**Assignment #8**

Part 1: Array-ders of the Lost Ark!

Help Indiana “Python” Jones to solve a variety of list-based challenges. The game can be found [**here**](http://cs.nyu.edu/elearning/CSCI_UA_0002/PythonArrayders).

For this program you will not be writing a formal program with user input, etc. - you will simply be using Python as “scratch paper” to solve the problems being asked. Submit your “scratch work” solutions to the challenges, including all of the “passwords” you obtain along the way, to Brightspace using the following filename: LastnameFirstname\_assign8\_part1.py

Part 2: Secret Picture (extra credit, keep going - there are more required parts!)

The following list contains a series of coordinates which describe a secret picture:

secret = ['S',-204,198,'P',-204,198,'P',-204,198,'P',-201,135,'P',-199,104,'P',-199,104,'E',-199,104,'S',-199,140,'P',-199,140,'P',-187,140,'P',-176,144,'P',-165,147,'E',-165,147,'S',-168,205,'P',-168,205,'P',-168,149,'P',-165,87,'P',-164,74,'E',-164,74,'S',-141,91,'P',-141,98,'P',-141,161,'P',-128,150,'P',-104,99,'P',-100,93,'E',-100,93,'S',-140,116,'P',-132,116,'P',-115,116,'P',-115,116,'E',-115,116,'S',-96,152,'P',-94,139,'P',-72,50,'P',-71,44,'E',-71,44,'S',-95,147,'P',-95,147,'P',-85,152,'P',-61,143,'P',-77,108,'P',-87,104,'E',-87,104,'S',-48,148,'P',-48,148,'P',-46,130,'P',-26,42,'P',-26,42,'E',-26,42,'S',-48,147,'P',-48,147,'P',-31,147,'P',-16,138,'P',-17,125,'P',-36,110,'P',-41,107,'P',-41,107,'E',-41,107,'S',-5,152,'P',-5,152,'P',-3,125,'P',14,117,'P',22,139,'P',22,151,'P',31,121,'P',34,94,'P',30,67,'P',13,65,'E',13,65,'S',-194,-23,'P',-194,-23,'P',-173,-22,'P',-127,-15,'E',-127,-15,'S',-156,-24,'P',-156,-25,'P',-155,-89,'P',-153,-103,'E',-153,-103,'S',-140,-35,'P',-140,-47,'P',-140,-91,'P',-140,-96,'E',-140,-96,'S',-140,-75,'P',-140,-75,'P',-127,-69,'P',-119,-78,'P',-119,-96,'E',-119,-96,'S',-87,-68,'P',-87,-68,'P',-103,-70,'P',-110,-87,'P',-98,-95,'P',-89,-82,'P',-87,-70,'P',-80,-93,'P',-79,-97,'E',-79,-97,'S',-65,-99,'P',-65,-99,'P',-65,-79,'P',-65,-71,'P',-40,-100,'P',-30,-91,'P',-30,-65,'E',-30,-65,'S',-19,-20,'P',-18,-51,'P',-18,-87,'P',-18,-99,'E',-18,-99,'S',5,-53,'P',5,-57,'P',-13,-74,'P',-18,-77,'P',8,-95,'P',13,-101,'E',14,-101,'S',42,-57,'P',35,-57,'P',22,-58,'P',31,-76,'P',46,-82,'P',37,-106,'P',30,-100,'E',30,-100,'S',81,-64,'P',80,-64,'P',68,-68,'P',67,-93,'P',83,-85,'P',82,-71,'P',85,-88,'P',91,-132,'P',60,-129,'E',59,-128,'S',92,-67,'P',92,-89,'P',92,-97,'E',92,-97,'S',103,-68,'P',103,-68,'P',105,-88,'P',107,-93,'P',110,-85,'P',114,-70,'E',114,-69,'S',127,-68,'P',127,-71,'P',127,-96,'E',127,-96,'S',144,-96,'P',144,-96,'P',144,-77,'P',144,-69,'P',169,-101,'P',174,-74,'P',174,-71,'E',174,-71,'S',200,-69,'P',191,-69,'P',188,-87,'P',202,-90,'P',202,-72,'P',205,-82,'P',216,-127,'P',198,-136,'P',181,-143,'E',181,-143]

