**CHAPTER 3-Introducing Lists**

## 3-1. Names: Store the names of a few of your friends in a list called names. Print

## each person’s name by accessing each element in the list, one at a time.

# CODE

names=list(["Shahab","Sami","Arham","Kanwal","Ramisha","Niazi"])

for i in names:

    {

        print(i)

    }

# OUTPUT

Text

Description automatically generated

## 3-2. Greetings: Start with the list you used in Exercise 3-1, but instead of just printing each person’s name, print a message to them. The text of each mes-sage should be the same, but each message should be personalized with the person’s name.

# CODE

names=list(["Shahab","Sami","Arham","Kanwal","Ramisha","Niazi"])

for i in names:

    {

        print(i+" is my friend.")

    }

# OUTPUT

Text

Description automatically generated

## 3-3. Your Own List: Think of your favorite mode of transportation, such as a motorcycle or a car, and make a list that stores several examples. Use your list to print a series of statements about these items, such as “I would like to own a Honda motorcycle.”

# CODE

fav\_vehicles=["Audi A5", "Mercedes Benz C class","Harley bike","Hyundai Sonata"]

for i in range(0,len(fav\_vehicles)):

    {

        print("I have a dream to own "+fav\_vehicles[i])

    }

# OUTPUT

Text

Description automatically generated

## 3-4. Guest List: If you could invite anyone, living or deceased, to dinner, who would you invite? Make a list that includes at least three people you’d like to

## invite to dinner. Then use your list to print a message to each person, inviting

## them to dinner.

# CODE

invitation=["salman","shahab","kanwal"]

for i in invitation:

    {

        print(f"Hello {i} please come to my dinner party tonight!")

    }

# OUTPUT

Text

Description automatically generated

## 3-5. Changing Guest List: You just heard that one of your guests can’t make the dinner, so you need to send out a new set of invitations. You’ll have to think of

## someone else to invite.

## • Start with your program from Exercise 3-4. Add a print statement at the

## end of your program stating the name of the guest who can’t make it.

## • Modify your list, replacing the name of the guest who can’t make it with

## the name of the new person you are inviting.

## • Print a second set of invitation messages, one for each person who is still

## in your list.

# CODE

invitation=["salman","shahab","kanwal"]

for i in invitation:

    {print(f"Hello {i} please come to my dinner party tonight!")}

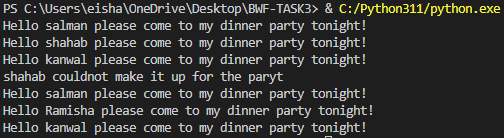
print(f"{invitation[1]} couldnot make it up for the paryt")

invitation[1]="Ramisha"

for i in invitation:

    {print(f"Hello {i} please come to my dinner party tonight!")}

# OUTPUT



## 3-6. More Guests: You just found a bigger dinner table, so now more space is

## available. Think of three more guests to invite to dinner.

## • Start with your program from Exercise 3-4 or Exercise 3-5. Add a print

## statement to the end of your program informing people that you found a

## bigger dinner table.

## • Use insert() to add one new guest to the beginning of your list.

## • Use insert() to add one new guest to the middle of your list.

## • Use append() to add one new guest to the end of your list.

## • Print a new set of invitation messages, one for each person in your list.

# CODE

import math

invitation=["salman","shahab","kanwal"]

for i in invitation:

    {print(f"Hello {i} please come to my dinner party tonight!")}

for i in invitation:

    {print(f"Hello {i} ,I have found a bigger table and now a total of 6 guests would be invited")}

middle= math.ceil((len(invitation)/2))

# adding at start of list

invitation.insert(0,"Ahmed")

#adding at middle of list

invitation.insert(middle,"Farva")

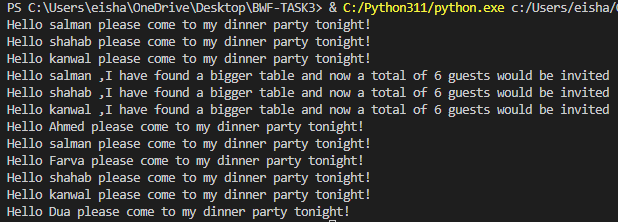
#adding at the end of list

invitation.append("Dua")

for i in invitation:

