Python Bootcamp 3 Part 2

This is a Jupyter Notebook. To run a gray code cell, click in the cell and either click on the "Run" arrow, or type shift+enter (or shift+return on a Mac).

Importing modules

Later in this notebook we are going to be using the `mean()` function again, from the `statistics` module. We have learned that we can import the package like this:

```
import statistics

statistics.mean([6, 4, 2, 7, 6])
```

```
5
```

This can sometimes make our function names long, like `statistics.variance()`.

We can also import modules with a shortened nickname so that we don't have to type out the full module name every time we use a function:

```
import statistics as stats

stats.mean([6, 4, 2, 7, 6])
```

```
5
```

OR, if we know that we are only going to use one or two functions from a module, we can import only those functions. When we do this, we do not have to include the module name when calling the function:

```
from statistics import mean

mean([6, 4, 2, 7, 6])
```

```
5
```

```
from statistics import mean, mode

mean([6, 4, 2, 7, 6])
```

```
5
```

```
mode([6, 4, 2, 7, 6])
```

```
6
```

A very quick lesson on installing modules

You can install and update Python modules onto your computer using the command line, but you can also do it from inside a Jupyter Notebook. If you use `!` directly before a command in a Jupyter Notebook, it tells the computer that you are going to be speaking to the computer in your command line language instead of Python. We will practice by installing Pandas, which we will be using tomorrow, and making sure the Statistics package is upgraded.

Make sure you are connected to the internet before running `!pip`.

```
%pip install pandas #panel data

Note: you may need to restart the kernel to use updated packages.

ERROR: Invalid requirement: '#panel': Expected package name at the
start of dependency specifier
    #panel
    ^

%pip install statistics --upgrade

Collecting statistics
  Downloading statistics-1.0.3.5.tar.gz (8.3 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: docutils>=0.3 in c:\users\isaac\
anaconda3\lib\site-packages (from statistics) (0.18.1)
Building wheels for collected packages: statistics
  Building wheel for statistics (setup.py): started
  Building wheel for statistics (setup.py): finished with status
'done'
  Created wheel for statistics: filename=statistics-1.0.3.5-py3-none-
any.whl size=7450
sha256=d22958bee52082753df501db592f3b32c9036b44c869c0a3868d3cfe543daa5
d
  Stored in directory: c:\users\isaac\appdata\local\pip\cache\wheels\
10\9c\1a\0c68a89e4533a18e9f9902018b94ffc2481139818b67cbb05a
Successfully built statistics
Installing collected packages: statistics
Successfully installed statistics-1.0.3.5
Note: you may need to restart the kernel to use updated packages.
```

Dictionaries

The key must be of a type that is immutable. For example, integer, float, string, or Boolean. The value can be any object.

```
grade_dict = {"Charlie": [90, 96, 89, 79],
              "Tony": [99, 98, 96, 93],
              "Suman": [85, 88, 83, 87],
```

```
            "Yuvie": [66, 76, 80, 62],
            "May": [97, 94, 89, 91]}

print(grade_dict)
```

```
{'Charlie': [90, 96, 89, 79], 'Tony': [99, 98, 96, 93], 'Suman': [85,
88, 83, 87], 'Yuvie': [66, 76, 80, 62], 'May': [97, 94, 89, 91]}
```

Indexing a dictionary

```
grade_dict["Tony"]
```

```
[99, 98, 96, 93]
```

```
grade_dict["Tony"][-1]
```

```
93
```

Adding an entry to a dictionary

You don't have to use a function to add to a dictionary:

```
grade_dict["Mike"] = [82, 88, 90]

print(grade_dict)
```

```
{'Charlie': [90, 96, 89, 79], 'Tony': [99, 98, 96, 93], 'Suman': [85,
88, 83, 87], 'Yuvie': [66, 76, 80, 62], 'May': [97, 94, 89, 91],
'Mike': [82, 88, 90]}
```

Looping through a dictionary

```
for entry in grade_dict:
    print(entry)
```

```
Charlie
Tony
Suman
Yuvie
May
Mike
```

We can be more explicit to tell the computer that we only want to loop through the keys:

```
for key in grade_dict.keys():
    print(key)
```

```
Charlie
Tony
Suman
Yuvie
```

```
May
Mike
```

Or we can loop through the values:

```
for value in grade_dict.values():
    print(value)
```

```
[90, 96, 89, 79]
[99, 98, 96, 93]
[85, 88, 83, 87]
[66, 76, 80, 62]
[97, 94, 89, 91]
[82, 88, 90]
```

Remember that we can give our temporary variable any name we want in our for loop. This is commonly used:

```
for k in grade_dict.keys():
    print(k)
```

```
Charlie
Tony
Suman
Yuvie
May
Mike
```

```
for v in grade_dict.values():
    print(v)
```

```
[90, 96, 89, 79]
[99, 98, 96, 93]
[85, 88, 83, 87]
[66, 76, 80, 62]
[97, 94, 89, 91]
[82, 88, 90]
```

But it's also good to use more appropriate variable names:

```
for student in grade_dict.keys():
    print(student)
```

```
Charlie
Tony
Suman
Yuvie
May
Mike
```

```
for grade_list in grade_dict.values():
    print(grade_list)
```

```
[90, 96, 89, 79]
[99, 98, 96, 93]
[85, 88, 83, 87]
[66, 76, 80, 62]
[97, 94, 89, 91]
[82, 88, 90]
```

