```
"""
CP1404/CP5632 Practical
File and class example - opens/reads a file, stores in objects of custom class
(contains multiple versions for demonstration: using csv and namedtuple)
"""

import csv
from collections import namedtuple

from programming_language import ProgrammingLanguage


def main():
    """Read file of programming language details, save as objects, display."""
    languages = []
    # open the file for reading
    in_file = open('languages.csv', 'r')

    # file format is like: Language,Typing,Reflection,Year
    # 'consume' the first line (header) - we don't need its contents
    in_file.readline()

    # all other lines are language data
    for line in in_file:
        # print(repr(line))  # debugging

        # strip newline from end and split it into parts (CSV)
        parts = line.strip().split(',')
        # print(parts)  # debugging

        # reflection is stored as a string (Yes/No) and we want a Boolean
        reflection = parts[2] == "Yes"
        pointer_arithmetic = parts[3] == "Yes"

        # construct a ProgrammingLanguage object using the elements
        # year should be an integer
        language = ProgrammingLanguage(parts[0], parts[1], reflection, pointer_arithmetic, int(parts[4]))

        # add the language we've just constructed to the list
        languages.append(language)

    # close the file as soon as we've finished reading it
    in_file.close()

    # loop through and display all languages (using their str method)
    for language in languages:
        print(language)


main()


def using_csv():
    """Language file reader version using the csv module."""
    # first, open the file for reading - note: specify newline
    # to avoid quoted \n in strings being considered a new record
    in_file = open('languages.csv', 'r', newline='')
    in_file.readline()
    reader = csv.reader(in_file)  # use default dialect, Excel
    for row in reader:
        print(row)

    in_file.close()


# using_csv()


def using_namedtuple():
    """Language file reader version using a named tuple."""
    in_file = open('languages.csv', 'r', newline='')
    file_field_names = in_file.readline().strip().split(',')
    print(file_field_names)
    # Language will be a new subclass of the tuple data type class
    Language = namedtuple('Language', 'name, typing, reflection, pointer_arithmetic, year')
    reader = csv.reader(in_file)  # use default dialect, Excel

    for row in reader:
        # print(row)
        language = Language._make(row)
        print(repr(language))
    in_file.close()


# using_namedtuple()
```

```
def using_csv_namedtuple():
    """Language file reader version using both csv module and named tuple."""
    Language = namedtuple('Language', 'name, typing, reflection, pointer_arithmetic, year')
    in_file = open("languages.csv", "r")
    in_file.readline()
    for language in map(Language._make, csv.reader(in_file)):
        print(language.name, 'was released in', language.year)
        print(repr(language))

# using_csv_namedtuple()
```