Files and Loops: Takeaways

Syntax

COMMON FILE OPERATIONS

- Opening files:

```
f = open("file.txt", "r")
```

- Reading a file into a string:

```
data = f.read()
```

- Splitting the string representing file contents into a list:

```
list_elements = data.split("\n")
```

BASICS OF LOOPS

- Iterate over a list:

```
sample_list = [0, 1, 2]

for element in sample_list:

    print(element)
```

- Iterate over a list of lists:

```
lol = [

    ["China", 1384688986],

    ["India", 1296834042],

    ["United States", 329256465]

]

top_three_population = 0

for row in lol:

    top_three_sum = top_three_sum + row[1]
```

- Converting rows from a CSV file to different data types:

```
f = open('crime_rates.csv', 'r')
data = f.read()
rows = data.split('\n')
for row in rows:
    values = row.split(',')
    crime_rate = int(values[1])
    int_crime_rates.append(crime_rate)
```

Concepts

- To work with a data set stored as a file in Python, you need to convert it to the right representation:
    - Open the file using `open()` to return a File object, which stores the filename, how to open the file, etc.: `open("file.txt", "r")`
    - Use `file.read()` to read the contents of the file into a string: `str_data = open("file.txt", "r")`
    - Use the `str.split()` method to split the string into a list of strings, on the newline delimiter: `list_results = open("file.txt", "r").read().split("\n")`
    - Finally, use the `str.split()` method for each string (row) in the list we created in a loop into a list of lists (also called a nested list):
- Combined code example:

```
final_data = []
for row in list_results:
    split_list = row.split(',')
    final_data.append(split_list)
```

- To access elements in a nested list, you need to use a pair of brackets for each level of nesting:

```
one_level = [0,1,2]

first_element = one_level[0]

two_levels = [[0,1,2], [3,4,5]]

nested_element = two_levels[1][0]
```

Resources

- Python Documentation: Reading and Writing files
- Python Documentation: Control Flow tools