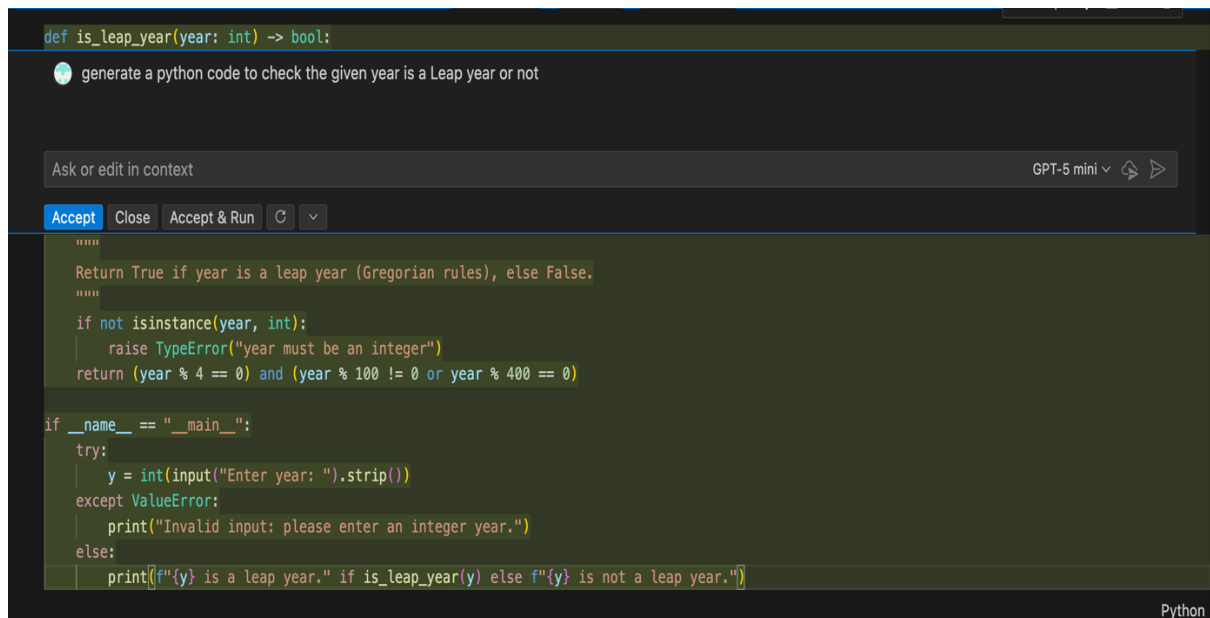
Name : V Sai Krishnachaithanya

Date : 28/01/2026

Hall Ticket no : 2303A52154

Task 1: Zero-Shot Prompting – Leap Year Check

Prompt : generate a python code to check given year is a Leap year or not



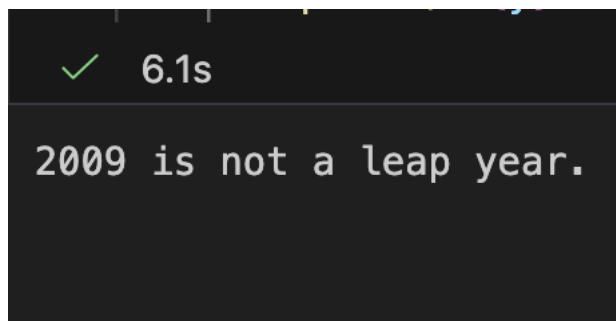
The screenshot shows an AI coding assistant interface. At the top, a prompt is entered: "generate a python code to check the given year is a Leap year or not". Below the prompt, there is a text input field with the placeholder "Ask or edit in context" and a dropdown menu showing "GPT-5 mini". Below the input field, there are buttons: "Accept", "Close", "Accept & Run", and a refresh icon. The main area displays the generated Python code:

```
def is_leap_year(year: int) -> bool:
    """
    Return True if year is a leap year (Gregorian rules), else False.
    """
    if not isinstance(year, int):
        raise TypeError("year must be an integer")
    return (year % 4 == 0) and (year % 100 != 0 or year % 400 == 0)

if __name__ == "__main__":
    try:
        y = int(input("Enter year: ").strip())
    except ValueError:
        print("Invalid input: please enter an integer year.")
    else:
        print(f"{y} is a leap year." if is_leap_year(y) else f"{y} is not a leap year.")
```

The code is written in Python and includes a docstring, type hints, and a main function that takes user input and prints the result.

Output :



The screenshot shows the output of the program. It features a green checkmark, a time of 6.1s, and the text "2009 is not a leap year.".