We can also loop through both the keys and values:

```
for k, v in grade_dict.items():
    print(k)
    print(v)
```

```
Charlie
[90, 96, 89, 79]
Tony
[99, 98, 96, 93]
Suman
[85, 88, 83, 87]
Yuvie
[66, 76, 80, 62]
May
[97, 94, 89, 91]
Mike
[82, 88, 90]
```

```
for student, grade_list in grade_dict.items():
    print(student)
    print(grade_list)
```

```
Charlie
[90, 96, 89, 79]
Tony
[99, 98, 96, 93]
Suman
[85, 88, 83, 87]
Yuvie
[66, 76, 80, 62]
May
[97, 94, 89, 91]
Mike
[82, 88, 90]
```

If you want to print each student name and grades in one line:

```
for student, grade_list in grade_dict.items():
    print(student, grade_list)
```

```
Charlie [90, 96, 89, 79]
Tony [99, 98, 96, 93]
Suman [85, 88, 83, 87]
Yuvie [66, 76, 80, 62]
May [97, 94, 89, 91]
Mike [82, 88, 90]
```

Since our values are list objects, we can also loop through the lists:

```
for student, grade_list in grade_dict.items():
    print(student)
    for grade in grade_list:
        print(grade)
```

```
Charlie
90
96
89
79
Tony
99
98
96
93
Suman
85
88
83
87
Yuvie
66
76
80
62
May
97
94
89
91
Mike
82
88
90
```

```
from statistics import mean
for student, grade_list in grade_dict.items():
    print(student, mean(grade_list))
    # print(mean(grade_list))
```

```
Charlie 88.5
Tony 96.5
Suman 85.75
Yuvie 71
May 92.75
Mike 86.66666666666667
```

That code is called a **nested loop** - a loop inside a loop!

Adding key:value pairs to an empty dictionary

Here we will create a new dictionary from the data in the grades_dict. The keys will be the student's names and the values will be their final score for the class. The final score will be calculated as the mean of all the scores in their grade list.

First, we create an empty dictionary:

```
final_dict = {}

final_dict
```

```
{}
```

Next, we loop through the old dictionary, calculate each person's final grade, and add them to the new dictionary:

```
for student, grade_list in grade_dict.items():
    final_score = mean(grade_list)
    final_dict[student] = final_score

print(final_dict)
```

```
{'Charlie': 88.5, 'Tony': 96.5, 'Suman': 85.75, 'Yuvie': 71, 'May': 92.75, 'Mike': 86.66666666666667}
```

Working with messy data

If you remember, one of our students, "Mike", only had 3 grades entered, while everyone else had 4. That's something we might want to know when we're calculating final grades. Let's add an if/else statement to our code:

```
final_dict = {}
for student, grade_list in grade_dict.items():
    if len(grade_list) >= 4:
        final_score = mean(grade_list)
        final_dict[student] = final_score
    else:
        print(student + " is missing grades.")
print(final_dict)
```

```
Mike is missing grades.
{'Charlie': 88.5, 'Tony': 96.5, 'Suman': 85.75, 'Yuvie': 71, 'May':
92.75}
```

This code is ok, but it contains that number `4` for the length of the list. Let's say you teach the same class next year and you want to reuse the code, only next year you give 5 tests instead of 4.

When there are details in the code specific to your data, we say they are **hard coded**. As a beginner, it is likely that you will do a lot of hard coding to solve your problems, but if you ever want to reuse your scripts or share them with someone else, you will need to try to **not** hard code.

First, let's change our grade dictionary to reflect Mike's missing grade. The (original) grade dictionary looks like this:

```
grade_dict
```

```
{'Charlie': [90, 96, 89, 79],
 'Tony': [99, 98, 96, 93],
 'Suman': [85, 88, 83, 87],
 'Yuvie': [66, 76, 80, 62],
 'May': [97, 94, 89, 91],
 'Mike': [82, 88, 90]}
```

Mike's value is:

```
grade_dict["Mike"]
```

```
[82, 88, 90]
```

We can reflect Mike's missing grade by adding another data point to Mike's list:

```
grade_dict["Mike"].append("Missed")
print(grade_dict)
```

```
{'Charlie': [90, 96, 89, 79], 'Tony': [99, 98, 96, 93], 'Suman': [85,
88, 83, 87], 'Yuvie': [66, 76, 80, 62], 'May': [97, 94, 89, 91],
'Mike': [82, 88, 90, 'Missed']}
```