You can use Turtle graphics to draw the secret picture to the screen using the following information:

The canvas size of the picture is 500 pixels by 500 pixels. Use turtle.setup() to do this. Information in the list above is organized in groups of 3 - the first item in a group describes an action to be taken (‘S’, ‘P’, or ‘E’), the second item describes an X position on the canvas and the third item describes a Y position on the canvas. The action character ‘S’ means that we want to start drawing at the point describe directly after the ‘S’ character. Pick up your turtle’s pen (turtle.penup()) and move to the X and Y values that exist in the new two positions after the ‘S’ character. Use turtle.goto() to do this. The action character ‘P’ means we are drawing a line to the point described in the next two positions. The turtle should trace out a line to this point using a black color. The action character ‘E’ means that we are done drawing for now. The turtle can pick up its pen.

Submit your program to Brightspace using the following filename: LastnameFirstname\_assign8\_part2.py

Part 3a: Fast Food Restaurant

Note that the programs you will be writing for Part3a through Part 3f build on one another. You will want to attempt these parts in order. You can also feel free to save all of your work in one big file (call it “LastNameFirstName\_assign8\_part3.py”)

Imagine that you are helping to build a store management system for a fast food restaurant. Given the following lists, write a program that asks the user for a product name. Next, find out if the restaurant sells that item and report the status to the user. Allow the user to continually inquire about product names until they elect to quit the program.

Here are some lists to get you started:

# product lists

product\_names = ["soft drink", "onion rings", "small fries"]

product\_costs = [0.99, 1.29, 1.49]

And here’s a sample running of your program:

(s)earch for product or (q)uit: s

Enter a product name: soft drink

We sell "soft drink" at 0.99 per unit

(s)earch for product or (q)uit: s

Enter a product name: salad

Sorry, we don't sell "salad"

(s)earch for product or (q)uit: voldemort

Invalid option, try again

(s)earch for product or (q)uit: q

See you soon!

Part 3b

Next, extend your program so that it keeps track of how many products the store currently has to sell. Default the store to have 10 soft drinks, 5 onion ringss and 20 small fries. Hint: you might want to create a new list to store this data! - once you have this information in place you should modify your program to do the following:

Update the search feature to include a report of how many products the store currently has Add a new feature that lists ALL products, their prices and the amount the store has available to sell.

Here’s a sample running of your program:

(s)earch, (l)ist or (q)uit: s

Enter a product name: soft drink

We sell "soft drink" at 0.99 per unit

We currently have 10 in stock

(s)earch, (l)ist or (q)uit: l

Product Price Quantity

soft drink 0.99 10

onion rings 1.29 5

small fries 1.49 20

(s)earch, (l)ist or (q)uit: s

Enter a product name: pikachu

Sorry, we don't sell "pikachu"

(s)earch, (l)ist or (q)uit: q

See you soon!

Part 3c

Next, add in an “add” feature that lets users add a new product to the store. When you add a product you will need to ask the user for the name of the product, the cost of the product and how many items the store has available to sell. Validate this data - you cannot add a product that already exists and both the price and inventory amount must be positive. Here’s a sample running of the program:

(s)earch, (l)ist, (a)dd or (q)uit: l

Product Price Quantity

soft drink 0.99 10

onion rings 1.29 5

small fries 1.49 20

(s)earch, (l)ist, (a)dd or (q)uit: a

Enter a new product name: onion rings

Sorry, we already sell that product. Try again.

Enter a new product name: dbl onion rings

Enter a product cost: 0

Invalid cost. Try again.

Enter a product cost: 1.49

How many of these products do we have? -5

Invalid quantity. Try again.

How many of these products do we have? 100

Product added!

(s)earch, (l)ist, (a)dd or (q)uit: l

Product Price Quantity

soft drink 0.99 10

onion rings 1.29 5

small fries 1.49 20

dbl onion rings 1.49 100

(s)earch, (l)ist, (a)dd or (q)uit: s

Enter a product name: dbl onion rings

We sell "dbl onion rings" at 1.49 per unit

We currently have 100 in stock

(s)earch, (l)ist, (a)dd or (q)uit: q

See you soon!

Part 3d

Next, add in a “remove” feature that lets users remove products from the database. Here’s a sample running of the program:

(s)earch, (l)ist, (a)dd, (r)emove or (q)uit: l

Product Price Quantity

soft drink 0.99 10

onion rings 1.29 5

small fries 1.49 20

(s)earch, (l)ist, (a)dd, (r)emove or (q)uit: r

Enter a product name: pikachu

Product doesn't exist. Can't remove.

(s)earch, (l)ist, (a)dd, (r)emove or (q)uit: r

Enter a product name: small fries

Product removed!

(s)earch, (l)ist, (a)dd, (r)emove or (q)uit: l

Product Price Quantity

soft drink 0.99 10

onion rings 1.29 5

(s)earch, (l)ist, (a)dd, (r)emove or (q)uit: r

Enter a product name: soft drink

Product removed!

(s)earch, (l)ist, (a)dd, (r)emove or (q)uit: l

Product Price Quantity

onion rings 1.29 5

(s)earch, (l)ist, (a)dd, (r)emove or (q)uit: q

See you soon!

Part 3e

Add in the ability to modify a product. Users should be able to modify the name, cost or quantity of a product. Note that you will need to perform data validation as needed. Here’s a sample running of your program:

(s)earch, (l)ist, (a)dd, (r)emove, (u)pdate or (q)uit: l

Product Price Quantity

soft drink 0.99 10

onion rings 1.29 5

small fries 1.49 20

(s)earch, (l)ist, (a)dd, (r)emove, (u)pdate or (q)uit: u

Enter a product name: burger

Product doesn't exist. Can't update.

(s)earch, (l)ist, (a)dd, (r)emove, (u)pdate or (q)uit: u

Enter a product name: soft drink

What would you like to update?

(n)ame, (c)ost or (q)uantity: n

Enter a new name: onion rings

Duplicate name!

Enter a new name: soft drink royale

Product name has been updated

(s)earch, (l)ist, (a)dd, (r)emove, (u)pdate or (q)uit: l

Product Price Quantity

soft drink royale 0.99 10

onion rings 1.29 5

small fries 1.49 20

(s)earch, (l)ist, (a)dd, (r)emove, (u)pdate or (q)uit: u

Enter a product name: onion rings

What would you like to update?

(n)ame, (c)ost or (q)uantity: c

Enter a new cost: 0

Invalid cost!

Enter a new cost: 9.50

Product cost has been updated

(s)earch, (l)ist, (a)dd, (r)emove, (u)pdate or (q)uit: l

Product Price Quantity

soft drink royale 0.99 10

onion rings 9.50 5

small fries 1.49 20

(s)earch, (l)ist, (a)dd, (r)emove, (u)pdate or (q)uit: u

Enter a product name: small fries

What would you like to update?

(n)ame, (c)ost or (q)uantity: q

Enter a new quantity: -500

Invalid quantity!

Enter a new quantity: 30

Product quantity has been updated

(s)earch, (l)ist, (a)dd, (r)emove, (u)pdate or (q)uit: l

Product Price Quantity

soft drink royale 0.99 10

onion rings 9.50 5

small fries 1.49 30

(s)earch, (l)ist, (a)dd, (r)emove, (u)pdate or (q)uit: u

Enter a product name: soft drink royale

What would you like to update?

(n)ame, (c)ost or (q)uantity: nothing

Invalid option

(s)earch, (l)ist, (a)dd, (r)emove, (u)pdate or (q)uit: q

See you soon!

Part 3f

Finally, add in a reporting feature to your program that finds the highest priced product, the lowest priced product and the total value of all inventory in the store (product cost \* product quantity). Here’s a sample running of your program:

(s)earch, (l)ist, (a)dd, (r)emove, (u)pdate, r(e)port or (q)uit: l

Product Price Quantity

soft drink 0.99 10

onion rings 1.29 5

small fries 1.49 20

(s)earch, (l)ist, (a)dd, (r)emove, (u)pdate, r(e)port or (q)uit: e

Most expensive product: 1.49 (small fries)

Least expensive product: 0.99 (soft drink)

Total value of all products: 46.15

(s)earch, (l)ist, (a)dd, (r)emove, (u)pdate, r(e)port or (q)uit: q

See you soon!

