#faces.py

#turn old style emoticons into emojis

def main():

    moods = input("How's your day going? ")

    convert(moods)

    print("do you mean ", moods)

#dictionary of moods

moods = {

    ":)", "🙂"

    ":(" "🙁"

    }

response= ""

for mood in moods:()

print (moods)

main()

#indoor.py

# turn uppercase into lower case letters

name = input ("What is your name? Please capitalize. ")

print ("hello,",(name).lower())

#playback.py

#to add spaces between words

name = input ()

print ((name).replace(" ", "..."))

#playing with spaces

friends=input("who is your best friend? ")

friends=f"\t{friends}   "

print (friends)

print (friends.lstrip())

print (friends.rstrip())

print (friends.strip())

#print quotations with line break

message=('Someone quoted Solomon as saying "A good name \nis rather to be chosen than riches."')

print (message)

#print extra long quote with value for name

famous\_person=input("who was just quoted? ")

message=(f'{famous\_person} is also quoted saying ')

message +=('"All the rivers run into the sea, yet the sea is not yet full"')

print (message)

#mixing integers(#'s) and str(text)

age=int(input("What is your age? "))

print(f'You were born in {2023-age}.')

#remove prefixes and suffixes

book\_website='https://nostarch.com/python-crash-course-3rd-edition'

book\_website=book\_website.removeprefix('https://').removesuffix('-3rd-edition')

print ("These exercises are from chapter two at", book\_website)

#play with numbers

print (5+3)

print (10-2)

print (2\*4)

print (16/2)

favorite\_number=16

print ('My favorite number is', favorite\_number)

#Python Crash Course ch 3 bcg 7/12/23

#making lists, individualize greetings, & replace name

friends=['Eric', 'Megan', 'Stacy', 'Frances']

print (friends)

print (friends[0], 'knows', friends[1],'and', friends[2])

for i in range (len (friends)):

    print (friends[i],'how are you doing?')

friends[0]='Monica'

print (friends)

#playing with lists to remove and pop

guest\_list=['Shakespeare', 'Greene', 'Atwood', 'King']

print (guest\_list)

for i in range (len(guest\_list)):

    print ('\nI would like to have dinner one day with',guest\_list[i],'.')

passed\_away=guest\_list.pop(0)

#guest\_list.remove(passed\_away)

#for i in range (len(guest\_list)):

print(f'\nBut {passed\_away} is no longer alive, so I will have to settle for the others (and a new one!):')

#playing with insert (at selected spots) & append (at end)

guest\_list.insert(0,'Moon')

for i in range (len(guest\_list)):

    print(i+1,(guest\_list[i]))

print ('I can invite more people? Okay, then! Let us add a few!')

guest\_list.insert(2,'Whedon')

guest\_list.append('Weber')

for i in range (len(guest\_list)):

    print ('\tPlease,',guest\_list[i],', come have dinner with an illustrious crew and me!')

#guest list count(

for i in range (len(guest\_list)):

    print (i+1)

#playing with pop to remove selected people (can I still keep them accessible?)

print('So maybe that is too many people.')

too\_many=guest\_list.pop(0)

print ('My apologies. Can I reschedule with you,',too\_many,'? Thank you!')

too\_many=guest\_list.pop(1)

print ('My apologies. Can I reschedule with you,',too\_many,'? Thank you!')

too\_many=guest\_list.pop(2)

print ('My apologies. Can I reschedule with you,',too\_many,'? Thank you!')

too\_many=guest\_list.pop(0)

print ('My apologies. Can I reschedule with you,',too\_many,'? Thank you!')

for i in range (len(guest\_list)):

    print ('\tPlease,',guest\_list[i],', I still would love to have dinner with you!')

#guest list count

for i in range (len(guest\_list)):

    print (1+i)

#to delete whole list and print empty list

unavailable='Weber'

guest\_list.remove(unavailable)

unavailable='Atwood'

guest\_list.remove(unavailable)

print ('Oh, well. I guess the guest list only includes me?',guest\_list)

#guest list count

for i in range (len(guest\_list)):

    print (1+i)

#sort and sorted, and reversed lists

bucket\_list=['London','Madrid','Jerusalem','Alaska','Oslo','Budapest']

print ('Here are places I would love to visit:')

for i in range (len(bucket\_list)):

    print ('\t',bucket\_list[i])

print ('You wanted them in alphabetical order?')

print(sorted(bucket\_list))

print ('You wanted them backwards from their original order?')

bucket\_list.reverse()

print (bucket\_list)

#permanently alphabetizing & reversing list order

print ('But alphabetical order is better.')

bucket\_list.sort()

print (bucket\_list)

print ('Cannot do it. I have an ornery streak.')

bucket\_list.sort(reverse=True)

print (bucket\_list)

#Python Crash Course ch 4 bcg 7/14/23

#lists, counting, min, max, range, cube

#4.1 list, & loops

pizzas=['pepperoni','pineapple','cheese', 'olives', 'sausage', 'onions']

for pizza in pizzas:

    print(f'{pizza.title()} tastes great on a pizza.')

    print(f'Should we order {pizza}?\n')

print ('I think we have some great choices!')

#4.10; 4.11 slices

callies\_pizza=pizzas[:]

print ('My favorite foods are:', (pizzas), '\nCallies favorite foods are:',(callies\_pizza))

pizzas.append('hamburger')

callies\_pizza.append('ranch')

print ('The complete list for me is:', (pizzas), '\nwhile Callies list is:',(callies\_pizza))

print ('The first three items in the list are:')

for pizza in pizzas[:3]:

    print('\t',(pizza))

print ('Three items in the middle of the list are')

for pizza in pizzas[2:5]:

    print('\t',(pizza))

print ('The last three items in the list are')

for pizza in pizzas[-3:]:

    print('\t',(pizza))

#4.3; 4.5 loop & counting, specifying odd numbers

numbers=list(range (1,21))

print (numbers)

odds=list(range (1,21,2))

for odd in odds:

    print (odd)

#4.4; 4.5 for loop to a million, min, max, & sum

numbers=list(range (1,1000001))

print ('Out of a million, the lowest nubmer is', min(numbers))

print ('The highest nubmer is', max(numbers))

large\_number=sum(numbers)

print (f'The sum of the first million numbers is {large\_number}.')

#4.7; 4.8; 4.9 Threes, cubes, append, & list comprehension

numbers=list(range(3, 30, 3))

for number in numbers:

    print (number)

cubes=[]

for value in range(1,11):

    cube=value\*\*3

    cubes.append(cube)

print (f'The first ten cubed numbers are {cubes}.')

squares=(f'The squares of the first ten numbers are', [value\*\*2 for value in range (1,11)])

print (squares)

#4.13 tuple on a buffet & re-assigned tuple

buffets=('chicken','pizza','wings','hot dogs','salad')

print ('The buffet serves: ')

for buffet in buffets:

    print (buffet)

print ('New Menu: The buffet now serves: ')

buffets=('chicken','pizza','wings','hot dogs','salad','garlic','ranch')

for buffet in buffets:

    print ('\t',(buffet))

#Python Crash Course ch 5 bcg 7/21/23

#if statements

#5.1 conditional tests

cars=['Honda','Toyota','BMW','Chevy','Mini']

print ('Is the car a Cadillac? I predict False:', 'Cadillac'in cars)

print ('Is the car a Ford? I predict False:', 'Ford' in cars)

print ('Is the car a Buick? I predict False:', 'Buick'in cars)

print ('Is the car a Tesla? I predict False:', 'Tesla'in cars)

print ('Is the car a Jeep? I predict False:', 'Jeep'in cars)

print ('Is the car a Honda? I predict True:', 'Honda'in cars)

print ('Is the car a Toyota? I predict True:', 'Toyota'in cars)

print ('Is the car a BMW? I predict True:', 'BMW'in cars)

print ('Is the car a Chevy? I predict True:', 'Chevy'in cars)

print ('Is the car a Mini? I predict True:', 'Mini'in cars)

#if with user input

your\_car=input ('What kind of car do you drive?')

if your\_car in cars:

    print (f'I have owned a {your\_car.title()} too!')

else:

    print  (f'I have never owned a {your\_car.title()}!')

age\_of\_car=int(input('How old is your car? '))

if age\_of\_car<=10:

    print (f'To me, your {your\_car.title()} is brand new!')

else:

    print ('There is something about owning an older car!')

if your\_car.title()=='Honda' and age\_of\_car==10:

    print ('We might be car twins!!')

if your\_car.title()=='Honda' or age\_of\_car==10:

    print ('We make similar choices!!')