Now we will remove the hard coding and instead handle the missing data through a try/except statement. First, let's run the previous code we wrote, but with our altered grade_dict:

```
final_dict = {}
for student, grade_list in grade_dict.items():
    if len(grade_list) >= 4:
        final_score = mean(grade_list)
        final_dict[student] = final_score
    else:
```

```
        print(student + " is missing grades.")
print(final_dict)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
File /usr/local/python/3.12.1/lib/python3.12/statistics.py:327, in _exact_ratio(x)
    325 try:
    326     # x may be an Integral ABC.
--> 327     return (x.numerator, x.denominator)
    328 except AttributeError:

AttributeError: 'str' object has no attribute 'numerator'

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
Cell In[37], line 4
      2 for student, grade_list in grade_dict.items():
      3     if len(grade_list) >= 4:
----> 4         final_score = mean(grade_list)
      5         final_dict[student] = final_score
      6     else:

File /usr/local/python/3.12.1/lib/python3.12/statistics.py:484, in mean(data)
    468 def mean(data):
    469     """Return the sample arithmetic mean of data.
    470
    471     >>> mean([1, 2, 3, 4, 4])
   (...)
    482     If ``data`` is empty, StatisticsError will be raised.
    483     """
--> 484     T, total, n = _sum(data)
    485     if n < 1:
    486         raise StatisticsError('mean requires at least one data point')

File /usr/local/python/3.12.1/lib/python3.12/statistics.py:193, in _sum(data)
    191 for typ, values in groupby(data, type):
    192     types_add(typ)
--> 193     for n, d in map(_exact_ratio, values):
    194         count += 1
    195         partials[d] = partials_get(d, 0) + n

File /usr/local/python/3.12.1/lib/python3.12/statistics.py:330, in
```

```
_exact_ratio(x)
    328 except AttributeError:
    329     msg = f"can't convert type '{type(x).__name__}' to
numerator/denominator"
--> 330     raise TypeError(msg)

TypeError: can't convert type 'str' to numerator/denominator
```

The error gets thrown because we added a string, `Missed`, to the grade_list. Python cannot calculate the mean of a list that included a string. Instead of specifying "4" as the number of grades required, we can use a try/except statement from last week that references the error we just saw:

```
final_dict = {}
for student, grade_list in grade_dict.items():
    try:
        final_grade = mean(grade_list)
        final_dict[student] = final_grade
    except:
        print(student + " has missing grades.")

print(final_dict)
```

```
Mike has missing grades.
{'Charlie': 88.5, 'Tony': 96.5, 'Suman': 85.75, 'Yuvie': 71, 'May':
92.75}
```

List of dictionaries

Sometimes it is useful to have a list of dictionaries because that is how your data is best represented. You can index individual data points in the list or dictionaries, and you can loop through both levels.

```
gradebook = [{"name": "Zygon", "HW1": 10, "HW2": 10, "HW3": 10},
             {"name": "Vogon", "HW1": 10, "HW2": 10, "HW3": 10},
             {"name": "Cylon", "HW1": 10, "HW2": 10, "HW3": 10},
             {"name": "Mudokon", "HW1": 7, "HW2": 8, "HW3": 6}]
```

To return an individual dictionary, you use list indexing because each dictionary is an item in the list:

```
gradebook[2]
```

```
{'name': 'Cylon', 'HW1': 10, 'HW2': 10, 'HW3': 10}
```

To return a value in one of the dictionaries, you first index the dictionary, and then index the key in your key:value pair of interest:

```
gradebook[2]["HW1"]
```

```
10
```

Looping through the list:

```
for dictionary in gradebook:
    name = dictionary["name"]
    HW_total = dictionary["HW1"] + dictionary["HW2"] +
dictionary["HW3"]
    print(name + " scored " + str(HW_total) + " points on Homework")
```

```
Zygon scored 30 points on Homework
Vogon scored 30 points on Homework
Cylon scored 30 points on Homework
Mudokon scored 21 points on Homework
```

That code worked well, but it wouldn't work if more than 3 homework assignments were added. Instead, you can loop through the list and then loop through the dictionary. I've included comments in the code to explain what I'm doing:

```
for dictionary in gradebook: #loop through the list
    name = dictionary["name"] #get student's name
    HW_total = 0 # initialize total HW grade
    for k, v in dictionary.items(): #loop through the key:value pairs
        if k != "name": #get every key:value pair except name
            HW_total = HW_total + v #add the value to our HW total
    print(name + " scored " + str(HW_total) + " points on Homework")
```

```
Zygon scored 30 points on Homework
Vogon scored 30 points on Homework
Cylon scored 30 points on Homework
Mudokon scored 21 points on Homework
```

Dictionary of dictionaries

You can also format your data as a dictionary of dictionaries.

```
grade_dict = {"Zygon": {"HW1": 3, "HW2": 2, "HW3": 4},
              "Vogon": {"HW1": 10, "HW2": 10, "HW3": 10},
              "Cylon": {"HW1": 10, "HW2": 10, "HW3": 10},
              "Mudokon": {"HW1": 7, "HW2": 8, "HW3": 6}}

grade_dict["Zygon"]
```

```
{'HW1': 3, 'HW2': 2, 'HW3': 4}
```

```
grade_dict["Zygon"]["HW1"]
```

```
3
```

BACK TO THE SLIDES

Files

First, where are the files we are working with today?

If you are using Jupyter:

The files should be in your working directory - where you are right now. You should see them in the folder in Jupyter.

Reading files

We can first store the names of the files we will be working with as strings:

```
alice_filename = "alice.txt"
dog_filename = "dogs.txt"
```

We will use a with/as statement to open the file. Let's try opening the file "alice.txt" and printing it to see what it looks like. We will use the read mode:

```
with open(alice_filename, "r", encoding="utf-8") as f:
    print(f)
```

```
<_io.TextIOWrapper name='alice.txt' mode='r' encoding='utf-8'>
```

The file object isn't directly readable. We can use a file object method function, `read()`, to change the file object into a string:

```
with open(alice_filename, "r") as f:
    alice_text = f.read()
```

We have now exited the with/as statement, so the file is closed. `alice_text` is stored in memory, but `f` is closed and cannot be accessed again without reopening the file.

```
f.read()
```

```
---------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[4], line 1
----> 1 f.read()
```

```
ValueError: I/O operation on closed file.
```

```
type(alice_text)
```

```
str
```

```
print(alice_text)
```

```
Alice was beginning to get very tired of sitting by her sister on the
bank, and of having nothing to do: once or twice she had peeped into
the book her sister was reading, but it had no pictures or
conversations in it, "and what is the use of a book," thought Alice,
"without pictures or conversations?"

So she was considering in her own mind (as well as she could, for the
hot day made her feel very sleepy and stupid) whether the pleasure of
making a daisy-chain would be worth the trouble of getting up and
picking the daisies, when suddenly a White Rabbit with pink eyes ran
close by her.

There was nothing so very remarkable in that; nor did Alice think it
so very much out of the way to hear the Rabbit say to itself, "Oh
dear! Oh dear! I shall be too late!" (when she thought it over
afterwards, it occurred to her that she ought to have wondered at
this, but at the time it all seemed quite natural); but when the
Rabbit actually took a watch out of its waistcoat-pocket, and looked
at it, and then hurried on, Alice started to her feet, for it flashed
across her mind that she had never before seen a rabbit with either a
waistcoat-pocket, or a watch to take out of it, and burning with
curiosity, she ran across the field after it, and was just in time to
see it pop down a large rabbit-hole under the hedge.

In another moment down went Alice after it, never once considering how
in the world she was to get out again.
```

Notice that `alice_text` is now stored as one long string. Sometimes you will want that. Other times it will be convenient to instead store your text as a list of individual lines instead of one big string.

To store the text as a list of strings, use the file method `readlines()` instead of `read()`:

```
with open(alice_filename, "r") as f:
    alice_list = f.readlines() # to read all lines and then return
them as each line a string element in a list
```

```
type(alice_list)
```

```
list
```

```
alice_list
```

```
['Alice was beginning to get very tired of sitting by her sister on
the bank, and of having nothing to do: once or twice she had peeped
into the book her sister was reading, but it had no pictures or
conversations in it, "and what is the use of a book," thought Alice,
"without pictures or conversations?"\n',
 '\n',
 'So she was considering in her own mind (as well as she could, for
the hot day made her feel very sleepy and stupid) whether the pleasure
of making a daisy-chain would be worth the trouble of getting up and
picking the daisies, when suddenly a White Rabbit with pink eyes ran
close by her.\n',
 '\n',
 'There was nothing so very remarkable in that; nor did Alice think it
so very much out of the way to hear the Rabbit say to itself, "Oh
dear! Oh dear! I shall be too late!" (when she thought it over
afterwards, it occurred to her that she ought to have wondered at
this, but at the time it all seemed quite natural); but when the
Rabbit actually took a watch out of its waistcoat-pocket, and looked
at it, and then hurried on, Alice started to her feet, for it flashed
across her mind that she had never before seen a rabbit with either a
waistcoat-pocket, or a watch to take out of it, and burning with
curiosity, she ran across the field after it, and was just in time to
see it pop down a large rabbit-hole under the hedge.\n',
 '\n',
 'In another moment down went Alice after it, never once considering
how in the world she was to get out again.']
```

```
len(alice_list)
```

```
7
```

```
for line in alice_list:
    print(line)
```

```
Alice was beginning to get very tired of sitting by her sister on the
bank, and of having nothing to do: once or twice she had peeped into
the book her sister was reading, but it had no pictures or
conversations in it, "and what is the use of a book," thought Alice,
"without pictures or conversations?"



So she was considering in her own mind (as well as she could, for the
hot day made her feel very sleepy and stupid) whether the pleasure of
making a daisy-chain would be worth the trouble of getting up and
picking the daisies, when suddenly a White Rabbit with pink eyes ran
close by her.



There was nothing so very remarkable in that; nor did Alice think it
```

```
so very much out of the way to hear the Rabbit say to itself, "Oh
dear! Oh dear! I shall be too late!" (when she thought it over
afterwards, it occurred to her that she ought to have wondered at
this, but at the time it all seemed quite natural); but when the
Rabbit actually took a watch out of its waistcoat-pocket, and looked
at it, and then hurried on, Alice started to her feet, for it flashed
across her mind that she had never before seen a rabbit with either a
waistcoat-pocket, or a watch to take out of it, and burning with
curiosity, she ran across the field after it, and was just in time to
see it pop down a large rabbit-hole under the hedge.



In another moment down went Alice after it, never once considering how
in the world she was to get out again.
```

```
alice_list[0]
```

```
'Alice was beginning to get very tired of sitting by her sister on the
bank, and of having nothing to do: once or twice she had peeped into
the book her sister was reading, but it had no pictures or
conversations in it, "and what is the use of a book," thought Alice,
"without pictures or conversations?"\n'
```

```
alice_list[1]
```

```
'\n'
```

```
alice_list[2]
```

```
'So she was considering in her own mind (as well as she could, for the
hot day made her feel very sleepy and stupid) whether the pleasure of
making a daisy-chain would be worth the trouble of getting up and
picking the daisies, when suddenly a White Rabbit with pink eyes ran
close by her.\n'
```