Part 4: List Functions

Imagine that the most of the functions that we have been using to process lists in Python do not exist. For the problems below you are not allowed to use any list methods (append, remove, insert, etc) or certain core Python functions specified in each problem. You may, however, use your own custom-written functions to solve these problems (i.e. if you find that your listlen() function would be useful in writing another function you are welcome to use it). Given these restrictions, write a series of functions that do the following:

(1) Write a new function called “listlen” based on the following IPO:

# function: listlen

# INPUT: a list

# PROCESSING: determines the size of the list

# OUTPUT: an integer representing the size of the list

Note that you cannot use the “len” function in your program. If the list is empty your function should return a value of 0. Here is a sample running of this program:

mylist = [10, 20, 30]

x = listlen(mylist)

print (x)

>> 3

(2) Write a new function called “listmax” based on the following IPO

# function: listmax

# INPUT: a list

# PROCESSING: obtains the largest element in the list

# OUTPUT: returns the largest element in the list

Make sure that you DO NOT use the “max” function in your program. You cannot assume that the list supplied will be non-empty - if this is the case your function should return the value None. You can assume that the supplied list contains data elements of the same type (i.e. the list will contain all strings, or all ints, or all floats, or all Boolean values). The original list should NOT be changed after you call the ‘listmax’ function (and sorting the list will cause the list to change). Here is a sample running of this program:

mylist = [10, 20, 30, 5, 3]

x = listmax(mylist)

print (x)

print (mylist)

>> 30

>> [10, 20, 30, 5, 3]

(3) Write a new function called “listcopy” based on the following IPO

# function: listcopy

# INPUT: a list

# PROCESSING: creates a new list which serves as a copy of the supplied list

# OUTPUT: returns a new copy of the list

Make sure that you DO NOT use any list methods, including “append”, “insert” or “copy”. You cannot assume that the list supplied will be non-empty - if this is the case your function should return an empty list. Here is a sample running of this program:

mylist = [10, 20, 30]

x = listcopy(mylist)

print (x)

>> [10, 20, 30]

(4) Write a new function called “listappend” based on the following IPO

# function: listappend

# INPUT: a list and an element to add to the list (any data type)

# PROCESSING: creates a new list which includes the new element appended

# to the end of the list

# OUTPUT: returns a new copy of the list

Make sure that you DO NOT use any list methods such as “append” in your program. Note that your original list should not change when you run this function. Here is a sample running of this program:

mylist = [10, 20, 30]

x = listappend(mylist, 999)

print (x)

>> [10, 20, 30, 999]

(5) Write a new function called “listinsert” based on the following IPO

# function: listinsert

# INPUT: a list, a location (integer) and a data

# element (can be a string, float, Boolean or

# int)

# PROCESSING: inserts the supplied data element into the

# list at the desired location

# OUTPUT: return a new copy of the list that contains

# the inserted element

Make sure that you DO NOT use any list methods, such as the “insert” or “append” methods, in your program. Note that your original list should not change when you run this function. Here is a sample running of this program:

mylist = [10, 20, 30]

x = listinsert(mylist, 1, 999)

print (x)

print (mylist)

>> [10, 999, 20, 30]

>> [10, 20, 30]

(6) Write a new function called “listremove” based on the following IPO

# function: listremove

# INPUT: a list and a data element (can be a string,

# float, Boolean or int)

# PROCESSING: removes all occurrences of the supplied

# data element from the list. You can assume

# that the data element is present in the list

# somewhere.

# OUTPUT: return a new copy of the list with the

# desired element removed

Make sure that you DO NOT use any list methods, such as the “remove” method, or the “del” command in your program. Note that your original list should not change when you run this function. Here is a sample running of this program:

mylist = [10, 20, 30]

x = listremove(mylist, 20)

print (x)

print (mylist)

>> [10, 30]

>> [10, 20, 30]

(7) Write a new function called “listreverse” based on the following IPO

# function: listreverse

# INPUT: a list

# PROCESSING: creates a copy of the supplied list that

# contains the same elements as the original

# list, but in reverse order

# OUTPUT: return a new copy of the list in reverse order

Make sure that you DO NOT use any list methods such as “reverse” your program. Note that your original list should not change when you run this function. Here is a sample running of this program:

mylist = [10, 20, 30]

x = listreverse(mylist)

print (x)

print (mylist)

>> [30, 20, 10]

>> [10, 20, 30]

Submit your solutions to the challenges to Brightspace using the following filename. Note that you do not need to create a module to hold your functions: LastnameFirstname\_assign8\_part4.py

Part 5 (extra credit)

For this part you can choose to write a playable game of Blackjack for extra credit. In Blackjack the players is dealt a series of cards that have an associated point value. The goal is to collect a hand of cards that comes as close to 21 points as possible without going over 21.

