Introduction to Terminal apps as well as pickle()

A terminal app is an application which runs on simple terminal.
We have see how it works via *print()* and *input()*.
Terminal apps are quite useful in the world of **Linux & Mac**.
Some examples of apps in **Linux & Mac** are:

- *top*: which shows the running processes
- *vi* or *pico* or *nano* or *emacs*: which are some popular text editors

Let's examine how we can create some *simple* terminal apps.

Created by John C.S. Lui, May 28, 2018.

Important note: If you want to use and modify this notebook file, please acknowledge the author.

In [ ]:
```
# Import the sleep function.
from time import sleep

# Display a title bar.
print("\t**********************************************")
print("\t***  Hello to my students in CSCI2040!     ***")
print("\t**********************************************\n\n")

# Generate a series of output
for x in range(0,11):  #generate 11 numbers from 0 to 10
    print("See, I know how to count: %d" %x)

# Let's invoke the sleep function to "pause" for secs seconds
secs = int(input("Please enter the # of seconds you want me to sleep:"))  # request input
print("Going to sleep now .....")
sleep(secs)
print ("Great, I had a %d secs of nap." %secs)
```

Creating a simple greeter terminal program

REMEMBER TO SHOW THIS ON A TERMINAL !!!!!!

In [ ]:
```
# important some functions and libaries for our use
from time import sleep
import os

# Greeter is a terminal application that greets students,
#   and remembers new students, if any.

# Display a title bar.
print("\t**********************************************************")
print("\t***  Greeter - Hello old and new students of CSCI2040  ***")
print("\t**********************************************************")

sleep(3)   # Let's sleep for 3 seconds first

# Print a bunch of information, in short intervals
students = ['mary', 'nancy', 'elaine', 'susan', 'joey']

# Print each name 3 times.
for name in students:
    # Clear the screen before listing names.
    os.system('clear')

    # Display the title bar.
    print("\t**********************************************")
    print("\t***  Greeter - Hello old and new friends!  ***")
    print("\t**********************************************")

    print("\n\n")
    for x in range(0,3):
        print("Hello " + name.title(), "how are you doing?")

    # Pause for 1 second between batches.
    sleep(1)
```

Another example

In [ ]:
```
# ************ TRY THIS ON A COMPUTER **************
```

In [ ]:
```
# ^^^^^^^^^^^^ TRY THIS ON A COMPUTER ^^^^^^^^^^^^^^

from time import sleep
import os

# define a display function
def display_title_bar():
    # Clears the terminal screen, and displays a title bar.
    os.system('clear')

    print("\t**********************************************")
    print("\t***              TITLE BAR                 ***")
    print("\t**********************************************")



# Let us create a loop such that users can choose various options.

selection = ''     # initialize the selection
while selection != 'q':
    display_title_bar()

    # Display options for users
    print("\n[1] Meet a new friend.")
    print("[2] Talk to an existing friend.")
    print("[q] Quit.")

    selection = input("\nPlease selection the above options: ")

    # Based on the user's selection, provide proper response
    if selection == '1':
        print("\nHere is Lulu, your new friend.\n")
        sleep(3)   # Let's sleep for 3 seconds first
    elif selection == '2':
        print("\nHere is Mary, you have met her last week!\n")
        sleep(3)   # Let's sleep for 3 seconds first
    elif selection == 'q':
        print("\nThanks for playing. Bye.")
    else:
        print("\nIllegal option !!\n")
```

Let's extend the above program

In [ ]:

```
# ************ TRY THIS ON A COMPUTER **************

from time import sleep
import os

# define a display function
def display_title_bar():
    # Clears the terminal screen, and displays a title bar.
    os.system('clear')

    print("\t**********************************************")
    print("\t***              TITLE BAR                 ***")
    print("\t**********************************************")


def get_user_selection():
# Let users know what they can do.
    print("\n[1] Meet a new friend.")
    print("[2] Talk to an existing friend.")
    print("[q] Quit.")

    return input("What would you like to do? ")

# Let us create a loop such that users can choose various options.

names = []     # create a list to store names of existing friends
selection = ''     # initialize the selection
while selection != 'q':
    display_title_bar()

    selection = get_user_selection()

    # Based on the user's selection, provide proper response
    if selection == '1':
```

```
        name = input ("\nEnter name of your new friend")
        names.append(name)
    elif selection == '2':
        print("\nYour existing friends are:")
        for name in names:
            print(name.title())
    elif selection == 'q':
        print("\nThanks for playing. Bye.")
    else:
        print("\nIllegal option !!\n")
```

The use of "Pickle"

You use *pickle* to store the state of the Python on disk such that we can get it back in its original form later. Let's illustrate.

In [ ]:
```
# Enable pickle so we can do data storage and retrieval
import pickle

# This program asks the user for some professors, and stores them.

# Make an empty list to store new professor in.
professors = []

# Create a loop that lets users store new professors.
new_professor = ''
while new_professor != 'quit':
    print("\nPlease tell me a new professor to remember.")
    new_professor = input("Enter 'quit' to quit: ")
    if new_professor != 'quit':
        professors.append(new_professor)

# Try to save the data to the file 'professors.pydata'.
try:
    file_object = open('professors.pydata', 'wb')   # create a file descriptor
    pickle.dump(professors, file_object)            # write to the file pointed by the file descriptor
    file_object.close()                             # remember to close the file
```

```
    print("\nI have remembered the following professors: ")
    for professor in professors:
        print(professor.title())

except Exception as e:
    print(e)
    print("\nI couldn't figure out how to store the professors list. Sorry.")


# Let us reset the professors list
professors = []
print("\nThe professors list is now:", professors)

# Now let us try to open the file we created (and closed)

try:
    file_object = open('professors.pydata', 'rb')
    professors = pickle.load(file_object)
    file_object.close()
except:
    professors = []

# Show the professors that are stored so far.
if len(professors) > 0:
    print("The following professors are in stable storage: ")
    for professor in professors:
        print(professor.title())
else:
    print("We have an empty professors list.")
```

Description of the above program

The above program introduces four **new** concepts:

- A **try-except** block
    - A try-except block is used when you think a section of code might create an error. If an error occurs in a try block, the program does not end. Instead, program execution drops into the except block.
    - For the above program, we try to open a file to write out a list of professors.
    - If the file can not be opened for some reason, e.g., because the program doesn't have the permission to create a new file, then the program drops to the except block. In this case, we print the actual error message and a friendly message.

- **Opening and closing files**
    - It tries to open the file 'professors.pydata'.
        -- The program uses the flag 'w' to tell Python to open the file for writing. The 'b' argument tells Python to write the file in bytes.
        -- If successful, the open file can be used through the file_object file descriptor.
        -- If the file does not yet exist, this line creates the file, in the same directory as the program.
    - Finally, it closes the file once we are finished working with it.

- **A call to pickle.dump()**
    - Finally, it 'dumps' the list animals into the file that was opened. (*It is not dumped in a format that we can read.*)

- **A call to pickle.load()**
    - We reset the professors list, then try to read it out again on stable storage via pickle.load(fd)

Exercise

Write a program to to ask the users to have the following options:

- Input the names of friends he/she knows.
- Display all names of his/her friends
- Display the first 3 (or less) friends
- Remove a friend from the list
- Store the list in stable storage

In [ ]: