

## PROGRAMMING IN PYTHON

### ASSIGNMENT #2

This new homework is not to be turned in, because it involves examining code, not writing code.

I have attached a file named "testfordill.py".

Download it and run it. You will get one of two outputs:

If you are lucky, you will get:

```
% python3 testfordill.py
Your system has dill installed.
%
```

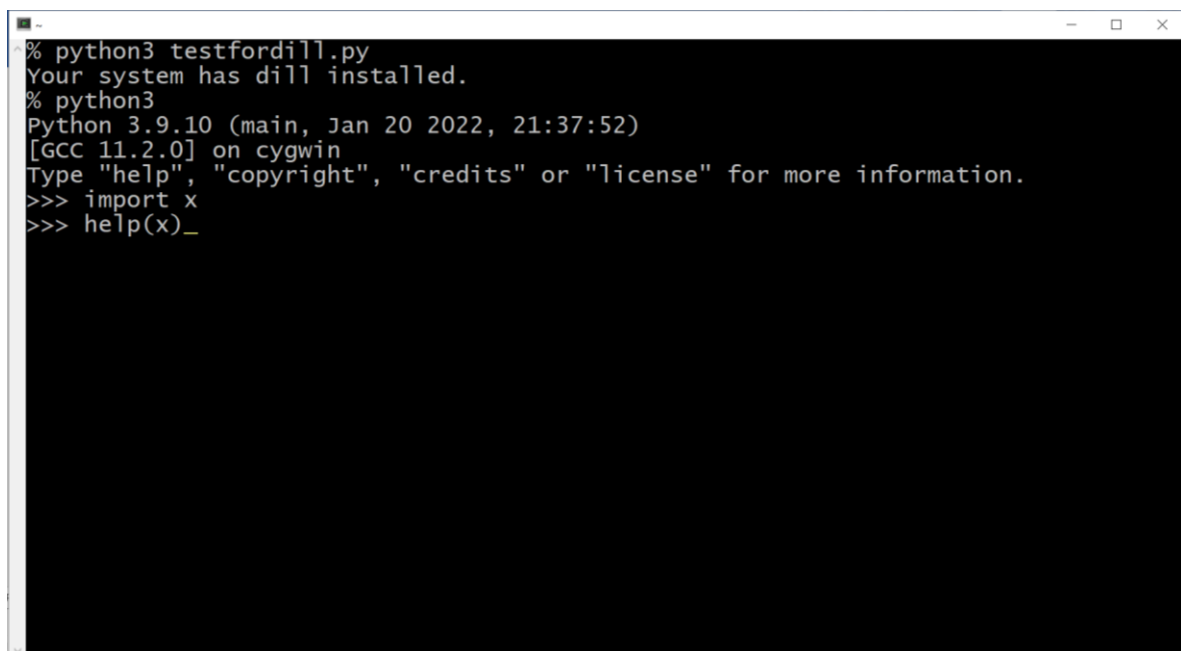
And if not lucky, you will get:

```
% python3 testfordill.py
You must install the 'dill' package. The procedure to do this will vary.
For cygwin running Python 3.9, type: /usr/bin/python3.9 -m pip install dill
%
```

If you get the second output, **then install dill into you Python system.**

Now, once you've done that, download the other provided file, "x.py".

Then import it and type help on it:



```
% python3 testfordill.py
Your system has dill installed.
% python3
Python 3.9.10 (main, Jan 20 2022, 21:37:52)
[GCC 11.2.0] on cygwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import x
>>> help(x)_
```

When you hit enter, you will get the following help screen:

```
Help on module x:

NAME
  x

DESCRIPTION
  This extension module has some useful functions.
  To use it, you MUST type: >>> from x import *; exec(x)
  (Don't worry about the details of x.py, or about what "exec" does.)

  To briefly summarize the tools provided in this module:
  -source :This prints out colorized source code.
  -shorterrs:This simplifies error messages. This is the default for x.py.
  -longerrs :This restores Python's original long error messages.

  -lprint: This stands for "line-print". It won't add a trailing newline.
  -lrange: This stands for "list-range". It's the same as: list(range(...)).
  -lmap: This stands for "list-map". It's the same as: list(map(...)).
  -lzip: This stands for "list-zip". It's mostly like: list(zip(...)).

  -pokerDemo:This is a provided example for learning x.py's features.

FUNCTIONS
  lmap(*args, **kws)
  :_
```

As you see above, this module has some little functions to slightly modify the python functions print, range, map, and zip. As you also see, it shortens the way error messages display. As you furthermore can see, it has a function called “source”, which we will be using in a few moments.

Finally, you can see that there is a function called “pokerDemo”. Let’s run that now:

```
>>> from x import *; exec(x)
>>> pokerDemo()
Unlike the other functions provided in x.py, you are encouraged to
inspect the source code of this function, by typing: source(pokerDemo).
The reason to inspect the source is that pokerDemo is a good teaching
example of a mostly-simple program that performs a real, useful task.
It also demos some of 'x.py': cardit, lrange, and lzip.

You are dealt these cards:
[['a', 2♣], ['b', 3♣], ['c', A♦], ['d', 6♦], ['e', 4♣]]

Enter the letters beside all cards to discard (a-e): a,b,d,e
[['a', A♦], ['b', A♣], ['c', 8♦], ['d', 8♣], ['e', 4♥]]

Enter the letters beside all cards to discard (a-e): e
[['a', A♦], ['b', A♣], ['c', 8♦], ['d', 8♣], ['e', A♥]]

Enter the letters beside all cards to discard (a-e):
[A♦, A♣, 8♦, 8♣, A♥]

Full House!

>>> source(pokerDemo)_
```

As you see, the poker game gives you 3 tries, then it scores your result. Test out the game yourself.

Also, you can see that we are next getting ready to run “source(pokerDemo)”.

Let’s hit enter and see what this “source” function does:

```

^>>> source(pokerDemo)
def pokerDemo():
    """Unlike the other functions provided in x.py, you are encouraged to
    inspect the source code of this function, by typing: source(pokerDemo).
    The reason to inspect the source is that pokerDemo is a good teaching
    example of a mostly-simple program that performs a real, useful task.
    It also demos some of 'x.py': cardit, lrange, and lzip."""
    fb('YB'); print(pokerDemo.__doc__); fb('WB')#Prints above comment in yellow.
    from random import shuffle
    deck=lrange(52); shuffle(deck)
    print("\nYou are dealt these cards:")
    for i in range(3):
        thisIs2x5=["abcde",deck[:5]]#Both the list's elements have lengths of 5.
        thisIs5x2=lzip(*thisIs2x5) #Transpose to 5-element list (see Lecture 4).
        cardit(thisIs5x2) #Type "x.help(cardit)" or "print(x.cardit.__doc__)".
        d=input("\nEnter the letters beside all cards to discard (a-e): ")
        if 'e' in d.lower(): del deck[4]
        if 'd' in d.lower(): del deck[3] # Ask yourself:
        if 'c' in d.lower(): del deck[2] # Why did these cards need to be
        if 'b' in d.lower(): del deck[1] # deleted from the right (ie: 4 to 0)?
        if 'a' in d.lower(): del deck[0]
    print() # First, try to understand this pokerDemo function.
    cardit(deck[:5]) # Only afterward, should you look at the scoreit function.
    scoreit(deck[:5]) # Then, when you are ready, type: source(x.scoreit)
^>>> source(x.scoreit)_

```

As you can see, it displays the source code for the `pokerDemo()` function, with pretty colors that make it easier to read.

**Part 1 of your current homework assignment** is to go through this function, line by-line, to understand how it works.

One thing you will notice is that you can type “`print(x.cardit.__doc__)`” to understand what `cardit` does (but don’t type “`source(x.cardit)`”, because it is too advanced for right now). Another thing you will notice is that there is a comment telling us to proceed onward by typing “`source(x.scoreit)`”. And in the screenshot above, I have typed that and am ready to hit enter. Let’s do that now:

```

^def scoreit(D):
    """This function prints the poker score of the five passed-in cards."""
    suit,face=lzip( *lmap(divmod,D,[13]*5) ) # Run: x.explain1stLineOfScoreit()
    z=""; print(); fb('WB') # Initialize z; print a space; set the pen color
    from operator import add # add(x,y) == x+y
    face=lmap(add,face,[2]*5) # shift the faces so that Two==2, Three==3, etc
    face=list(reversed(sorted(face))) # same as sorted(face,reverse=True)
    N = len(set(face))#This leverages the fact that sets don't have duplicates
    if N==2:
        if face[1]==face[3]: print("Four of a kind!")
        else: print("Full House!")
    elif N==3:
        if face.count(face[2])==3: print("Three of a kind!")
        else: print("Two pair!")
    elif N==4:
        print("One pair.")
    else:
        if face[:1]==[14,5]: face=[5,4,3,2,1] # Ace => One
        if face[4]==10: z="royal straight "
        elif face[4]+4==face[0]: z="straight "
        if len(set(suit))==1: z+="flush "
        if z != "": print(z[:-1].capitalize()+"!")
        else: print(["Seven","Eight","Nine","Ten","Jack",
                    "Queen","King","Ace"][face[0]-7],"high.")
    print(); fb('WB')
^>>> x.explain1stLineOfScoreit()

```

As you can see, the first line of code in `scoreit` is “`suit,face=lzip(*lmap(divmod,D,[13]*5))`”. This is an important line, and the comment for this line says that you should run `x.explain1stLineOfScoreit()`, in order to understand it. As you also see above, I have typed that command and am ready to hit enter:

```
if face.count(face[2])==3:print("Three of a kind!")
else:
    print("Two pair!")
elif N==4:
    print("One pair.")
else:
    if face[:1]==[14,5]:
        face=[5,4,3,2,1] # Ace => One
    if face[4]==10:
        z="royal straight "
    elif face[4]+4==face[0]:
        z="straight "
    if len(set(suit))==1:
        z+="flush "
    if z != "":
        print(z[:-1].capitalize()+"!")
    print(["Seven","Eight","Nine","Ten","Jack",
        "Queen","King","Ace"][face[0]-7],"high.")

print(); fb("WB")
>>> x.explain1stLineOfScoreit()

Let's unpack what 'suit,face=lzip(*lmap(divmod,D,[13]*5))' does:

We'll explain it slowly, in six steps:
1. ....[13]*5...
2. ....divmod(..., ...)..
3. ....lmap(divmod,D,[13]*5)..
4. ....( *lmap(divmod,D,[13]*5),)
5. ....lzip( *lmap(divmod,D,[13]*5) )
6. suit,face = lzip( *lmap(divmod,D,[13]*5) )

Hit enter to continue.
```

This is an interesting program. It runs interactively with you, and simulates that you were typing commands. **The second part of this homework** is to run `x.explain1stLineOfScoreit()` and take the time to understand it. You should run it several times to try to understand it well.

Once you have understood that first line of `scoreit()`, **the third part of your homework** is to understand all of the rest of the lines in `scoreit()`.