Assignment 3 - Strings and Collection Types

Instructions

For this assignment we will be working with the `avengers.csv` dataset used for the Joining The Avengers Is As Deadly As Jumping Off A Four-Story Building (fivethirtyeight.com/features/avengers-death-comics-age-of-ultron). You can find the original data in the fivethirtyeight/data (https://github.com/fivethirtyeight/data/tree/master/avengers) Github repository. Copy this data into the same directory as your assignment notebook.

Follow the instructions for submitting a Jupyter Notebook assignment in the submitting assignments documentation.

1. Reading data from a file (5 points)

The `avengers.csv` is encoded with ISO-8859-1 character encoding. This is common in CSV files generated by Microsoft Excel.

Try opening the the file `avengers.csv` and reading the data. What error message do you receive? What does it mean?

In [ ]:
```
#Often times when you create csv files (especially with older versions of Excel)
#of encoding called iso-8859-1.  When you attempt to open the file in Python you
#
# So how exactly do we open a file in the first place.
# There are many ways to open files for reading.  Below is an example:

with open('avengers.csv') as f:
    lines = f.readlines()

#When you attempt to open an iso-8859-1 file you get the below error message:
#
#---------------------------------------------------------------------------
#UnicodeDecodeError                        Traceback (most recent call last)
#<ipython-input-1-72d9eb4ffa4e> in <module>()
#      1 # Approach 1
#      2 with open('avengers.csv') as f:
#----> 3     lines = f.readlines()
#
#/usr/local/var/pyenv/versions/3.6.1/lib/python3.6/codecs.py in decode(self, inpu
#    319         # decode input (taking the buffer into account)
#    320         data = self.buffer + input
#--> 321         (result, consumed) = self._buffer_decode(data, self.errors, fina
#    322         # keep undecoded input until the next call
#    323         self.buffer = data[consumed:]
#
#UnicodeDecodeError: 'utf-8' codec can't decode byte 0xe6 in position 5764: inval
#
```

2. Changing file encodings (5 points)

In order to work with the data in `avengers.csv`, we need to change its character encoding to `utf-8`.

Open the `avengers.csv` as a binary file, decode the binary data from `ISO-8859-1`, and write the `utf-8` encoded data to `avengers_utf.csv`.

In [ ]:
```
#So how do we fix this?  We need to convert the file to UTF-8 character encoding (

#Open the file in read/binary mode
with open('avengers.csv', 'rb') as f:
    #Read the file
    data = f.read()
#decode the iso-8859-1 file and save to a variable named decoded_data
decoded_data = data.decode('iso-8859-1')
#encode the previously decoded data as utf8 and save to a variable called encoded_
encoded_data = decoded_data.encode('utf8')

#open a new file called avengers_utf8.csv in write/binary
with open('avengers_utf8.csv', 'wb') as f:
    #write our encoded data to the avengers_utf8.csv file
    f.write(encoded_data)
```

3. Read and count the lines (3 points)

Open the `resouces/avengers_utf8.csv` data and read the lines from the file. Assign the lines to the variable `lines`.

How many lines does the file contain?

In [38]:
```
# Note: since avengers.csv was already correctly formated in zip file it was copi(

with open('avengers_utf8.csv') as file_in:
    # read lines
    lines = file_in.readlines()
    # clean up new line char
    #lines = [line.rstrip('\n') for line in lines]
    # print length
    print(len(lines))
```

```
174
```

4. Parse the header row (12 Points)

The first row of the CSV file is the header row.

1. Using list slicing, assign the header row to the variable `header_row`.
2. CSV files use commas to separate fields. Create a list of header fields from the `header_row` variable using the string `split` method.

3. Using list slicing, make a list that contains last two fields in the header.
4. Using list slicing, make a list that contains the 3rd through 6th elements (Hint: Remember that Python uses zero-based indexing).

In [39]:
```
# 1. assign header_row to the first row of avengers file
header_row = lines[0]
print(header_row)
```

```
URL,Name/Alias,Appearances,Current?,Gender,Probationary Introl,Full/Reserve Ave
ngers Intro,Year,Years since joining,Honorary,Death1,Return1,Death2,Return2,Dea
th3,Return3,Death4,Return4,Death5,Return5,Notes
```

In [40]:
```
#2. create a list of header fields from header_row using split
fields = header_row.split(',')
print(fields)
```

```
['URL', 'Name/Alias', 'Appearances', 'Current?', 'Gender', 'Probationary Intro
l', 'Full/Reserve Avengers Intro', 'Year', 'Years since joining', 'Honorary',
'Death1', 'Return1', 'Death2', 'Return2', 'Death3', 'Return3', 'Death4', 'Retur
n4', 'Death5', 'Return5', 'Notes\n']
```

In [41]:
```
# 3. Using list slicing, make a list that contains last two fields in the header.
last_two_fields = fields[-2:]
print(last_two_fields)
```

```
['Return5', 'Notes\n']
```

In [42]:
```
# 4. Using list slicing, make a list that contains the 3rd through 6th elements
# # (Hint: Remember that Python uses zero-based indexing)
other_elements = fields[2:7:3]
print(other_elements)
```

```
['Appearances', 'Probationary Introl']
```

5. Working with Tuples (6 Points)

Using the `header` variable created in the previous problem, we are going to create a tuple from that header and assign it to the variable `header_tuple`. We also create a copy of the original header and assign it to `header_copy`.

```
import copy

header_tuple = tuple(header)
header_copy = copy.copy(header)
```

1. In the original header, the last value in the list has an extra newline. Strip the newline character from the last header field and reassign..
2. In the `header_copy` list, append the value `More info` to the end of the list. Verify the value has been added.