    {print(f"Hello {i} please come to my dinner party tonight!")}

# OUTPUT



## 3-7. Shrinking Guest List: You just found out that your new dinner table won’t

## arrive in time for the dinner, and you have space for only two guests.

## • Start with your program from Exercise 3-6. Add a new line that prints a

## message saying that you can invite only two people for dinner.

## • Use pop() to remove guests from your list one at a time until only two

## names remain in your list. Each time you pop a name from your list, print

## a message to that person letting them know you’re sorry you can’t invite

## them to dinner.

## • Print a message to each of the two people still on your list, letting them

## know they’re still invited.

## • Use del to remove the last two names from your list, so you have an empty

## list. Print your list to make sure you actually have an empty list at the end

## of your program.

# CODE

import math

invitation=["salman","shahab","kanwal"]

middle= math.ceil((len(invitation)/2))

# adding at start of list

invitation.insert(0,"Ahmed")

#adding at middle of list

invitation.insert(middle,"Farva")

#adding at the end of list

invitation.insert(len(invitation),"Dua")

for i in invitation:

    {print(f"Hello {i} I just found I can invite only 2 people to the part")}

while len(invitation)!=2:

    invitation.pop()

    i=len(invitation)-1

    print(f"Sorry {invitation[i]} I cannot invite you to the party due to limited space")

    i-=1

for i in invitation:

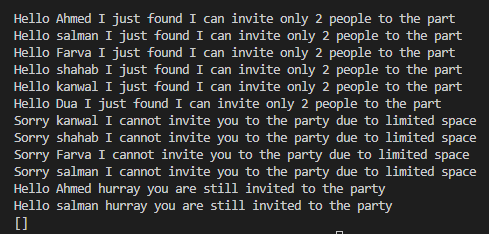
    print(f"Hello {i} hurray you are still invited to the party")

del invitation[0]

del invitation[0]

print(invitation)

# OUTPUT SCREENSHOT



## 3-8. Seeing the World: Think of at least five places in the world you’d like to

## visit.

## • Store the locations in a list. Make sure the list is not in alphabetical order.

## • Print your list in its original order. Don’t worry about printing the list neatly,

## just print it as a raw Python list.

## • Use sorted() to print your list in alphabetical order without modifying the

## actual list.

## • Show that your list is still in its original order by printing it.

## • Use sorted() to print your list in reverse alphabetical order without chang-

## ing the order of the original list.

## • Show that your list is still in its original order by printing it again.

## • Use reverse() to change the order of your list. Print the list to show that its

## order has changed.

## • Use reverse() to change the order of your list again. Print the list to show

## it’s back to its original order.

## • Use sort() to change your list so it’s stored in alphabetical order. Print the

## list to show that its order has been changed.

## • Use sort() to change your list so it’s stored in reverse alphabetical order.

## Print the list to show that its order has changed.

# CODE

places=["Switzerland","Italy","London","Iceland","Germany"]

print(f"Original list:{places}")

print(f"Sorted alphabetically:{sorted(places)}")

print(f"Original list:{places}")

print(f"Sorted reverse alphabetically:{sorted(places,reverse=True)}")

print(f"Original list:{places}")

places.reverse()

print(f"Using reverse() on list, the list becomes:{places}")

places.reverse()

print(f"Using reverse() again on list, the list comes in original order:{places}")

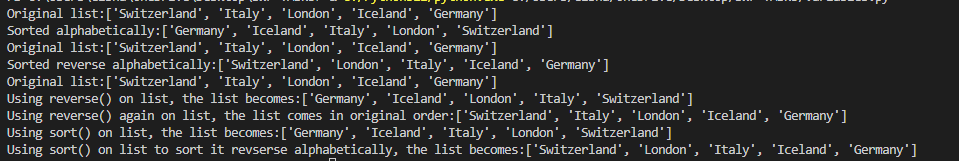
places.sort()

print(f"Using sort() on list, the list becomes:{places}")

places.sort(reverse=True)

print(f"Using sort() on list to sort it revserse alphabetically, the list becomes:{places}")

# OUTPUT SCREENSHOT



## 3-9. Dinner Guests: Working with one of the programs from Exercises 3-4

## through 3-7 (page 46), use len() to print a message indicating the number

## of people you are inviting to dinner.

# CODE

invitation=["salman","shahab","kanwal"]

for i in invitation:

    {

        print(f"Hello {i} please come to my dinner party tonight!")

    }

print(f"I am inviting {len(invitation)} people to the party.")

