# Variables

*Note: You can explore the associated workbook for this chapter in the cloud.*

Variables are one of the fundamental building blocks of Python. A variable is like a tiny container where you store values and data, such as filenames, words, numbers, collections of words and numbers, and more.



## Assigning Variables

The variable name will point to a value that you "assign" it. You might think about variable assignment like putting a value "into" the variable, as if the variable is a little box 🎁

You assign variables with an equals `=` sign. In Python, a single equals sign `=` is the "assignment operator." A double equals sign `==` is the "real" equals sign.

```
new_variable = 100
print(new_variable)
```

```
4
```

```
2 * 2 == 4
```

```
True
```

```
different_variable = "I'm another variable!"
print(different_variable)
```

```
I'm another variable!
```

Let's look at some of the variables that we used when we counted the most frequent words in Charlotte Perkins Gilman's "The Yellow Wallpaper."

```
# Import Libraries and Modules

import re
from collections import Counter

# Define Functions

def split_into_words(any_chunk_of_text):
    lowercase_text = any_chunk_of_text.lower()
    split_words = re.split("\W+", lowercase_text)
    return split_words

# Define Filepaths and Assign Variables

filepath_of_text = "../texts/literature/The-Yellow-Wallpaper_Charlotte-Perkins-
Gilman.txt"
number_of_desired_words = 40

stopwords = ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',
'your', 'yours',
'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers',
 'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
 'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are',
 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does',
 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until',
 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down',
 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once',
'here',
 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',
 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so',
 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now', 've',
'll', 'amp']

# Read in File

full_text = open(filepath_of_text, encoding="utf-8").read()

# Manipulate and Analyze File

all_the_words = split_into_words(full_text)
meaningful_words = [word for word in all_the_words if word not in stopwords]
meaningful_words_tally = Counter(meaningful_words)
most_frequent_meaningful_words =
meaningful_words_tally.most_common(number_of_desired_words)

# Output Results

most_frequent_meaningful_words
```

We made the variables:

- `filepath_of_text`
- `stopwords`
- `number_of_desired_words`
- `full_text`

```
filepath_of_text = "../texts/literature/The-Yellow-Wallpaper_Charlotte-Perkins-
Gilman.txt"
number_of_desired_words = 40
stopwords = ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',
'your', 'yours',
 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers',
 'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
 'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are',
 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does',
 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until',
 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down',
 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once',
'here',
 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',
 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so',
 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now', 've',
'll', 'amp']

full_text = open(filepath_of_text, encoding="utf-8").read()
```

## Jupyter Display vs `Print()`

We can check to see what's "inside" these variables by running a cell with the variable's name. This is one of the handiest features of a Jupyter notebook. Outside the Jupyter environment, you would need to use the `print()` function to display the variable.

```
filepath_of_text
```

```
stopwords
```

```
number_of_desired_words
```

Your turn! Pick another variable from the script above and see what's inside it below.

```
#your_chosen_variable
```

You can run the `print` function inside the Jupyter environment, too. This is sometimes useful because Jupyter will only display the last variable in a cell, while `print()` can display multiple variables. Additionally, Jupyter will display text with `\n` characters (which means "new line"), while `print()` will display the text appropriately formatted with new lines.

For example, with the `print()` function, each of the variables are printed, and the "The Yellow Wallpaper" is properly formatted with new lines.

```
print(filepath_of_text)
print(stopwords)
print(number_of_desired_words)
print(full_text)
```

# Variable Names

Though we named our variables `filepath_of_text`, `stopwords`,`number_of_desired_words`, and `full_text`, we could have named them almost anything else.

Variable names can be as long or as short as you want, and they can include:

- upper and lower-case letters (A-Z)
- digits (0-9)
- underscores (_)

However, variable names *cannot* include:

- ❌ other punctuation (-.!?@)
- ❌ spaces ( )
- ❌ a reserved Python word

Instead of `filepath_of_text`, we could have simply named the variable `filepath`.

```
filepath = "../texts/literature/The-Yellow-Wallpaper_Charlotte-Perkins-Gilman.txt"
filepath
```

Or we could have gone even simpler and named the filepath `f`.

```
f = "../texts/literature/The-Yellow-Wallpaper_Charlotte-Perkins-Gilman.txt"
f
```

## Striving for Good Variable Names

As you start to code, you will almost certainly be tempted to use extremely short variables names like `f`. Your fingers will get tired. Your coffee will wear off. You will see other people using variables like `f`. You'll promise yourself that you'll definitely remember what `f` means. But you probably won't.