Task 2: One-Shot Prompting Centimetres to Inches Conversion

Prompt :Generate a python function which converts input 10 centimeters to inches and display it up to 2 decimal points

```
def cm_to_inches(cm: float) -> float:

Generate a python function which converts input 10 centimeters to inches and display it upto 2 decimal points

Ask or edit in context GPT-5 mini

Accept Close Accept & Run C v

"""Convert centimeters to inches (unrounded)."""
return cm / 2.54

def display_10cm_in_inches():
    """Display the conversion of 10 cm to inches rounded to 2 decimal places.
    Uses existing inches_for_10cm variable if available."""
    try:
        val = inches_for_10cm # use notebook variable if present
    except NameError:
        val = cm_to_inches(10)
    print(f"10 cm = {val:.2f} inches")

# call the display function
display_10cm_in_inches()
```

✓ 0.0s Python

Output :

```
✓ 0.0s

10 cm = 3.94 inches
```

Task 3: Few-Shot Prompting – Name Formatting

Prompt :Generate a python function that accept the full name as input and format it as "Last , first". full names are "John Smith" and "Anita Rao"

```
def format_name(full_name: str) -> str:

generate a python function that accept the full name as input and format it as "Last,first". full names are "John Smith" and "Anita Rao"

Ask or edit in context GPT-5 mini

Accept Close Accept & Run C v

"""
Format a full name as "Last,first".
Uses the first token as first name and the last token as last name.
"""
if not isinstance(full_name, str):
    raise TypeError("full_name must be a string")
parts = full_name.strip().split()
if len(parts) < 2:
    raise ValueError("full_name must contain at least first and last name")
first, last = parts[0], parts[-1]
return f"{last},{first}"

# Examples
print(format_name("John Smith")) # -> "Smith,John"
print(format_name("Anita Rao")) # -> "Rao,Anita"
```

Python

Output :

```
✓ 0.0s

"Smith","John"
"Rao","Anita"
```

Task 4: Comparative Analysis – Zero-Shot vs Few-Shot

Prompt :

Generate a python function that count vowels in a string using zero shot and few shot and generate a comparative analysis for both shots with accuracy and readability

```
import inspect

generate a python function that count vowels in a string using zero shot and few shot and generate a comparative analysis for both shots with accuracy and readability

Ask or edit in context GPT-5 mini v ↗ ▶

Accept Close Accept & Run ↻ ⌵

from typing import List, Tuple, Set, Callable

def zero_shot_count_vowels(s: str) -> int:
    """Zero-shot: count standard English vowels (a,e,i,o,u), case-insensitive."""
    return sum(1 for ch in s if ch.lower() in set("aeiou"))

def _count_with_vowels(s: str, vowels: Set[str]) -> int:
    return sum(1 for ch in s if ch.lower() in vowels)

def few_shot_train_vowels(examples: List[Tuple[str, int]]) -> Set[str]:
    """
    Few-shot: infer an extended vowel set from a few labeled examples.
    Starts from base vowels and greedily adds characters that reduce total absolute error.
    Returns a set of lowercase vowel characters to use (e.g. {'a','e','i','o','u','y'}).
    """
    base = set("aeiou")
    # Candidate characters to consider (lowercase alphabetic) seen in examples but not in base
    candidates = set(
        ch.lower()
        for s, _ in examples
        for ch in s
        if ch.isalpha() and ch.lower() not in base
    )
    def total_error(vset: Set[str]) -> int:
        return sum(abs(_count_with_vowels(s, vset) - true) for s, true in examples)
```

Output :

```
✓ 0.0s

Comparative analysis (zero-shot vs few-shot):
- Zero-shot vowels: aeiou
- Few-shot learned vowels: aeiouy

Performance on eval set (8 samples):
- Zero-shot: accuracy=50.00%, MAE=0.625
- Few-shot: accuracy=100.00%, MAE=0.000

Readability heuristic (0..1, higher is more readable):
- Zero-shot function readability: 1.0
- Few-shot function readability: 1.0

Notes:
- Zero-shot is simple and fast; it uses the standard vowel set (a,e,i,o,u).
- Few-shot adapts to examples (e.g., it can learn to treat 'y' as a vowel).
- Accuracy gains depend on how representative the few-shot examples are.
```

Task 5: Few-Shot Prompting – File Handling

Prompt :

Write a Python function that reads a text file and counts the number of words.

```
def count_lines_in_file(path: str, encoding: str = "utf-8") -> int:
    """
    Read a text file and count the number of lines using Python.
    Examples:
    Example 1:
    File content:
    Hello
    Welcome to Python
    File handling is easy
    Output:
    Number of lines: 3
    """
    with open(path, "r", encoding=encoding) as f:
        return sum(1 for _ in f)

# Example usage (creates a sample file like in your example and counts lines)
if __name__ == "__main__":
    sample_path = "sample.txt"
    with open(sample_path, "w", encoding="utf-8") as f:
        f.write("Hello\nWelcome to Python\nFile handling is easy\n")
    print(f"Number of lines: {count_lines_in_file(sample_path)}")
```

Output:

```
✓ 0.0s

Number of lines: 3
```