# OUTPUT SCREENSHOT

Text

Description automatically generated

## 3-10. Every Function: Think of something you could store in a list. For example,

## you could make a list of mountains, rivers, countries, cities, languages, or any-

## thing else you’d like. Write a program that creates a list containing these items

## and then uses each function introduced in this chapter at least once.

# CODE

flowers=["chrysanthemum","roses","sunflowers","babybreaths","orchids"]

#adding on end of list

flowers.append("lillies")

print(f"Adding on end of list:{flowers}")

#inserting at position 2 of list

flowers.insert(2,"tulips")

print(f"Inserting at position 2 of list:{flowers}")

#popping at the end

flowers.pop(1)

print(f"Popping from position 1 of list:{flowers}")

#deleting from position 0

del flowers[0]

print(f"Deleting from position 0 of list:{flowers}")

#removing an item of list by its name

flowers.remove('sunflowers')

print(f"removing an item of list by its name:{flowers}")

#sorting the list without modifying original

print(f"sorting the list without modifying original:{sorted(flowers)}")

#sorting reverse alphabetically the list without modifying original

print(f"sorting reverse alphabetically the list without modifying original:{sorted(flowers,reverse=True)}")

# sorting the list without by modifying original

flowers.sort()

print(f"sorting the list without by modifying original:{flowers}")

# sorting reverse alphabetically the list by modifying original

flowers.sort(reverse=True)

print(f"sorting reverse alphabetically the list by modifying original:{flowers}")

#printing length of list

print(f"The list has {len(flowers)} items.")

# OUTPUT SCREENSHOT

Text

Description automatically generated

## 3-11. Intentional Error: If you haven’t received an index error in one of your programs yet, try to make one happen. Change an index in one of your programs to produce an index error. Make sure you correct the error before closing the program.

# Index error CODE

flowers=["chrysanthemum","roses","sunflowers","babybreaths","orchids"]

print(flowers[5])

# OUTPUT

Text

Description automatically generated

# Corrected CODE

flowers=["chrysanthemum","roses","sunflowers","babybreaths","orchids"]

print(flowers[4])

# OUTPUT



**CHAPTER 4-Working With Lists**

## 4-1. Pizzas: Think of at least three kinds of your favorite pizza. Store these pizza names in a list, and then use a for loop to print the name of each pizza.

## • Modify your for loop to print a sentence using the name of the pizza instead of printing just the name of the pizza. For each pizza you should have one line of output containing a simple statement like I like pepperoni

## pizza.

## • Add a line at the end of your program, outside the for loop, that states how much you like pizza. The output should consist of three or more lines about the kinds of pizza you like and then an additional sentence, such as I really love pizza!

# CODE

pizza=["fajita pizza","bbq pzza","pepporoni pizza"]

for i in pizza:

    print(i)

for i in pizza:

    print("I really love "+i)

print(f'My most favourite pizzas are:\n1-{pizza[0]}\n2-{pizza[1]}\n3-{pizza[2]}\nI really love pizza')

# OUTPUT

Text

Description automatically generated

## 4-2. Animals: Think of at least three different animals that have a common characteristic. Store the names of these animals in a list, and then use a for loop to print out the name of each animal.

## • Modify your program to print a statement about each animal, such as

## A dog would make a great pet.

## • Add a line at the end of your program stating what these animals have in

## common. You could print a sentence such as Any of these animals would

## make a great pet!

# CODE

animals=["crow","parrot","sparrow"]

for i in animals:

    print(i)

print(animals[0]+" mostly has black color")

print(animals[1]+" is a great pet")

print(animals[2]+" is very beautiful")

print("All of these animals have common characteristic and that is that they can fly")

# OUTPUT

Text

Description automatically generated

## 4-3. Counting to Twenty: Use a for loop to print the numbers from 1 to 20,

## inclusive.

# CODE

for i in range(1,21):

    print(i)

# OUTPUT

Graphical user interface, application, Teams

Description automatically generated

## 4-4. One Million: Make a list of the numbers from one to one million, and then

## use a for loop to print the numbers. (If the output is taking too long, stop it by

## pressing ctrl-C or by closing the output window.)

# CODE

for i in range(1, 1000001):

    print(i)

## 4-5. Summing a Million: Make a list of the numbers from one to one million,

## and then use min() and max() to make sure your list actually starts at one and

## ends at one million. Also, use the sum() function to see how quickly Python can

## add a million numbers.

# CODE

num\_list = list(range(1, 1000001))

print(min(num\_list))

print(max(num\_list))

print(sum(num\_list))