#5.3, 5.4, 5.5 if, elif and else

aliens=['green','red','blue']

your\_alien=(input('What color is your alien: red, blue, or green? '))

if your\_alien=='green' in aliens:

    print (f'{your\_alien.title()}','just earned 5 points!')

elif your\_alien=='red' in aliens:

    print (f'{your\_alien.title()}','just earned 10 points!')

elif your\_alien=='blue' in aliens:

    print (f'{your\_alien.title()}','just earned 15 points!')

else:

    print (f'{your\_alien.title()}','is not one of your choices.')

#5.6 if, elif, else for integers

your\_age=int(input('How old are you? '))

if your\_age<2:

    print('Your stage of life is a baby.')

elif your\_age<4:

    print('Your stage of life is a toddler.')

elif your\_age<13:

    print('Your stage of life is a kid.')

elif your\_age<20:

    print('Your stage of life is a teenager.')

elif your\_age<65:

    print('Your stage of life is an adult.')

else:

    print('Your stage of life is an elder.')

#5.7 if statements with list

fruits=['apple','banana','orange']

favorite\_fruit=input('What is your favorite fruit? ')

if favorite\_fruit in fruits:

    print(f'Yeah!! I really like {favorite\_fruit}s.')

if favorite\_fruit=='grapes':

    print('Grapes can be taste good but they are not my favorite.')

if favorite\_fruit=='lemons':

    print('Lemons can be taste good but they are not my favorite.')

if favorite\_fruit=='limes':

    print('Limes can be taste good but they are not my favorite.')

if favorite\_fruit=='cantaloupe':

    print('Cantaloupe can be taste good but it is not my favorite.')

print ('Maybe we should get some today!')

#5.8 lists

usernames=['John','Philip','Becky','Eric','Beth','admin']

user\_login=input('What is your login name? ')

if user\_login=='admin':

    print ('Hello Admin. Would you like to see a current status report?')

elif user\_login in usernames:

    print (f'Hello, {user\_login}. Thank you for logging in today.')

else:

    print ('I am sorry. You are not a registered user. Please sign up.')

#5.9 empty lists

usernames=[]

print ('The following are users on this system: ',usernames)

if usernames==[]:

    print ('We need to find more users!')

#5.10 double lists to compare

current\_users=['eric','philip','becky','beth','morgan']

new\_users=['eric','beth','sunny','preston','maggie']

user\_login=input('What is your login name? ')

if user\_login in current\_users:

    print (f'Hello, {user\_login}. Thank you for logging in today.')

elif user\_login in new\_users:

    print (f'Hello, {user\_login}. Welcome to our program.')

#5.8 lists

usernames=['john','philip','becky','eric','beth','admin']

user\_login=input('What is your login name? ')

if user\_login=='admin':

    print ('Hello Admin. Would you like to see a current status report?')

elif user\_login in usernames:

    print (f'Hello, {user\_login.title()}. Thank you for logging in today.')

else:

    print ('I am sorry. You are not a registered user. Please sign up.')

#5.9 empty lists

usernames=[]

print ('\nThe following are users on this system: ',usernames)

if usernames==[]:

    print ('We need to find more users!\n')

#5.10 double lists to compare

current\_users=['eric','philip','becky','beth','morgan']

new\_users=['eric','beth','sunny','preston','maggie']

current\_users\_lower = [user.lower() for user in current\_users]

#user\_login=input('What is your login name? ')

for new\_user in new\_users:

    if new\_user in current\_users:

        print (f'I am sorry. The name {new\_user.title()} is not available.')

    else:

        print (f'Hello, {new\_user.title()}. Welcome to our program.')

#5.11 ordinal numbers in list

numbers=[1,2,3,4,5,6,7,8,9]

for number in numbers:

    if number==1:

        print (number,'st')

    elif number==2:

        print (number,'nd')

    elif number==3:

        print (number,'rd')

    else:

        print (number,'th')

#Python Crash Course ch 6 bcg 7/25/23

#dictionaries

#6.1 dictionary of 1 person

person={'first\_name':'Bethany','last\_name':'Gerlach','age':'51','city':'New Braunfels'}

print(person['first\_name'], person['last\_name'], person['age'], person['city'])

print (person)

for name,answer in person.items():

    print (f'\t{answer.title()}')

#6.2 #6.10 dictionary of favorite numbers, sorted, keys

favorite\_names={

    'Becky':['1','2'],

    'Philip':['2','3'],

    'Eric':['3','4'],

    'Morgan':['4','5'],

    'Bethany':'5'

    }

for name in sorted(favorite\_names.keys()):

    print (f'{name.title()}, I have your number!!')

last=favorite\_names['Bethany']

print (f'The highest number chosen is {last}.')

for name,number in favorite\_names.items():

    print (f"{name.title()}'s favorite number is {number}.")

#6.6 polling new input from old dictionary

need\_inputs=('Philip','Mallory','Morgan','Leslie','Becky')

for input in need\_inputs:

    if input in favorite\_names:

        print (f'\tHi, {input.title()}.')

    else:

        print (f'\tHi, {input.title()}. I need a number for you!')

#6.3, 6.4 dictionary of programming words keys, items

dictionary={

    'integers':'numbers that can be computed',

    'strings':'values that are interpreted as words and not read by a computer',

    'list':'collection of items in a particular order',

    'pop':'to remove an item from a list but retain access',

    'append':'to add an item to an already constructed list',

    }

for word, definition in dictionary.items():

    print (f'{word.title()}:  \t {definition.title()}\n')

#list words defined, and then definitions given

print ('The dictionary defines the following words:')

for word in dictionary.keys():

    print (word.title())

print ('These are the definitions used:')

for word, definition in dictionary.items():

    print (f'\t{definition.title()}')

#6.5 rivers dictionary, keys, sorted, set

rivers={

    'Mississippi':'United States',

    'Colorado':'United States',

    'Nile':'Egypt',

    'Thames':'England',

    'Rio Grande':'United States',

    'Rio Grande':'Mexico',

    }

for river, country in rivers.items():

    print (f'The {river.title()} is found in the country of {country.title()}.')

print ('These rivers \n')

for river in set(rivers.keys()):

    print (river)

print ('\nare in the following countries:')

for river, country in set(rivers.items()):

    print (country)

#6.8 dictionary of pets and their owners

pets = {

    'original': {

        'name': 'Kala',

        'owner': 'the Gerlachs',

        'origin': 'a gift from a friend',

        'status': 'died at age 15',

        },

    'second': {

        'name': 'Xena',

        'owner': 'the Gerlachs',

        'origin': 'adopted from PetSmart',

        'status': 'died at age 16',

        },

    'third': {

        'name': 'Buttercup',

        'owner': 'Deeadra Hill',

        'origin': 'adopted when owner became ill',

        'status': 'loved by Maggie',

        },

    }

#display list of information

for position, answers in pets.items():

    place=(position)

    name= answers['name']

    owner= answers['owner']

    origin= answers['origin']

    status= answers['status']

    print (f'The {place} dog, {name}, spoiled by {owner}, was {origin}, and {status}.')

#Python Crash Course ch 6 bcg 7/29/23

#dictionaries ordering pizza

#name of the orderer

name=input('What is your name? ')

print (f'\nHello, {name.title()}. Would you like to order a pizza?')

#choose a crust

crusts={'thin','regular','thick'}

crust\_choice=input(f'What kind of crust would you like, {name.title()}? ')

if crust\_choice in crusts:

    print (f'A {crust\_choice} crust sounds good.\n')

else:

    crust\_choice=input(f'I did not understand your choice. Would you like a thin, thick, or regular crust? ')

    if crust\_choice in crusts:

        print (f'A {crust\_choice} crust sounds good.\n')

#choose a sauce

sauces={'tomato','garlic butter','pesto'}

sauce\_choice=input(f'What kind of sauce would you like, {name.title()}? ')

if sauce\_choice in sauces:

    print (f'{sauce\_choice.title()} it is.\n')

else:

    sauce\_choice=input(f'I did not understand your choice. Would you like tomato, garlic butter, or pesto sauce? ')

    if sauce\_choice in sauces:

        print (f'{sauce\_choice.title()} it is.\n')

#choose toppings

toppings={

    'pepperoni','mushrooms','extra cheese',

    'ham','pineapple','green peppers',

    'onions','anchovies','sausage',

    'hamburger','olives','chicken',

    }

#ask how many toppings; ask that number of times

#7.4 pizza toppings #7.6 three exits

#take respondent's number of toppings then input that number of toppings

topping\_selection = []

current\_number = 0

request=input('\nHow many toppings do you want? \n')

request=int(request)

while current\_number < request:

    top=input ('What toppings would you like? ')

    topping\_selection.append(top)

    current\_number += 1

    continue

print (f'You have asked for {topping\_selection}.')

#print a summary of order

final\_toppings = []

#compare their order to available toppings

for topping\_sel in topping\_selection:

    if topping\_sel in toppings:

        final\_toppings.append(topping\_sel)

        print(f'We have {topping\_sel}, {name.title()}!')

    else:

        print(f'{name.title()}, we do not carry {topping\_sel}.')

        del (topping\_sel)

print (final\_toppings)

num\_topping=len(final\_toppings)

print (num\_topping)

#review order

print(f'You have ordered a {crust\_choice} crust pizza',

      f'with a {sauce\_choice} sauce,',

      f'and the following {num\_topping} toppings:\n{final\_toppings}. ',

      'Thank you for your order!')

#Python Crash Course ch 7 bcg 8/1/23

#user input and while loops

#7.1 rental car

car=input('What kind of car would you like? ')

print (f'Let me see if I can find you a {car.title()}.')

#7.2 restaurant seating

group=input ('\nHow many people are in your party? ')

group=int(group)

if group <=8:

    print ('We have a table for you. Please come this way.')

else:

    print ('I am sorry. The wait time will be about twenty minutes.')

#7.3 multiples of ten

multiple=input ('\nPlease input a number between 1 and 10,000. ')

multiple=int(multiple)

if multiple % 10 == 0:

    print (f'Your number, {multiple}, is divisible by ten.')

else:

    print (f'Your number, {multiple}, is not divisible by ten.')

#7.4 pizza toppings

#take respondent's order

order\_choice = []

item = ('\nWhat toppings would you like? ')

item +=('(Enter quit when finished.) ')

while True:

    question = input (item)

    if question == 'quit':

        break

    else:

        order\_choice.append(question)

        print (f'You have asked for {order\_choice}.')

#movie tickets

ticket\_age=input('\nYou would like to buy a movie ticket? How old are you? ')

ticket\_age=int(ticket\_age)

if ticket\_age<=3:

    print ('Your ticket is free.')

elif ticket\_age<=12:

    print ('Your ticket is $10.')

else:

    print ('Your ticket is $15.')

#three exits

#creating a loop to ask for inputted number of pizza toppings

toppings = []

current\_number = 0

request=input('\nHow many toppings do you want? ')

request=int(request)

while current\_number <request:

    top=input ('What toppings would you like? ')

    toppings.append(top)

    current\_number += 1

    continue

print (toppings)

#7.6 playing with while and active statements

#playing with printing messages until directed to quit

prompt = ('\nType something and I will repeat it; type quit to end. ')

active = True

while active:

    message = input (prompt)

    if message == 'quit':

        active = False

    elif message == 'salt':

        active = False

    else:

        print (message)

#7.8 deli

#filling a dictionary with user input

responses = {}

sandwiches = []

#set flag to indicate polling is active

polling\_active = True

while polling\_active:

    #prompt for name and order

    name = input ('\nWhat is your name? ')

    print ('\nThe deli has run out of pastrami.')

