Business Programming (using Python)

Zhaohu (Jonathan) Fan

Information Technology Management

Scheller College of Business

Georgia Institute of Technology

September 28, 2023

Main topics

- Data Structures
  - List
  - Dictionaries
  - Exercises

In-class presentation

- **Question?**
  - How can we adjust the **regex pattern** to meet the **new matching criteria** using Python?
    - Current pattern: `^\w{3,}\s(\w\s)?\w{3,}`
  - **Charles McCartney** will present his answer to this question today.

Built-In data structures

Type Name	Example	Description
`list`	`[1, 2, 3]`	Ordered collection
`tuple`	`(1, 2, 3)`	Immutable ordered collection
`dict`	`{'a':1, 'b':2, 'c':3}`	Unordered (key,value) mapping
`set`	`{1, 2, 3}`	Unordered collection of unique values

Correction of final standings: 2023 Tour U.S

- Overview of the final standings for the 2023 Tour U.S
    - Geraint Thomas (Great Britain)
    - George Burdell (GA Tech)
    - Steven Kruijswijk (New Zealand)
    - Listing the issue identified: Incorrect inclusion of George Burdell.

Step 1: Confirmation of error

- Initialize a **list** called `winners` that contains the original standings.
  - Use an `if` statement to check if 'George Burdell, GA Tech' is present in the `winners` list. If found, the program prints **"I found George in the list!"**

Python

```
# List of winners
winners = ['Geraint Thomas, Great Britain', 'George Burdell, GA Tech', 'Steven Kruijsv

# Confirming George's presence in the list
if 'George Burdell, GA Tech' in winners:
    print("I found George in the list!")
```

Step 2: Flagging the error

- Use the `enumerate` function to loop through the `winners list`. `enumerate` returns both the index `i` (starting from 1 as specified by start=1) and the value `winner` from the list.
    - In each iteration of the loop, enumerate provides a **tuple** containing two elements: the first element is the index i (which starts at 1 in this case), and the second element is the value winner from the winners list.
        - For example:
            - i is 1 (as specified by start=1)
            - winner is 'Geraint Thomas, Great Britain'

Python

```python
# Printing winners with placing, flagging George's name
for i, winner in enumerate(winners, start=1):
    if 'George Burdell' in winner:
        print(f'{i}. {winner} - Potential Error')
    else:
        print(f'{i}. {winner}')
```

Step 3: Correcting the List

- use the **`remove`** method to delete 'George Burdell, GA Tech' from the winners list
  - use the **`append`** method to add 'Julian Alaphilippe, France' to the end of the winners list.

Python

```
# Removing George's information from the list
winners.remove('George Burdell, GA Tech')

# Adding Julian Alaphilippe to the list
winners.append('Julian Alaphilippe, France')
```

Step 4: Updated list of countries

- Create a **list comprehension** to extract the country name from each winner in the winners list, by splitting the string at `', '` and taking the second element `([1])`. This generates a list of countries.
- Convert this list to a set to remove any duplicates, then convert it back to a list which is sorted alphabetically using the sorted function. -This new sorted list is assigned to the variable countries

Python

```python
# Extracting and printing ordered list of countries
countries = sorted(set(winner.split(', ')[1] for winner in winners))
for country in countries:
    print(country)
```

Extracting specific sub-strings using Python

- Extracting specific sub-strings `"2399"` from a given text below.
- Description of the `re.findall()` method.

Input

```
# Assuming myText is a list of strings where each string is a line from your text file
  myText = ["Some text with Text Node 2399 and other text",
  "Another line of text", " ... "]
```

Step 1

- A variable named `myText` is initialized with a list of strings.

Python

```python
import re  # Import the regular expression module
# Assuming myText is a list of strings where each string is a line from your text file
myText = ["Some text with Text Node 2399 and other text", "Another line of text", " ...
```

Step 2

- An empty list named **`listValues`** is initialized to store the extracted substrings.

Python

```
listValues = []  # Initialize an empty list to store the extracted sub-strings
```

Step 3

- A for loop begins, iterating through each string (referred to as line) in the myText list

Python

```
# Loop through each line in the text
for line in myText:
    line = line.rstrip()  # Remove trailing whitespace
    newVal = re.findall('2399', line)  # Find "2399" in the line
    if newVal:
        listValues.append(newVal[0])  # Append "2399" to listValues

print(listValues)
```

Step 4

- The rstrip() method is called on line to remove any trailing whitespace characters (spaces, tabs, newline characters).

Python

```
# Loop through each line in the text
for line in myText:
line = line.rstrip()  # Remove trailing whitespace
    newVal = re.findall('2399', line)  # Find "2399" in the line
    if newVal:
        listValues.append(newVal[0])  # Append "2399" to listValues

print(listValues)
```

Step 5

- The re.findall() function is called with the arguments '2399' and line. This function searches line for all occurrences of the substring '2399', returning a list of all the matches.
- This list is assigned to the variable newVal.

Python

```
# Loop through each line in the text
for line in myText:
   line = line.rstrip()  # Remove trailing whitespace
newVal = re.findall('2399', line)  # Find "2399" in the line
    if newVal:
        listValues.append(newVal[0])  # Append "2399" to listValues
print(listValues)
```

Step 6

- An if statement checks whether newVal is non-empty, i.e., if '2399' was found in line.
- If newVal is non-empty, the append() method is called on listValues to add the first item in newVal (which will be '2399') to listValues.

Python

```
# Loop through each line in the text
for line in myText:
   line = line.rstrip()  # Remove trailing whitespace
  newVal = re.findall('2399', line)  # Find "2399" in the line
if newVal:
  listValues.append(newVal[0])  # Append "2399" to listValues
print(listValues)
```

Exercises

- Please click on the link provided below.
  - Built-In Data Structures:Dictionaries
  - In-Class Exercise