# OUTPUT



## 4-6. Odd Numbers: Use the third argument of the range() function to make a list

## of the odd numbers from 1 to 20. Use a for loop to print each number.

# CODE

for i in range(1, 20, 2):

    print(i)

# OUTPUT

Graphical user interface

Description automatically generated

## 4-7. Threes: Make a list of the multiples of 3 from 3 to 30. Use a for loop to

## print the numbers in your list.

# CODE

for i in range(3, 33, 3):

    print(i)

# OUTPUT



## 4-8. Cubes: A number raised to the third power is called a cube. For example,

## the cube of 2 is written as 2\*\*3 in Python. Make a list of the first 10 cubes (that

## is, the cube of each integer from 1 through 10), and use a for loop to print out

## the value of each cube.

# CODE

for i in range(1,11):

    print(f"The cube of {i} is {i\*\*3}")

# OUTPUT

A screenshot of a computer

Description automatically generated with medium confidence

## 4-9. Cube Comprehension: Use a list comprehension to generate a list of the

## first 10 cubes.

# CODE

cubes=[num\*\*3 for num in range(1,11)]

print(cubes)

# OUTPUT



## 4-10. Slices: Using one of the programs you wrote in this chapter, add several

## lines to the end of the program that do the following:

## • Print the message, The first three items in the list are:. Then use a slice to

## print the first three items from that program’s list.

## • Print the message, Three items from the middle of the list are:. Use a slice

## to print three items from the middle of the list.

## • Print the message, The last three items in the list are:. Use a slice to print

## the last three items in the list.

# CODE

cubes=[num\*\*3 for num in range(1,11)]

print(cubes)

print(f"The first three items of the list are:{cubes[0:3]}")

print(f"The middle three items of the list are:{cubes[3:6]}")

print(f"The last three items of the list are:{cubes[7:]}")

# OUTPUT

Text

Description automatically generated

## 4-11. My Pizzas, Your Pizzas: Start with your program from Exercise 4-1

## (page 60). Make a copy of the list of pizzas, and call it friend\_pizzas.

## Then, do the following:

## • Add a new pizza to the original list.

## • Add a different pizza to the list friend\_pizzas.

## • Prove that you have two separate lists. Print the message, My favorite

## pizzas are:, and then use a for loop to print the first list. Print the message,

## My friend’s favorite pizzas are:, and then use a for loop to print the second list. Make sure each new pizza is stored in the appropriate list.

# CODE

pizza=["fajita pizza","bbq pzza","pepporoni pizza"]

friend\_pizza=pizza[:]

pizza.append("margarittha pizza")

friend\_pizza.append("Veggie pizza")

print("My favourite pizzas are:")

for i in pizza:

    print("-"+i)

print("My friends's favourite pizzas are:")

for i in friend\_pizza:

    print("-"+i)

# OUTPUT

Text

Description automatically generated

## 4-13. Buffet: A buffet-style restaurant offers only five basic foods. Think of five

## simple foods, and store them in a tuple.

## • Use a for loop to print each food the restaurant offers.

## • Try to modify one of the items, and make sure that Python rejects the

## change.

## • The restaurant changes its menu, replacing two of the items with different

## foods. Add a block of code that rewrites the tuple, and then use a for

## loop to print each of the items on the revised menu.

# CODE

food = (

    "burgers","fish n chips","alfredo pasta","tarragon steak","cheesy pasta"

    )

print("The name of the foods are:")

for i in food:

    print("- " + i)

food = (

    "burgers","fish n chips","alfredo pasta","chinese rice","noodles"

    )

print("The revised menu is:")

for i in food:

    print("- " + i)

# OUTPUT

Text

Description automatically generated

## 4-15. Code Review: Choose three of the programs you’ve written in this chapter

## and modify each one to comply with PEP 8:

## • Use four spaces for each indentation level. Set your text editor to insert

## four spaces every time you press tab, if you haven’t already done so (see

## Appendix B for instructions on how to do this).

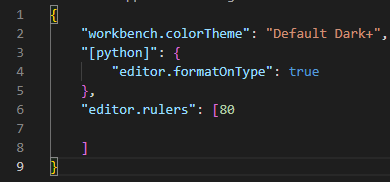
## • Use less than 80 characters on each line, and set your editor to show a

## vertical guideline at the 80th character position.

## • Don’t use blank lines excessively in your program files.

# ANSWER

* Text

  Description automatically generated
* 
* Text

  Description automatically generated with medium confidence