In a Blackjack deck of cards every card has an associated value. All of the possible cards and the corresponding values can be found in the following lists. Note that the card at position 0 (10 of hearts) has its point value at position 0 in the second list:

cards = ['10 of Hearts', '9 of Hearts', '8 of Hearts', '7 of Hearts', '6 of Hearts', '5 of Hearts', '4 of Hearts', '3 of Hearts', '2 of Hearts', 'Ace of Hearts', 'King of Hearts', 'Queen of Hearts', 'Jack of Hearts', '10 of Diamonds', '9 of Diamonds', '8 of Diamonds', '7 of Diamonds', '6 of Diamonds', '5 of Diamonds', '4 of Diamonds', '3 of Diamonds', '2 of Diamonds', 'Ace of Diamonds', 'King of Diamonds', 'Queen of Diamonds', 'Jack of Diamonds', '10 of Clubs', '9 of Clubs', '8 of Clubs', '7 of Clubs', '6 of Clubs', '5 of Clubs', '4 of Clubs', '3 of Clubs', '2 of Clubs', 'Ace of Clubs', 'King of Clubs', 'Queen of Clubs', 'Jack of Clubs', '10 of Spades', '9 of Spades', '8 of Spades', '7 of Spades', '6 of Spades', '5 of Spades', '4 of Spades', '3 of Spades', '2 of Spades', 'Ace of Spades', 'King of Spades', 'Queen of Spades', 'Jack of Spades']

values = [10, 9, 8, 7, 6, 5, 4, 3, 2, [1,11], 10, 10, 10, 10, 9, 8, 7, 6, 5, 4, 3, 2, [1,11], 10, 10, 10, 10, 9, 8, 7, 6, 5, 4, 3, 2, [1,11], 10, 10, 10, 10, 9, 8, 7, 6, 5, 4, 3, 2, [1,11], 10, 10, 10]

Write a game of Blackjack that functions as follows. Note that you can earn partial extra credit if you only implement portions of this game:

* Start by dealing two cards at random to the player. Store these cards and the associated values in separate lists. You should remove the two cards from the deck once they have been dealt to the player so they cannot be dealt a second time.
* Report the player’s hand as well as the value of their hand. Note that Aces can be treated as having a value of 1 or 11 depending on the user’s preference. If the user is holding an Ace in their hand you should report two possible values for their hand (i.e. a 10 and an Ace could be reported as 21 or 11). Note that in the ‘values’ list Aces do not have a corresponding integer to represent their point value - instead, they have a list of their two possible values.
* Next, as the player if they want to “hit” or “stand”. “Hitting” means that the player wants another card. “Standing” means that the player is happy with their hand and doesn’t want any more cards.
* If the player chooses to “hit” you should deal them another card (and remove it from the main deck). If their hand is worth 21 points they win the game. If their hand is over 21 points they lose the game (they went “bust”). If they have less than 21 points the process should be repeated (they should be re-prompted to see if they want to “hit” or “stand”)
* If the player chooses to “stand” they will have less than 21 points. Next, you need to give the computer the opportunity to beat the player’s score. Begin by dealing two cards to the computer and report the computer’s hand and point value.
* Note that if the user has an Ace in their hand and treating that Ace as a 11 would cause them to go bust you should no longer report that to to the user (i.e. if they have a 10, 2 and Ace their two possible hand values are 23 and 13. Since a hand value of 23 would end the game you should not report that and instead automatically treat the Ace as though it was a 1)
* The computer should continually “hit” until one of the following conditions is met:
* The computer gets 21 points (blackjack)
* The computer earns more points than the player
* The computer goes “bust” (over 21 points)
* If the computer did not go bust then you will need to make a final determination to see who is the winner. You can do this by seeing which player has more points.
* Write in a system that allows the user to maintain a “bank” of money. Allow them to keep playing as long as their bank is above $0. Each hand that they choose to play should cost a small amount of money (say, $5) and if they win they will double their money. If the lose their initial bet is forfeited. This feature is not shown in the output below.