3. Change the value of the first item in the header list from `URL` to `url`. Verify the value has been changed.
4. Try steps 1 and 2 on `header_tuple` instead of `header`. What happens? Why can we change `header`, but not `header_tuple`?

In [43]:
```
# SET UP
import copy

# fields (list) is the list from header_row (string line)
# assign header to fields for tuple
header = fields

header_tuple = tuple(header)
header_copy = copy.copy(header)
```

In [44]:
```
# 1. In the original header, the last value in the list has an extra newline.
# Strip the newline character from the last header field and reassign.
header = header_row.rsplit()    # rsplit works on strings so I had to go back to 
print(header_row)
print(header)
```

```
URL,Name/Alias,Appearances,Current?,Gender,Probationary Introl,Full/Reserve Ave
ngers Intro,Year,Years since joining,Honorary,Death1,Return1,Death2,Return2,Dea
th3,Return3,Death4,Return4,Death5,Return5,Notes

['URL,Name/Alias,Appearances,Current?,Gender,Probationary', 'Introl,Full/Reserv
e', 'Avengers', 'Intro,Year,Years', 'since', 'joining,Honorary,Death1,Return1,D
eath2,Return2,Death3,Return3,Death4,Return4,Death5,Return5,Notes']
```

In [45]:
```
# 2. In the header_copy list, append the value More info to the end of the list.
# Verify the value has been added.
header_copy.append("More info")
print(header_copy)
```

```
['URL', 'Name/Alias', 'Appearances', 'Current?', 'Gender', 'Probationary Intro
l', 'Full/Reserve Avengers Intro', 'Year', 'Years since joining', 'Honorary',
'Death1', 'Return1', 'Death2', 'Return2', 'Death3', 'Return3', 'Death4', 'Retur
n4', 'Death5', 'Return5', 'Notes\n', 'More info']
```

In [46]:
```
# 3.Change the value of the first item in the header list from URL to url.
# Verify the value has been changed
header[0] = "url"
print(header[0])
print(header_copy[0])
```

```
url
URL
```

In [47]:
```
# 4. Try steps 1 and 2 on header_tuple instead of header.
# What happens? Why can we change header, but not header_tuple?
# tuples a immutable - you can't change them once they are set
```

6. Parsing Row Data (9 points)

From the `lines` variable you set in part 3, set the sixth line to the variable `line`. This should be the entry for *Thor Odinson*. From this entry, we will create a dictionary with information from this line in the CSV file.

1. Create a dictionary record

Given the field definitions defined below, create a dictionary called `record` from the string data in the `line` variable. The record should have keys corresponding to the `name` variable and type corresponding with the `type` variable. The `index` variable gives the index position of field value when the comma separated line is parsed into a list.

```
fields = [
    {'index': 0, 'name': 'url', 'type': str},
    {'index': 1, 'name': 'name_alias', 'type': str},
    {'index': 2, 'name': 'appearances', 'type': int},
    {'index': 3, 'name': 'is_current', 'type': bool},
    {'index': 4, 'name': 'gender', 'type': str},
    {'index': 7, 'name': 'year', 'type': int},
    {'index': 8, 'name': 'years_since_joining', 'type': int},
]
```

2. Since this is from an older dataset, the `years_since_joining` value is no longer accurate. Update the `years_since_joining` using the current year.

3. Using the `name_alias` field, add two new names to the record called `first_name` and `last_name`.

In [54]:
```
# 1. create dictionary record

# get the 6th row data entry
line = lines[5].rstrip()
line = line.split(',')

#create record dict
record = {fields[0]: str(line[0]),
          fields[1]: str(line[1]),
          fields[2]: int(line[2]),
          fields[3]: bool(line[3]),
          fields[4]: str(line[4]),
          fields[7]: int(line[7]),
          fields[8]: int(line[8]),
          }

print(record)
```

```
{'URL': 'http://marvel.wikia.com/Thor_Odinson_(Earth-616)', 'Name/Alias': 'Thor
Odinson', 'Appearances': 2402, 'Current?': True, 'Gender': 'MALE', 'Year': 196
3, 'Years since joining': 52}
```

In [55]:
```
# 2. update years_since_joining
num = 2018 - int(line[7])
years = {fields[8]: num}
record.update(years)
print(record)
```

```
{'URL': 'http://marvel.wikia.com/Thor_Odinson_(Earth-616)', 'Name/Alias': 'Thor
Odinson', 'Appearances': 2402, 'Current?': True, 'Gender': 'MALE', 'Year': 196
3, 'Years since joining': 55}
```

In [56]:
```
# 3. using the name/alias add two new names to the record
names = {'first_name': 'Thor', 'last_name': 'Odinson'}
name_update = {fields[1]: names}
record[fields[1]] = name_update
print(record)
```

```
{'URL': 'http://marvel.wikia.com/Thor_Odinson_(Earth-616)', 'Name/Alias': {'Nam
e/Alias': {'first_name': 'Thor', 'last_name': 'Odinson'}}, 'Appearances': 2402,
'Current?': True, 'Gender': 'MALE', 'Year': 1963, 'Years since joining': 55}
```

In [ ]: