In [1]:
```
animals = ['lion', 'tiger', 'crocodile', 'vulture', 'hippo']
print(animals)
```

```
['lion', 'tiger', 'crocodile', 'vulture', 'hippo']
```

In [2]:
```
for creature in animals:
    print(creature)
```

```
lion
tiger
crocodile
vulture
hippo
```

In [3]:
```
animals = ['lion', 'tiger', 'crocodile', 'vulture', 'hippo']
for creature in animals:
    pass
```

In [4]:
```
print('The loop variable is now: ' + creature)
```

```
The loop variable is now: hippo
```

In [5]:
```
animals = ['lion', 'tiger', 'crocodile', 'vulture', 'hippo']
for creature in animals:
```

```
  File "<ipython-input-5-3e5445e46d23>", line 2
    for creature in animals:
                           ^
SyntaxError: unexpected EOF while parsing
```

In [6]:
```
for creature in animals:
    print(creature + ',', end='')
```

```
lion,tiger,crocodile,vulture,hippo,
```

In [7]:
```
', '.join(animals)
```

Out[7]:
```
'lion, tiger, crocodile, vulture, hippo'
```

In [9]:
```
import os

os.mkdir('yearly_files')
```

In [12]:
```
os.listdir('./') #文件都保存在根目录
```

Out[12]:
```
['.conda',
 '.ipynb_checkpoints',
 '.ipython',
 '.matplotlib',
 '.python_history',
 '03startingwithdata.ipynb',
 '04IndexingSlicingandSubsettingDataFrameinPython.ipynb',
 '05DataTypesandFormats.ipynb',
 '06CombiningDataFrameswithPandas.ipynb',
 '07DataWorkflowsandAutomation.ipynb',
 '3D Objects',
 'AppData',
 'Application Data',
 'Contacts',
 'Cookies',
 'Desktop',
 'Documents',
 'Downloads',
 'Favorites',
 'IntelGraphicsProfiles',
 'Links',
 'Local Settings',
 'Music',
 'My Documents',
 'NetHood',
 'NTUSER.DAT',
 'ntuser.dat.LOG1',
 'ntuser.dat.LOG2',
 'NTUSER.DAT{fd9a35db-49fe-11e9-aa2c-248a07783950}.TM.blf',
 'NTUSER.DAT{fd9a35db-49fe-11e9-aa2c-248a07783950}.TMContainer00000000000000000001.r
egtrans-ms',
 'NTUSER.DAT{fd9a35db-49fe-11e9-aa2c-248a07783950}.TMContainer00000000000000000002.r
egtrans-ms',
 'ntuser.ini',
 'OneDrive',
 'out.csv',
 'Pictures',
 'plots.csv',
 'PrintHood',
 'Recent',
 'Saved Games',
 'Searches',
 'SendTo',
 'species.csv',
 'surveys.csv',
 'surveys_complete.csv',
 'Templates',
 'Untitled Folder',
 'Videos',
 'yearly_files',
 '「开始」菜单']
```

In [13]:
```
import pandas as pd

# Load the data into a DataFrame
surveys_df = pd.read_csv('surveys.csv')
```

In [14]:
```
# Select only data for the year 2002
surveys2002 = surveys_df[surveys_df.year == 2002]
```

In [15]:
```
# Write the new DataFrame to a CSV file
surveys2002.to_csv('yearly_files/surveys2002.csv')
```

In [16]:
```
surveys_df['year']
```

Out[16]:

0	1977
1	1977
2	1977
3	1977
4	1977
5	1977
6	1977
7	1977
8	1977
9	1977
10	1977
11	1977
12	1977
13	1977
14	1977
15	1977
16	1977
17	1977
18	1977
19	1977
20	1977
21	1977
22	1977
23	1977
24	1977
25	1977
26	1977
27	1977
28	1977
29	1977
...	...
35519	2002
35520	2002
35521	2002
35522	2002
35523	2002
35524	2002
35525	2002
35526	2002
35527	2002
35528	2002
35529	2002
35530	2002
35531	2002
35532	2002
35533	2002
35534	2002
35535	2002
35536	2002
35537	2002
35538	2002
35539	2002
35540	2002
35541	2002
35542	2002
35543	2002
35544	2002
35545	2002
35546	2002
35547	2002

```
35548    2002
Name: year, Length: 35549, dtype: int64
```

In [17]:
```
surveys_df['year'].unique()
```

Out[17]:
```
array([1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987,
       1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998,
       1999, 2000, 2001, 2002], dtype=int64)
```

In [18]:
```
for year in surveys_df['year'].unique():
   filename='data/yearly_files/surveys' + str(year) + '.csv'
   print(filename)
```

```
data/yearly_files/surveys1977.csv
data/yearly_files/surveys1978.csv
data/yearly_files/surveys1979.csv
data/yearly_files/surveys1980.csv
data/yearly_files/surveys1981.csv
data/yearly_files/surveys1982.csv
data/yearly_files/surveys1983.csv
data/yearly_files/surveys1984.csv
data/yearly_files/surveys1985.csv
data/yearly_files/surveys1986.csv
data/yearly_files/surveys1987.csv
data/yearly_files/surveys1988.csv
data/yearly_files/surveys1989.csv
data/yearly_files/surveys1990.csv
data/yearly_files/surveys1991.csv
data/yearly_files/surveys1992.csv
data/yearly_files/surveys1993.csv
data/yearly_files/surveys1994.csv
data/yearly_files/surveys1995.csv
data/yearly_files/surveys1996.csv
data/yearly_files/surveys1997.csv
data/yearly_files/surveys1998.csv
data/yearly_files/surveys1999.csv
data/yearly_files/surveys2000.csv
data/yearly_files/surveys2001.csv
data/yearly_files/surveys2002.csv
```

In [21]:
```
# Load the data into a DataFrame
surveys_df = pd.read_csv('surveys.csv')

for year in surveys_df['year'].unique():

    # Select data for the year
    surveys_year = surveys_df[surveys_df.year == year]

    # Write the new DataFrame to a CSV file
    filename = 'yearly_files/surveys' + str(year) + '.csv'
    surveys_year.to_csv(filename)
```

In [22]:
```
filename = 'yearly_files/surveys' + str(year) + '.csv'
```

In [23]:
```
surveys_year = surveys_df[surveys_df.year == year].dropna()
```

In [24]:
```
n_year = 5  # better overview by making variable from it
first_year = surveys_df['year'].min()
last_year = surveys_df['year'].max()

for year in range(first_year, last_year, n_year):
    print(year)

    # Select data for the year
    surveys_year = surveys_df[surveys_df.year == year].dropna()
```

```
1977
1982
1987
1992
1997
```

In [27]:
```
for species in surveys_df['species_id'].dropna().unique():
    surveys_species = surveys_df[surveys_df.species_id == species]
    filename = 'surveys' + species + '.csv'
    surveys_species.to_csv(filename)
```

In [28]:
```
def this_is_the_function_name(input_argument1, input_argument2):
    # The body of the function is indented
    # This function prints the two arguments to screen
    print('The function arguments are:', input_argument1, input_argument2, '(this is do

    # And returns their product
    return input_argument1 * input_argument2
```

In [29]:
```
product_of_inputs = this_is_the_function_name(2, 5)
```

```
The function arguments are: 2 5 (this is done inside the function!)
```

In [30]:
```
print('Their product is:', product_of_inputs, '(this is done outside the function!)')
```

```
Their product is: 10 (this is done outside the function!)
```

In [31]:
```
product_of_inputs = this_is_the_function_name(3, 'clap')
print(product_of_inputs)
```

```
The function arguments are: 3 clap (this is done inside the function!)
clapclapclap
```

In [32]:
```
product_of_inputs = this_is_the_function_name(3, 18, 33)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-32-fe6c4bc29afc> in <module>()
----> 1 product_of_inputs = this_is_the_function_name(3, 18, 33)

TypeError: this_is_the_function_name() takes 2 positional arguments but 3 were give
n
```

In [33]:
```
this_is_the_function_name(23, 108)
```

```
The function arguments are: 23 108 (this is done inside the function!)
```

Out[33]: 2484

In [35]:
```
var_defined_outside = 'outside'
def this_is_the_function_name(input_argument1, input_argument2):
    var_defined_inside = 'inside'
    print('The function arguments are:', input_argument1, input_argument2, '(this is do
    print('This variable was created ' + var_defined_outside + ' the function')
    return input_argument1 * input_argument2
```

In [36]:
```
this_is_the_function_name(3.3, 7.9)
```

```
The function arguments are: 3.3 7.9 (this is done inside the function!)
This variable was created outside the function
```

Out[36]: 26.07

In [38]:
```
shared_variable_name = 'who would I be'

def this_is_the_function_name(input_argument1, input_argument2):
    shared_variable_name = 'without you'
    print('The function arguments are:', input_argument1, input_argument2, '(this is do
    return input_argument1 * input_argument2

print(shared_variable_name)
```

```
who would I be
```

In [39]:
```
this_is_the_function_name(2, 3) # does calling the function change the variable's value
```

```
The function arguments are: 2 3 (this is done inside the function!)
```

Out[39]: 6

In [40]:
```
print(shared_variable_name)
```

```
who would I be
```

In [41]:
```
def this_is_the_function_name(input_argument1, input_argument2):
    shared_variable_name = 'without you'
    shared_variable_name = shared_variable_name + ', without them?'
    print(shared_variable_name)
    print('The function arguments are:', input_argument1, input_argument2, '(this is do
    return input_argument1 * input_argument2

this_is_the_function_name(2, 3)
```

```
without you, without them?
The function arguments are: 2 3 (this is done inside the function!)
```

Out[41]: 6

In [42]:
```
print(shared_variable_name)
```

```
who would I be
```

In [44]:
```
help(one_year_csv_writer)
```

```
Help on function one_year_csv_writer in module __main__:

one_year_csv_writer(this_year, all_data)
    Writes a csv file for data from a given year.

    this_year -- year for which data is extracted
    all_data -- DataFrame with multi-year data
```

In [52]:
```
a = 5

if a<0:  # Meets first condition?

    # if a IS less than zero
    print('a is a negative number')

elif a>0:  # Did not meet first condition. meets second condition?

    # if a ISN'T less than zero and IS more than zero
    print('a is a positive number')

else:  # Met neither condition

    # if a ISN'T less than zero and ISN'T more than zero
    print('a must be zero!')
```

```
a is a positive number
```

In [55]:
```
def yearly_data_arg_test(all_data, start_year=None, end_year=None):
    """
    Modified from yearly_data_csv_writer to test default argument values!

    all_data -- DataFrame with multi-year data
    start_year -- the first year of data we want, Check all_data! (default None)
    end_year -- the last year of data we want; Check all_data! (default None)
    """

    if start_year is None:
        start_year = min(all_data.year)
    if end_year is None:
        end_year = max(all_data.year)

    return start_year, end_year
```

In [56]:
```
start, end = yearly_data_arg_test(surveys_df, 1988, 1993)
print('No keywords:\t\t\t', start, end)
```

```
No keywords:                    1988 1993
```

In [57]:
```
start, end = yearly_data_arg_test(surveys_df, start_year=1988, end_year=1993)
print('Both keywords, in order:\t', start, end)
```

```
Both keywords, in order:        1988 1993
```

In [58]:
```
start, end = yearly_data_arg_test(surveys_df, end_year=1993, start_year=1988)
print('Both keywords, flipped:\t\t', start, end)
```

```
Both keywords, flipped:		1988 1993
```

In [59]:
```
start, end = yearly_data_arg_test(surveys_df, start_year=1988)
print('One keyword, default end:\t', start, end)
```

```
One keyword, default end:	1988 2002
```

In [60]:
```
start, end = yearly_data_arg_test(surveys_df, end_year=1993)
print('One keyword, default start:\t', start, end)
```

```
One keyword, default start:	1977 1993
```

In [62]:
```
def one_year_csv_writer(this_year, all_data):
    """
    Writes a csv file for data from a given year.

    this_year -- year for which data is extracted
    all_data -- DataFrame with multi-year data
    """

    # Select data for the year
    surveys_year = all_data[all_data.year == this_year]

    # Write the new DataFrame to a csv file
    filename = 'yearly_files/function_surveys' + str(this_year) + '.csv'
    surveys_year.to_csv(filename)
    return filename

def yearly_data_csv_writer(start_year, end_year, all_data):
    """
    Writes separate CSV files for each year of data.

    start_year -- the first year of data we want
    end_year -- the last year of data we want
    all_data -- DataFrame with multi-year data
    """

    # "end_year" is the last year of data we want to pull, so we loop to end_year+1
    output_files = []
    for year in range(start_year, end_year+1):
        output_files.append(one_year_csv_writer(year, all_data))
    return output_files

print(yearly_data_csv_writer(2000, 2001, surveys_df))
```

```
['yearly_files/function_surveys2000.csv', 'yearly_files/function_surveys2001.csv']
```

In [64]:
```
def one_year_csv_writer(this_year, all_data, folder_to_save, root_name):
    """
    Writes a csv file for data from a given year.

    Parameters
    ---------
    this_year : int
        year for which data is extracted
    all_data: pd.DataFrame
        DataFrame with multi-year data
    folder_to_save : str
        folder to save the data files
    root_name: str
        root of the filenames to save the data
    """

    # Select data for the year
    surveys_year = all_data[all_data.year == this_year]

    # Write the new DataFrame to a csv file
    filename = os.path.join(folder_to_save, ''.join([root_name, str(this_year), '.csv']
    surveys_year.to_csv(filename)
```

In [65]:
```
directory_name = './'
```

In [66]:
```
root_file_name = '123'
```

In [67]:
```
for year in range(start_year, end_year+1):
     one_year_csv_writer(year, surveys_df, directory_name, root_file_name)
```

In [ ]:
```

```