Here are some sample runnings of your program:

>>> ================================ RESTART ================================

>>>

Player hand: ['6 of Spades', '8 of Diamonds'] is worth 14

(h)it or (s)tand? h

You drew 3 of Clubs

Player hand: ['6 of Spades', '8 of Diamonds', '3 of Clubs'] is worth 17

(h)it or (s)tand? h

You drew 8 of Clubs

Player hand: ['6 of Spades', '8 of Diamonds', '3 of Clubs', '8 of Clubs'] is worth 25

Bust!

Computer Wins!

>>> ================================ RESTART ================================

>>>

Player hand: ['2 of Hearts', '5 of Diamonds'] is worth 7

(h)it or (s)tand? h

You drew 3 of Hearts

Player hand: ['2 of Hearts', '5 of Diamonds', '3 of Hearts'] is worth 10

(h)it or (s)tand? h

You drew 5 of Hearts

Player hand: ['2 of Hearts', '5 of Diamonds', '3 of Hearts', '5 of Hearts'] is worth 15

(h)it or (s)tand? h

You drew 9 of Clubs

Player hand: ['2 of Hearts', '5 of Diamonds', '3 of Hearts', '5 of Hearts', '9 of Clubs'] is worth 24

Bust!

Computer Wins!

>>> ================================ RESTART ================================

>>>

Player hand: ['Jack of Spades', 'King of Spades'] is worth 20

(h)it or (s)tand? s

Computer hand: ['2 of Spades', '5 of Hearts'] is worth 7

Computer drew Jack of Clubs

Computer hand: ['2 of Spades', '5 of Hearts', 'Jack of Clubs'] is worth 17

Computer drew 4 of Clubs

Dealer hand: ['2 of Spades', '5 of Hearts', 'Jack of Clubs', '4 of Clubs'] is worth 21

Computer got 21! Blackjack!

Computer wins!

>>> ================================ RESTART ================================

>>>

Player hand: ['King of Diamonds', 'Queen of Diamonds'] is worth 20

(h)it or (s)tand? s

Computer hand: ['5 of Diamonds', '10 of Clubs'] is worth 15

Computer drew 9 of Spades

Computer hand: ['5 of Diamonds', '10 of Clubs', '9 of Spades'] is worth 24

Bust!

Player wins!

>>> ================================ RESTART ================================

>>>

Player hand: ['Ace of Clubs', '2 of Clubs'] is worth 3 or 13

(h)it or (s)tand? s

Computer hand: ['Queen of Diamonds', '4 of Spades'] is worth 14

Computer wins!

>>> ================================ RESTART ================================

>>>

Player hand: ['Ace of Hearts', '3 of Clubs'] is worth 4 or 14

(h)it or (s)tand? h

You drew King of Hearts

Player hand: ['Ace of Hearts', '3 of Clubs', 'King of Hearts'] is worth 14

(h)it or (s)tand? h

You drew 8 of Clubs

Player hand: ['Ace of Hearts', '3 of Clubs', 'King of Hearts', '8 of Clubs'] is worth 22

Bust!

Computer Wins!

>>> ================================ RESTART ================================

>>>

Player hand: ['4 of Spades', '6 of Diamonds'] is worth 10

(h)it or (s)tand? h

You drew Ace of Hearts

Player hand: ['4 of Spades', '6 of Diamonds', 'Ace of Hearts'] is worth 11 or 21

You got Blackjack! You win!

>>> ================================ RESTART ================================

>>>

Player hand: ['Jack of Spades', '2 of Spades'] is worth 12

(h)it or (s)tand? h

You drew 6 of Clubs

Player hand: ['Jack of Spades', '2 of Spades', '6 of Clubs'] is worth 18

(h)it or (s)tand? s

Computer hand: ['3 of Spades', '10 of Clubs'] is worth 13

Computer drew Queen of Hearts

Computer hand: ['3 of Spades', '10 of Clubs', 'Queen of Hearts'] is worth 23

Bust!

Player wins!