SDEV 1001

Programming Fundamentals

Dictionaries - 1

A LEADING POLYTECHNIC COMMITTED TO YOUR SUCCESS

Expectations - What I expect from you

- No Late Assignments

- No Cheating

- Be a good classmate

- Don’t waste your time

- Show up to class

Agenda

On the right is what we will cover today.

- Working with CSV Files in Python
- Reading a CSV File
- Writing to a CSV File
- Filtering and Aggregating CSV Data
- Updating CSV Data

Working with CSV Files in Python

CSV (Comma-Separated Values) files are a standard way to store tabular data. They are widely used for data exchange between programs and for data analysis.

Why is this important?

- CSV files are easy to read and write.
- They are supported by many applications, including spreadsheets and databases.
- Understanding how to work with CSV files is essential for data manipulation and analysis.

Reading a CSV File

```
import csv

def read_books(file_path):
    books = []
    with open(file_path, 'r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            books.append(row)
    return books

books = read_books("books.csv")
for book in books:
    print(book["Title"], "-", book["Author"])
```

- `csv.DictReader` reads each row as a dictionary using the header row as keys.

Writing to a CSV File

```
import csv

def save_scores(file_path, scores):
    with open(file_path, 'w', newline='') as file:
        fieldnames = ["Name", "Score"]
        writer = csv.DictWriter(file, fieldnames=fieldnames)
        writer.writeheader()
        for entry in scores:
            writer.writerow(entry)

scores = [{"Name": "Alice", "Score": 95}, {"Name": "Bob", "Score": 88}]
save_scores("scores.csv", scores)
```

- `csv.DictWriter` writes dictionaries to a CSV file, including headers.

Filtering and Aggregating CSV Data

```
students = read_books("students.csv")  # Assume each row has "Grade" and "Name"
passed = [s for s in students if int(s["Grade"]) ⩾ 60]
print("Students who passed:")
for s in passed:
    print(s["Name"])
```

- Use list comprehensions to filter or process CSV data.

Updating CSV Data

```
def update_prices(file_path):
    products = read_books(file_path)
    for p in products:
        p["Price"] = str(float(p["Price"]) * 1.1)  # Increase price by 10%
    save_scores(file_path, products)

update_prices("products.csv")
```

- Read, modify, and write back to update CSV files.

Summary

- Use the `csv` module for reading and writing CSV files.
- `DictReader` and `DictWriter` make working with headers and dictionaries easy.
- You can filter, aggregate, and update CSV data just like lists of dictionaries.


WE ARE
ESSENTIAL TO ALBERTA
NAIT

Example

Let’s go run a few examples together