We can now do anything with this list that we could do with any other list:

```
for line in alice_list:
    if "Alice" in line:
        print(line)
```

```
Alice was beginning to get very tired of sitting by her sister on the
bank, and of having nothing to do: once or twice she had peeped into
the book her sister was reading, but it had no pictures or
conversations in it, "and what is the use of a book," thought Alice,
"without pictures or conversations?"

There was nothing so very remarkable in that; nor did Alice think it
so very much out of the way to hear the Rabbit say to itself, "Oh
dear! Oh dear! I shall be too late!" (when she thought it over
```

```
afterwards, it occurred to her that she ought to have wondered at
this, but at the time it all seemed quite natural); but when the
Rabbit actually took a watch out of its waistcoat-pocket, and looked
at it, and then hurried on, Alice started to her feet, for it flashed
across her mind that she had never before seen a rabbit with either a
waistcoat-pocket, or a watch to take out of it, and burning with
curiosity, she ran across the field after it, and was just in time to
see it pop down a large rabbit-hole under the hedge.

In another moment down went Alice after it, never once considering how
in the world she was to get out again.
```

As a reminder, the `f` variable I've been using in the with/as statement is a temporary variable that can be anything, just like when writing a for loop. `f` is just a commonly used shorthand in with/as statements.

```
with open(alice_filename, "r") as FN_3214:
    alice_list = FN_3214.readlines()
len(alice_list)
```

Writing files

*Remember that when you open a file in write mode, it will first create a new empty file. If you already have a file with the same name, it will **EMPTY** that file.*

Let's work with our `alice_list`:

```
for line in alice_list:
    print(line)
```

```
Alice was beginning to get very tired of sitting by her sister on the
bank, and of having nothing to do: once or twice she had peeped into
the book her sister was reading, but it had no pictures or
conversations in it, "and what is the use of a book," thought Alice,
"without pictures or conversations?"



So she was considering in her own mind (as well as she could, for the
hot day made her feel very sleepy and stupid) whether the pleasure of
making a daisy-chain would be worth the trouble of getting up and
picking the daisies, when suddenly a White Rabbit with pink eyes ran
close by her.



There was nothing so very remarkable in that; nor did Alice think it
so very much out of the way to hear the Rabbit say to itself, "Oh
dear! Oh dear! I shall be too late!" (when she thought it over
```

```
afterwards, it occurred to her that she ought to have wondered at
this, but at the time it all seemed quite natural); but when the
Rabbit actually took a watch out of its waistcoat-pocket, and looked
at it, and then hurried on, Alice started to her feet, for it flashed
across her mind that she had never before seen a rabbit with either a
waistcoat-pocket, or a watch to take out of it, and burning with
curiosity, she ran across the field after it, and was just in time to
see it pop down a large rabbit-hole under the hedge.



In another moment down went Alice after it, never once considering how
in the world she was to get out again.
```

Let's open a new file and write the Alice text without those extra empty new lines.

First, we'll save the name we want for our new file as a string:

```
new_alice = "alice_clean.txt"
```

Now we will open this new file in write mode using a with/as statement. Inside that statement, we will write each line of the `alice_list` as long as the line contains more than just the new line character:

```
with open(new_alice, "w") as f:
    for line in alice_list:
        if line != "\n":
            f.write(line)
```

To check the file, we can open it in read mode. We will just print the file inside the with/as statement without even saving it as a string or list:

```
with open(new_alice, "r") as f:
    print(f.read())
```

```
Alice was beginning to get very tired of sitting by her sister on the
bank, and of having nothing to do: once or twice she had peeped into
the book her sister was reading, but it had no pictures or
conversations in it, "and what is the use of a book," thought Alice,
"without pictures or conversations?"
So she was considering in her own mind (as well as she could, for the
hot day made her feel very sleepy and stupid) whether the pleasure of
making a daisy-chain would be worth the trouble of getting up and
picking the daisies, when suddenly a White Rabbit with pink eyes ran
close by her.
There was nothing so very remarkable in that; nor did Alice think it
so very much out of the way to hear the Rabbit say to itself, "Oh
dear! Oh dear! I shall be too late!" (when she thought it over
afterwards, it occurred to her that she ought to have wondered at
```

```
this, but at the time it all seemed quite natural); but when the
Rabbit actually took a watch out of its waistcoat-pocket, and looked
at it, and then hurried on, Alice started to her feet, for it flashed
across her mind that she had never before seen a rabbit with either a
waistcoat-pocket, or a watch to take out of it, and burning with
curiosity, she ran across the field after it, and was just in time to
see it pop down a large rabbit-hole under the hedge.
In another moment down went Alice after it, never once considering how
in the world she was to get out again.
```

Example: Turning a file into a clean list of lines

```
dog_file = "dogs.txt"
```