So, resist the temptation of bad variable names! Clear and precisely-named variables will:

- make your code more readable (both to yourself and others)
- reinforce your understanding of Python and what's happening in the code
- clarify and strengthen your thinking

**Example Python Code ❌ With Unclear Variable Names❌**

For the sake of illustration, here's some of our same word count Python code with poorly named variables. The code works exactly the same as our original code, but it's a lot harder to read.

```
def sp(t):
    lt = t.lower()
    sw = re.split("\W+", lt)
    return sw

f = "../texts/literature/The-Yellow-Wallpaper_Charlotte-Perkins-Gilman.txt"
ft = open(f, encoding="utf-8").read()

words = sp(ft)
```

**Example Python Code ✨ With Clearer Variable Names ✨**

```
def split_into_words(any_chunk_of_text):
    lowercase_text = any_chunk_of_text.lower()
    split_words = re.split("\W+", lowercase_text)
    return split_words

filepath_of_text = "../texts/literature/The-Yellow-Wallpaper_Charlotte-Perkins-
Gilman.txt"
full_text = open(filepath_of_text, encoding="utf-8").read()

all_the_words = split_into_words(full_text)
```

## Off-Limits Names

The only variable names that are off-limits are names that are reserved by, or built into, the Python programming language itself — such as `print`, `True`, and `list`.

This is not something to worry too much about. You'll know very quickly if a name is reserved by Python because it will show up in green and often give you an error message.

```
True = "../texts/literature/The-Yellow-Wallpaper_Charlotte-Perkins-Gilman.txt"
```

```
  File "<ipython-input-10-fbaebf398d20>", line 1
    True = "../texts/The-Yellow-Wallpaper.txt"
                                              ^
SyntaxError: can't assign to keyword
```

# Re-Assigning Variables

Variable assignment does not set a variable in stone. You can later re-assign the same variable a different value.

For instance, I could re-assign `filepath_of_text` to the filepath for the lyrics of Beyonce's album *Lemonade* instead of Perkins-Gilman's "The Yellow Wallpaper."

```
filepath_of_text = "../texts/music/Beyonce-Lemonade.txt"
```

If I change this one variable in our example code, then we get the most frequent words for *Lemonade*.

```
import re
from collections import Counter

def split_into_words(any_chunk_of_text):
    lowercase_text = any_chunk_of_text.lower()
    split_words = re.split("\W+", lowercase_text)
    return split_words

filepath_of_text = "../texts/music/Beyonce-Lemonade.txt"
number_of_desired_words = 40

stopwords = ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',
'your', 'yours',
 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers',
 'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
 'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are',
 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does',
 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until',
 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down',
 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once',
'here',
 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',
 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so',
 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now', 've',
'll', 'amp']

full_text = open(filepath_of_text, encoding="utf-8").read()

all_the_words = split_into_words(full_text)
meaningful_words = [word for word in all_the_words if word not in stopwords]
meaningful_words_tally = Counter(meaningful_words)
most_frequent_meaningful_words =
meaningful_words_tally.most_common(number_of_desired_words)

most_frequent_meaningful_words
```

## Your Turn

Ok now it's your turn to change some variables and calculate a new word frequency! First, pick a new text file from the list below:

- `"../texts/music/Carly-Rae-Jepsen-Emotion.txt"`
- `"../texts/music/Mitski-Puberty-2.txt"`
- `"../texts/literature/Dracula_Bram-Stoker.txt"`
- `"../texts/literature/Little-Women_Louisa-May-Alcott.txt"`
- `"../texts/literature/Alice-in-Wonderland_Lewis-Carroll.txt"`

> **To choose from a wider list of files...**

Then assign `filepath_of_text` to the corresponding filepath below. Be sure to explore what happens when you change the values for `number_of_desired_words` and `stopwords`, as well.

```
import re
from collections import Counter

def split_into_words(any_chunk_of_text):
    lowercase_text = any_chunk_of_text.lower()
    split_words = re.split("\W+", lowercase_text)
    return split_words

filepath_of_text = #Insert a New Text File Here
number_of_desired_words = #Change number of desired words

#Explore how the stopwords below affect word frequency by adding or removing stopwords
stopwords = ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',
'your', 'yours',
 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers',
 'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
 'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are',
 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does',
 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until',
 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down',
 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once',
'here',
 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',
 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so',
 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now', 've',
'll', 'amp']

full_text = open(filepath_of_text, encoding="utf-8").read()

all_the_words = split_into_words(full_text)
meaningful_words = [word for word in all_the_words if word not in stopwords]
meaningful_words_tally = Counter(meaningful_words)
most_frequent_meaningful_words =
meaningful_words_tally.most_common(number_of_desired_words)

most_frequent_meaningful_words
```

By Melanie Walsh