hw1ec

- this time: recipe folders and files
- and getting them organized...
- and here is the hw1 assignment page with a link to the details on hw1ec

In [1]:
```
# Where are we?
%pwd
```

Out[1]:
```
'/Users/MyDocuments/MyPython/College Courses/Harvey Mudd CS35/Sci50 2024 Spring/Homework/Week1/week1_spr24/hw1ec'
```

In [2]:
```
# in case we need to move around
# Reminder:    . is the current directory, so cd . simply stays put!
%cd .
```

```
/Users/MyDocuments/MyPython/College Courses/Harvey Mudd CS35/Sci50 2024 Spring/Homework/Week1/week1_spr24/hw1ec
```

In [3]:
```
# we will use a few file-handling libraries. These are built-in to python:
# (feel free to look over thier online documentation; the official python docs are esp. go
import os
import os.path
import shutil
```

In [4]:
```
# How to use Python's built-in index-finding for lists (L.index)
#
# I can never remember these things... Happily, there's no need to! :-)
#

L = [5,6,7]    # a list...
L.index(7)     # the index should be 2
```

Out[4]:
```
2
```

In [5]:
```
#
# here is an example function that handles the contents of a file by
#      looping through its lines
#      and, from there, breaking up it
#

def kgcontext(filepath):
    """ shows how to get the context of the word "kilograms" from
        within the _contents_ of the file whose path is filepath
    """
    # probably should have an exception-handler here...
    # we won't for this example

    # open the file
    f = open(filepath, "r", encoding="utf-8")
    # get its full contents, as a string
    contents = f.read()
    # close the file (a gesture of generosity to your OS!)
    f.close()

    # let's get the file's lines:
    FILE_LINES = contents.split("\n")
    for line in FILE_LINES:
        LoW = line.split()
        if "kilograms" in LoW:
            print(f"The line with kilograms is\n  {line}")
            print(f"as a list of words, it's {LoW}")
            i = LoW.index('kilograms')
```

```
            print(f"and kg is at index {i}")

    return 42

filepath = "./recipe42.txt"
kgcontext(filepath)
```

```
The line with kilograms is
  4 kilograms of cumin
as a list of words, it's ['4', 'kilograms', 'of', 'cumin']
and kg is at index 1
42
```

Out[5]:

Onward into problem 3's file-analysis and -arrangements...

- Here is the homework page that describes all of hw0...
- As always, the problem has some specific challenges...
- And, some of the questions will be of your own design...
- We look forward to your customized-questions (and answers!)

Optional: File-moving, -creating, and -deleting

These are not needed, but it's good to know... you can programmatically create (and delete) files. *Caution!*

There here in case you create a final project that involves file-manipulation. These recipe files are a great starting point for such things (it's no worry if they're accidentally deleted, after all!)

In [ ]:
```
# An example of creating a directory named "poptarts"
currentdir = os.getcwd()  # cwd == current working directory
print(f"currentdir is {currentdir}")

# the "/" is the folder separator on Mac (it may work on windows, too - try it!)
dirtomake = currentdir + "/" + "poptarts"

os.mkdir(dirtomake)    # will work once, but not the second time!

print(f"created the directory\n {dirtomake}")  # announce that it worked
```

In [ ]:
```
#
# If "/" does not work, then "\\" may work   ("\\" is "backslash" and "/" is "forward slas
#
# Also, os.path.join is a built-in function to create pathnames with the appropriate
#       folder-separator character, either "/" or "\\"     Here is an example:
#
currentdir = os.getcwd()
newdir = "spam"
joined_dir = os.path.join(currentdir, newdir)
print(f"joined_dir is {joined_dir}")
```

In [ ]:
```
#
# let's copy a file (hw0pr3.py) to a new directory (one named poptarts)
#

filename = "hw0pr3.py"
```

```
currentfilepath = os.path.join(currentdir, filename)
newfilepath = os.path.join(currentdir, "poptarts", filename)

print(f"copying {currentfilepath}")
print(f"to {newfilepath}")

shutil.copy(currentfilepath,newfilepath)
```

In [ ]:

```
#
# let's try this again, this time using try ... except ... to handle possible errors
#

try:
    filename = "hw0pr3.py"

    currentfilepath = os.path.join(currentdir, filename)
    newfilepath = os.path.join(currentdir, "poptarts", filename)

    print(f"copying {currentfilepath}")
    print(f"to {newfilepath}")

    shutil.copy(currentfilepath,newfilepath)

except FileNotFoundError as e:
    print(f"***   Copying did not work   *** ")
    print(f"Python's error message: {e}")

# except NameError as e:
#     print(f"***   Copying did not work   *** ")
#     print(f"Python's error message: {e}")
```

In [ ]:

```
#
# It's important to know how powerful - and dangerous - a scripting language can be.
#
#      This cell illustrates this.
#      Do not run this without triple-checking it's what you want.
#      Even safer, simply never run it. Instead, delete from the User Interface's windows.
#

def delete_folder_with_contents( folder_path_or_name ):  # caution!
   """ be very, very careful! """
   import shutil
   if len(folder_path_or_name) < 7:     # too suspicious, don't run it
       print(f"Not deleting {folder_path_or_name}")
       return

   try:
       print(f"shutil.rmtree('{folder_path_or_name}')")
       # # shutil.rmtree(folder_path_or_name)   # this line does the deleting
       # Only uncomment the above line _after_ +ENSURING+ it will delete what's wanted!
   except OSError as e:
       print(f"Aargh! {folder_path_or_name} : {e.strerror}")

# here is the call to the function - note the many safety layers...
# delete_folder_with_contents("poptarts")
```