Let's read in the dog file and see what it looks like:

```
with open(dog_file, "r") as f:
    print(f.read())
```

```
affenpinscher
Afghan hound
Airedale terrier
Akita
Alaskan Malamute
American Staffordshire terrier
American water spaniel
Australian cattle dog
Australian shepherd
Australian terrier
basenji
basset hound
beagle
bearded collie
Bedlington terrier
Bernese mountain dog
bichon frise
black and tan coonhound
bloodhound
border collie
border terrier
borzoi
Boston terrier
bouvier des Flandres
boxer
briard
Brittany
Brussels griffon
bull terrier
bulldog
```

```
bullmastiff
cairn terrier
Canaan dog
Chesapeake Bay retriever
Chihuahua
Chinese crested
Chinese shar-pei
chow chow
Clumber spaniel
cocker spaniel
collie
curly-coated retriever
dachshund
Dalmatian
Doberman pinscher
English cocker spaniel
English setter
English springer spaniel
English toy spaniel
Eskimo dog
Finnish spitz
flat-coated retriever
fox terrier
foxhound
French bulldog
German shepherd
German shorthaired pointer
German wirehaired pointer
golden retriever
Gordon setter
Great Dane
greyhound
Irish setter
Irish water spaniel
Irish wolfhound
Jack Russell terrier
Japanese spaniel
keeshond
Kerry blue terrier
komondor
kuvasz
Labrador retriever
Lakeland terrier
Lhasa apso
Maltese
Manchester terrier
mastiff
Mexican hairless
Newfoundland
```

```
Norwegian elkhound
Norwich terrier
otterhound
papillon
Pekingese
pointer
Pomeranian
poodle
pug
puli
Rhodesian ridgeback
Rottweiler
Saint Bernard
saluki
Samoyed
schipperke
schnauzer
Scottish deerhound
Scottish terrier
Sealyham terrier
Shetland sheepdog
shih tzu
Siberian husky
silky terrier
Skye terrier
Staffordshire bull terrier
soft-coated wheaten terrier
Sussex spaniel
spitz
Tibetan terrier
vizsla
Weimaraner
Welsh terrier
West Highland white terrier
whippet
Yorkshire terrier
```

In the previous part, we learned that we can save this text as a list:

```
with open(dog_file, "r") as f:
    dog_list = f.readlines()

print(dog_list)
```

```
['affenpinscher\n', 'Afghan hound\n', 'Airedale terrier\n', 'Akita\n',
'Alaskan Malamute\n', 'American Staffordshire terrier\n', 'American
water spaniel\n', 'Australian cattle dog\n', 'Australian shepherd\n',
'Australian terrier\n', 'basenji\n', 'basset hound\n', 'beagle\n',
'bearded collie\n', 'Bedlington terrier\n', 'Bernese mountain dog\n',
```

```
'bichon frise\n', 'black and tan coonhound\n', 'bloodhound\n', 'border
collie\n', 'border terrier\n', 'borzoi\n', 'Boston terrier\n',
'bouvier des Flandres\n', 'boxer\n', 'briard\n', 'Brittany\n',
'Brussels griffon\n', 'bull terrier\n', 'bulldog\n', 'bullmastiff\n',
'cairn terrier\n', 'Canaan dog\n', 'Chesapeake Bay retriever\n',
'Chihuahua\n', 'Chinese crested\n', 'Chinese shar-pei\n', 'chow chow\
n', 'Clumber spaniel\n', 'cocker spaniel\n', 'collie\n', 'curly-coated
retriever\n', 'dachshund\n', 'Dalmatian\n', 'Doberman pinscher\n',
'English cocker spaniel\n', 'English setter\n', 'English springer
spaniel\n', 'English toy spaniel\n', 'Eskimo dog\n', 'Finnish spitz\
n', 'flat-coated retriever\n', 'fox terrier\n', 'foxhound\n', 'French
bulldog\n', 'German shepherd\n', 'German shorthaired pointer\n',
'German wirehaired pointer\n', 'golden retriever\n', 'Gordon setter\
n', 'Great Dane\n', 'greyhound\n', 'Irish setter\n', 'Irish water
spaniel\n', 'Irish wolfhound\n', 'Jack Russell terrier\n', 'Japanese
spaniel\n', 'keeshond\n', 'Kerry blue terrier\n', 'komondor\n',
'kuvasz\n', 'Labrador retriever\n', 'Lakeland terrier\n', 'Lhasa apso\
n', 'Maltese\n', 'Manchester terrier\n', 'mastiff\n', 'Mexican
hairless\n', 'Newfoundland\n', 'Norwegian elkhound\n', 'Norwich
terrier\n', 'otterhound\n', 'papillon\n', 'Pekingese\n', 'pointer\n',
'Pomeranian\n', 'poodle\n', 'pug\n', 'puli\n', 'Rhodesian ridgeback\
n', 'Rottweiler\n', 'Saint Bernard\n', 'saluki\n', 'Samoyed\n',
'schipperke\n', 'schnauzer\n', 'Scottish deerhound\n', 'Scottish
terrier\n', 'Sealyham terrier\n', 'Shetland sheepdog\n', 'shih tzu\n',
'Siberian husky\n', 'silky terrier\n', 'Skye terrier\n',
'Staffordshire bull terrier\n', 'soft-coated wheaten terrier\n',
'Sussex spaniel\n', 'spitz\n', 'Tibetan terrier\n', 'vizsla\n',
'Weimaraner\n', 'Welsh terrier\n', 'West Highland white terrier\n',
'whippet\n', 'Yorkshire terrier']
```

Each item in the list is a string. Most strings end in a new line character, which we would like to remove.