    response = input ('What type of sandwich would you like? ')

#store response in dictionary

    responses [name] = response

#ask for more input

    repeat = input ('Should I ask someone else? ')

    if repeat == 'no':

        polling\_active = False

#end of loop; show results

print ('\n    Poll Results')

for name, response in responses.items():

    sandwiches.append(response)

    print (f'{name.title()} would like to order a {response} sandwich.\n')

#move sandwiches from one list to a list of finished sandwiches

print(sandwiches)

finished\_sandwiches = []

#move sandwiches into finished sandwiches list

while sandwiches:

    current\_sandwich = sandwiches.pop()

    print (f'Making sandwich: {current\_sandwich.title()}')

    finished\_sandwiches.append(current\_sandwich)

#show all finished sandwiches

print ('\nThe following sandwiches have been made:')

for finished\_sandwich in finished\_sandwiches:

    print (finished\_sandwich.title())

#7.9 no pastrami

#removing pastrami sandwiches

sandwich\_menu = ['pastrami', 'cheese','ham', 'pastrami', 'turkey', 'cheese', 'pastrami', 'ham']

print (f'The following sandwiches are available: {sandwich\_menu}')

while 'pastrami' in sandwich\_menu:

    sandwich\_menu.remove('pastrami')

print ('\nThe corrected menu is as follows:')

print (f'The following sandwiches are available: \n {sandwich\_menu}')

#Python Crash Course ch 8 bcg 8/10/23

#functions

#8.1 message

def display\_message():

    '''to print message explaining purpose of ch 8: functions'''

    print ('This chapter explains how to use functions to eliminate repetitive coding.')

display\_message()

#8.2 favorite book

def favorite\_book(title):

    '''to print a message about the user's favorite book'''

    print (f'\nOne of my favorite books is {title.title()}!')

favorite\_book('Enders Game')

#print line between programs

print (f'\n')

#8.3 T-shirt

def make\_shirt(size, slogan):

        '''print size and slogan that goes on a T-shirt'''

        print (f'We are ordering a {size.title()} T-shirt.')

        print (f'The {size.title()} T-shirt should say: \n\t{slogan.title()}')

#run program make\_shirt

make\_shirt ('large', 'Firefly!')

make\_shirt (slogan='Ghostbusters', size='extra large')

#print line between programs

print (f'\n')

#8.4 Large Shirts

#playing with making & changing default arguments

def make\_shirts(slogan='I Love Python', size='large'):

        '''print differing size and slogan that goes on a T-shirt'''

        print (f'We are ordering a {size.title()} T-shirt. It should say: \n\t{slogan.title()}')

#run program make\_shirts (2nd version)

make\_shirts ()

make\_shirts (size='medium')

make\_shirts (slogan='Star Trek', size='extra large')

#print line between programs

print (f'\n')

#8.5 Cities

def describe\_city(city='Reykjavik', country='Iceland'):

      '''write simple sentence connecting a city and country'''

      print (f'{city.title()} is located in the country of {country.title()}.')

#run program describe\_city

describe\_city()

describe\_city(city='Akureyri')

describe\_city(country='Greenland',city='Nuuk')

#print line between programs

print (f'\n')

#8.6 City Names

def get\_city\_country\_format(city, country):

      '''write city and country address line'''

      address = f'{city}, {country}'

      return address.title()

location = get\_city\_country\_format('nuuk','greenland')

print (location)

location = get\_city\_country\_format('santiago', 'chile')

print (location)

location = get\_city\_country\_format('austin', 'united states')

print (location)

#print line between programs

print (f'\n')

#8.7 Album

#make a function to build a dictionary of music albums

def album\_list(musician, title, tracks=None):

    '''return a dictionary of music albums'''

    album = {'musician': musician, 'title': title}

    if tracks:

        album['tracks']= tracks

    return album

album = album\_list('Madonna','Like a Virgin', '9')

print (album)

album = album\_list('Billy Joel','Piano Man')

print (album)

album = album\_list('Michael Jackson','Thriller', '7')

print (album)

album = album\_list('Cyndie Lauper','Time After Time')

print (album)

#print line between programs

print (f'\n')

#8.8 User Albums

#get user input and return album info

def album\_list(musician, title):

    '''return a dictionary of music albums'''

    album = {'musician': musician, 'title': title}

    return album

while True:

    print ('\nWhat is your favorite album? (enter q to quit)')

    musician= input ('The musician: ')

    if musician== 'q':

        break

    title= input ('The title: ')

    if title== 'q':

        break

    album = (f'{title} by {musician}')

    print (f'The album, {album}, is awesome!')

#print line between programs

print (f'\n')

#8.9 Messages #8.10 Sending Messages

#use a function to use a list and move from one list to another

def messages(unsent, sents):

     '''to move messages from a list into new list'''

     while unsent:

          mail= unsent.pop()

          print (f'Message: {mail}')

          sents.append(mail)

def show\_messages(sents):

     print (f'\nThese are sent: ')

     for sent in sents:

          print (sent)

unsent = ['I love you!', 'You rock!', 'You and Me', 'BFFs 4ever']

sents = []

messages(unsent, sents)

show\_messages(sents)

print ('\nFinal List:')

print (f'\t{unsent}')

print (f'\t{sents}')

#print line between programs

print (f'\n')

#8.11 Messages using a copy of the list

#use a function to use a list without deleting list

def messages(unsent, sents):

     '''to move messages from a list into new list'''

     while unsent:

          mail= unsent.pop()

          print (f'Message: {mail}')

          sents.append(mail)

def show\_messages(sents):

     print (f'\nThese are sent: ')

     for sent in sents:

          print (sent)

unsent = ['I love you!', 'You rock!', 'You and Me', 'BFFs 4ever']

sents = []

messages(unsent[:], sents)

show\_messages(sents)

print ('\nFinal List:')

print (f'\t{unsent}')

print (f'\t{sents}')

#print line between programs

print (f'\n')

#8-12 Sandwiches

#make functions that operate regardless of # of arguments

def make\_sandwiches(\*ingredients):

     '''print the list of ingredients for each sandwich'''

     print ('The sandwich should include: ')

     print (f'\t~{ingredients}')

make\_sandwiches('ham','cheese', 'mustard')

make\_sandwiches('turkey','mayo','provolone','lettuce','tomato')

make\_sandwiches('egg salad, dry')

print (f'\n')

#8-13 User Profile

#play with function to create my own user profile

def build\_profile(first, last, \*\*user\_info):

     '''build a dictionary with information about a user'''

     user\_info['first\_name']= first

     user\_info['last\_name']= last

     return user\_info

user\_profile = build\_profile('Beth', 'Gerlach',

                             location='Texas',

                             age='51',

                             status='Masters degree')

print(user\_profile)

print (f'\n')

#8.14 Cars

#make a dictionary of car names; write function to print dictionary

def dealer\_info(manufacturer, model, \*\*car\_info):

     '''build a dictionary with info about cars'''

     car\_info['manufacturer']= manufacturer

     car\_info['model']= model

     return car\_info

car\_profile = dealer\_info('Kia','Sedona',

                          year='2001',

                          color='blue',

                          type='minivan')

print (car\_profile)

print (f'\n')

#(separate functions in separate py file for following program: )

#8.15 printing models

#import a function into another module

def print\_models (unprinted\_designs, completed\_models):

    '''move items from unfinished to completed list'''

    while unprinted\_designs:

        current\_design = unprinted\_designs.pop()

        print (f'Printing model: {current\_design}')

        completed\_models.append(current\_design)

def show\_completed\_models(completed\_models):

    '''show models that have been completed'''

    print (f'\nThe following models have been completed: ')

    for completed\_model in completed\_models:

        print (completed\_model)

#8.15 Printing Models

#import functions into this module

from printing\_functions.printing\_functions import print\_models

from printing\_functions.printing\_functions import show\_completed\_models

unprinted\_designs = ['phone case', 'robot pendant', 'dodecahedron']

completed\_models = []

print\_models(unprinted\_designs, completed\_models)

show\_completed\_models(completed\_models)