We can combine what we learned just now about opening files with what we learned about making new lists in a for loop with string functions.

First, make an empty list:

```
dog_list = []
```

Now, inside the with/as statement, you can loop through the lines in the file and append them to the empty list. But you also need to use a string function to remove the new line characters:

```
with open(dog_file, "r") as f:
    for line in f.readlines():
        dog_list.append(line.rstrip("\n")) # .rstrip() returns a copy
of the string with trailing characters removed
print(dog_list)
```

```
['affenpinscher', 'Afghan hound', 'Airedale terrier', 'Akita',
'Alaskan Malamute', 'American Staffordshire terrier', 'American water
spaniel', 'Australian cattle dog', 'Australian shepherd', 'Australian
terrier', 'basenji', 'basset hound', 'beagle', 'bearded collie',
'Bedlington terrier', 'Bernese mountain dog', 'bichon frise', 'black
and tan coonhound', 'bloodhound', 'border collie', 'border terrier',
'borzoi', 'Boston terrier', 'bouvier des Flandres', 'boxer', 'briard',
'Brittany', 'Brussels griffon', 'bull terrier', 'bulldog',
'bullmastiff', 'cairn terrier', 'Canaan dog', 'Chesapeake Bay
retriever', 'Chihuahua', 'Chinese crested', 'Chinese shar-pei', 'chow
chow', 'Clumber spaniel', 'cocker spaniel', 'collie', 'curly-coated
retriever', 'dachshund', 'Dalmatian', 'Doberman pinscher', 'English
cocker spaniel', 'English setter', 'English springer spaniel',
'English toy spaniel', 'Eskimo dog', 'Finnish spitz', 'flat-coated
retriever', 'fox terrier', 'foxhound', 'French bulldog', 'German
shepherd', 'German shorthaired pointer', 'German wirehaired pointer',
'golden retriever', 'Gordon setter', 'Great Dane', 'greyhound', 'Irish
setter', 'Irish water spaniel', 'Irish wolfhound', 'Jack Russell
terrier', 'Japanese spaniel', 'keeshond', 'Kerry blue terrier',
'komondor', 'kuvasz', 'Labrador retriever', 'Lakeland terrier', 'Lhasa
apso', 'Maltese', 'Manchester terrier', 'mastiff', 'Mexican hairless',
'Newfoundland', 'Norwegian elkhound', 'Norwich terrier', 'otterhound',
'papillon', 'Pekingese', 'pointer', 'Pomeranian', 'poodle', 'pug',
'puli', 'Rhodesian ridgeback', 'Rottweiler', 'Saint Bernard',
'saluki', 'Samoyed', 'schipperke', 'schnauzer', 'Scottish deerhound',
'Scottish terrier', 'Sealyham terrier', 'Shetland sheepdog', 'shih
tzu', 'Siberian husky', 'silky terrier', 'Skye terrier',
'Staffordshire bull terrier', 'soft-coated wheaten terrier', 'Sussex
spaniel', 'spitz', 'Tibetan terrier', 'vizsla', 'Weimaraner', 'Welsh
terrier', 'West Highland white terrier', 'whippet', 'Yorkshire
terrier']
```

A clean list of dogs!

Example: Turning a file into a dictionary

```
gradebook_file = "gradebook.csv"
```

Let's open the gradebook file and see what it looks like:

```
with open(gradebook_file, "r") as f:
    print(f.read())
```

```
name,hw1,hw2,hw3,exam1,exam2
Mary,10,7,9,91,89
Flo,6,6,7,79,82
Lia,8,9,10,92,95
Tim,7,6,7,93,87
```

```
Terry,8,10,10,93,90
```

Our end goal is to have a dictionary with the student's name as the key and a list of their grades as the values.

Ok, first let's store it as a list, but we want to leave out the first line of headers. When we call `f.readlines()` it turns the file into a list. We can index a list, so let's take all the lines except the first one:

```
with open(gradebook_file, "r") as f:
    gradebook = f.readlines()[1:]

for line in gradebook:
    print(line)
```

```
Mary,10,7,9,91,89

Flo,6,6,7,79,82

Lia,8,9,10,92,95

Tim,7,6,7,93,87

Terry,8,10,10,93,90
```

We can see that there are new line characters at the end of each line (because it is printing extra empty lines between the lines). Let's make a note of that.

We can apply what we know about lists and strings to make a list of what we need to code:

- make an empty dictionary
- loop through the gradebook list
- remove the new line characters from the end
- split the line on the commas
- separate the first item to be the key
- store the rest of the items as a list
- assign the key:value pairs to our dictionary

```
grade_dict = {} #make an empty dictionary
for line in gradebook: #loop through the gradebook list
    line2 = line.rstrip("\n") #remove the new line characters from the
end
    line_list = line2.split(",") #split the line on the commas
    name = line_list[0] #separate the first item to be the key
    grades = line_list[1:] #store the rest of the items as a list
    grade_dict[name] = grades #assign the key:value pairs to our
dictionary
```

```
print(grade_dict)
```

```
{'Mary': ['10', '7', '9', '91', '89'], 'Flo': ['6', '6', '7', '79',
'82'], 'Lia': ['8', '9', '10', '92', '95'], 'Tim': ['7', '6', '7',
'93', '87'], 'Terry': ['8', '10', '10', '93', '90']}
```

Reading files line by line

Sometimes you might be working with a very large file, with millions of lines, and you don't want to read it all into memory as a string or list.

There is a file method, `readline()`, that reads in only one line at a time. Notice the difference between `readlines()` that we have already seen and `readline()`. **I don't expect you to practice this method here, but I will give an example, so that you know it exists if you ever need to look it up.**

Let's imagine that there are millions of types of dogs (if only!) and our dogs.txt file is millions of lines long. We can use readline to loop through it and only store the dogs that we need for this notebook or script. This doesn't work the same way as `readlines()` because `readlines()` is a list and `readline()` is the string of only the first line. We need to use a while loop, which is something we aren't learning today:

```
dog_file = "dogs.txt"

hounds = []
with open(dog_file, "r") as f:
    line = f.readlines()
    print(line)
    while line:
        if "hound" in line:

            #****************************************
            hounds.append(line.rstrip("\n").lower())
            #****************************************

        #line = f.readline()
```

```
['affenpinscher\n', 'Afghan hound\n', 'Airedale terrier\n', 'Akita\n',
'Alaskan Malamute\n', 'American Staffordshire terrier\n', 'American
water spaniel\n', 'Australian cattle dog\n', 'Australian shepherd\n',
'Australian terrier\n', 'basenji\n', 'basset hound\n', 'beagle\n',
'bearded collie\n', 'Bedlington terrier\n', 'Bernese mountain dog\n',
'bichon frise\n', 'black and tan coonhound\n', 'bloodhound\n', 'border
collie\n', 'border terrier\n', 'borzoi\n', 'Boston terrier\n',
'bouvier des Flandres\n', 'boxer\n', 'briard\n', 'Brittany\n',
'Brussels griffon\n', 'bull terrier\n', 'bulldog\n', 'bullmastiff\n',
'cairn terrier\n', 'Canaan dog\n', 'Chesapeake Bay retriever\n',
'Chihuahua\n', 'Chinese crested\n', 'Chinese shar-pei\n', 'chow chow\
n', 'Clumber spaniel\n', 'cocker spaniel\n', 'collie\n', 'curly-coated
```

```
retriever\n', 'dachshund\n', 'Dalmatian\n', 'Doberman pinscher\n',
'English cocker spaniel\n', 'English setter\n', 'English springer
spaniel\n', 'English toy spaniel\n', 'Eskimo dog\n', 'Finnish spitz\
n', 'flat-coated retriever\n', 'fox terrier\n', 'foxhound\n', 'French
bulldog\n', 'German shepherd\n', 'German shorthaired pointer\n',
'German wirehaired pointer\n', 'golden retriever\n', 'Gordon setter\
n', 'Great Dane\n', 'greyhound\n', 'Irish setter\n', 'Irish water
spaniel\n', 'Irish wolfhound\n', 'Jack Russell terrier\n', 'Japanese
spaniel\n', 'keeshond\n', 'Kerry blue terrier\n', 'komondor\n',
'kuvasz\n', 'Labrador retriever\n', 'Lakeland terrier\n', 'Lhasa apso\
n', 'Maltese\n', 'Manchester terrier\n', 'mastiff\n', 'Mexican
hairless\n', 'Newfoundland\n', 'Norwegian elkhound\n', 'Norwich
terrier\n', 'otterhound\n', 'papillon\n', 'Pekingese\n', 'pointer\n',
'Pomeranian\n', 'poodle\n', 'pug\n', 'puli\n', 'Rhodesian ridgeback\
n', 'Rottweiler\n', 'Saint Bernard\n', 'saluki\n', 'Samoyed\n',
'schipperke\n', 'schnauzer\n', 'Scottish deerhound\n', 'Scottish
terrier\n', 'Sealyham terrier\n', 'Shetland sheepdog\n', 'shih tzu\n',
'Siberian husky\n', 'silky terrier\n', 'Skye terrier\n',
'Staffordshire bull terrier\n', 'soft-coated wheaten terrier\n',
'Sussex spaniel\n', 'spitz\n', 'Tibetan terrier\n', 'vizsla\n',
'Weimaraner\n', 'Welsh terrier\n', 'West Highland white terrier\n',
'whippet\n', 'Yorkshire terrier']
```

```
---------------------------------------------------------------------
-----
KeyboardInterrupt                         Traceback (most recent call
last)
Cell In[12], line 7
      5 line = f.readlines()
      6 print(line)
----> 7 while line:
      8     if "hound" in line:
      9
     10         #**************************************
     11         hounds.append(line.rstrip("\n").lower())

KeyboardInterrupt:
```

```
print(hounds)
```

```
['afghan hound', 'basset hound', 'black and tan coonhound',
'bloodhound', 'foxhound', 'greyhound', 'irish wolfhound', 'norwegian
elkhound', 'otterhound', 'scottish deerhound']
```