```
#Let’s use a for loop to print out each name in a list of magicians:


magicians = ['alice', 'david', 'carolina']
```

```
for magician in magicians:
  print(magician)
```

```
alice
david
carolina
```

```
for magician in magicians:
  print(f'{magician.title()},That was great trick!')
```

```
Alice,That was great trick!
David,That was great trick!
Carolina,That was great trick!
```

```
for magician in magicians:
  print(f'{magician.title()},That was great trick!')
  print(f"I can't wait to see your next trick, {magician.title()}.\n")
```

```
Alice,That was great trick!
I can't wait to see your next trick, Alice.

David,That was great trick!
I can't wait to see your next trick, David.

Carolina,That was great trick!
I can't wait to see your next trick, Carolina.

```

```
 for magician in magicians:
  print(f'{magician.title()},That was great trick!')
  print(f"I can't wait to see your next trick, {magician.title()}.\n")

 print("Thank you, everyone. That was a great magic show!")
```

```
Alice,That was great trick!
I can't wait to see your next trick, Alice.

David,That was great trick!
I can't wait to see your next trick, David.

Carolina,That was great trick!
I can't wait to see your next trick, Carolina.

Thank you, everyone. That was a great magic show!
```

```
#LIST

#Example :

pizza=  ['pizza1','pizza2','pizza3']
```

```
for pizzas in pizza:
  print(f"{pizzas.title()},I like pepperonipizza!")
  print(f'What kind of the pizza u like,{pizzas.title()}.')

print("i love the pizza.........")
```

```
Pizza1,I like pepperonipizza!
What kind of the pizza u like,Pizza1.
Pizza2,I like pepperonipizza!
What kind of the pizza u like,Pizza2.
Pizza3,I like pepperonipizza!
What kind of the pizza u like,Pizza3.
i love the pizza.........
```

```
#for the value or integer

#using the range function

for i in range(1,5):
  print(i)
```

```
1
2
3
4
```

```
#Using range() to Make a List of Numbers
#normal
num=list(range(0,5))
print(num)

#for the even number
num=list(range(0,5,2))
print(num)



#for the odd number
num=list(range(0,5,1))
print(num)
```

```
[0, 1, 2, 3, 4]
```

```
[0, 2, 4]
[0, 1, 2, 3, 4]
```

```
#square method

squares=[]

for i in range(1,5):
  square= i**2
  squares.append(square)

print(squares)
```

```
[1, 4, 9, 16]
```

```
#we can write same code in this ways

squares = []

for i in range(1,11):
  squares.append(i**2)

print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
#list of the number
digits = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
```

```
print(max(digits))

print(min(digits))


sum(digits)
```

```
9
0
45
```

The approach described earlier for generating the list squares consisted ofusing three or four lines of code.

#A list comprehension allows you to generate this same list in just one line of code.

A list comprehension combines the for loop and the creation of new elements into one line, and automatically appends each new element.

List comprehensions are not always presented to beginners, but I have included them here because you'll most likely see them as soon as you start looking at other people's code.

```
#List Comprehensions
squares = [i**2 for i in range(1, 11)]
print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
# Q1=Counting to Twenty: Use a for loop to print the numbers from 1 to 20,

# Q2=One Million: Make a list of the numbers from one to one million, and then use a for lo
#pressing ctrl-C or by closing the output window.)

#Q3=Make a list of the numbers from one to one million,and then use min() and max() to make
#ends at one million. Also, use the sum() function to see how quickly Python canadd a milli

#Q4:Use the third argument of the range() function to make a list of the odd numbers from 1

#Q5:Make a list of the multiples of 3 from 3 to 30. Use a for loop to print the numbers in

#Q6: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, th

#Q6:Use a list comprehension to generate a list of the first 10 cubes.
```

```
#Q1

for i in range(1,20):
  print(i)
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

```
14
15
16
17
18
19
```

```
#for i in range(1,1000000):
  #print(i)
```

```
#q5
num=[]

for i in range(0,11):
  number=  i**3
  num.append(number)
print(num)
```

```
[0, 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

```
#q6 comprensive list

cube= [i**3 for i in range(0,10)]
print(cube)
```

```
[0, 1, 8, 27, 64, 125, 216, 343, 512, 729]
```

```
#Looping Through a Slice

players = ['charles', 'martina', 'michael', 'florence', 'eli']
```

```
print("Here the list of the player")
for player in players[:4]:
  print(player.title())
```

```
Here the list of the player
Charles
Martina
Michael
Florence
```

```
#Copying a List:you’ll want to start with an existing list and make an entirely new list  b
my_foods = ['pizza', 'falafel', 'carrot cake']
friend_food=my_foods[:]
```

```
print("My favorite food are :")
print(my_foods)



print("My friend favorite food are:")
print(friend_food)
```

```
My favorite food are :
['pizza', 'falafel', 'carrot cake']
My friend favorite food are:
['pizza', 'falafel', 'carrot cake']
```

```
#add the new string

my_foods.append("mango")



friend_food.append("ice cream")
```

```
print("My favorite food are :")
print(my_foods)



print("My friend favorite food are:")
print(friend_food)
```

```
My favorite food are :
['pizza', 'falafel', 'carrot cake', 'mango']
My friend favorite food are:
['pizza', 'falafel', 'carrot cake', 'ice cream']
```

```
#if -else case

cars = ['audi', 'bmw', 'subaru', 'toyota']
for car in cars:
 if car == 'bmw':
  print(car.upper())
 else:
  print(car.title())
```

```
Audi
BMW
Subaru
Toyota
```

```
#checking the iniquality
requested_topping = 'mushrooms'



if requested_topping != 'anchovies':
  print("Hold the anchovies!")
```

```
Hold the anchovies!
```

```
ans =17

if ans !=20:
  print("That is not correct. Please try again later")
```

```
that is not correct. please try again later
```

```
a =22
b=18

print(a>=21 and b>=21)
print(a>=21 or b>=21)
```

```
False
True
```

```
list = ['mushrooms', 'onions', 'pineapple']
```

```
'mushrooms' in list
```

```
True
```

```
'satyam' in  list
```

```
False
```

```
#using the banned list of the customer

blist = ['Dinesh','Sarvesh','Suraj']
user='Satyam'
```

```
if user not in  blist:
  print(f"{user.title()}, you can post a response if you wish.")
```

```
Satyam, you can post a response if you wish.
```

```
requested_toppings = ['mushrooms', 'green peppers', 'extra cheese']


for requested topping in requested toppings:
 print(f"Adding {requested_topping}.")
print("\nFinished making your pizza!")
```

```
Adding mushrooms.
Adding green peppers.
Adding extra cheese.

Finished making your pizza!
```

```
for requested_topping in requested_toppings:
 if requested_topping == 'green peppers':
  print("Sorry, we are out of green peppers right now.")
else:
 print(f"Adding {requested_topping}.")
print("\nFinished making your pizza!")
```

```
Sorry, we are out of green peppers right now.
Adding extra cheese.

Finished making your pizza!
```

```
#Checking That a List Is Not Empty

requested_toppings = []
```

```
if requested_toppings:
 for requested_topping in requested_toppings:
  print(f"Adding {requested_topping}.")
  print("\nFinished making your pizza!")
else:
 print("Are you sure you want a plain pizza?")
```

```
Are you sure you want a plain pizza?
```

```
#multiple list

available_toppings = ['mushrooms', 'olives', 'green peppers','pepperoni', 'pineapple', 'ext
requested_toppings = ['mushrooms', 'french fries', 'extra cheese']
```

```
for requested_topping in requested_toppings:
  if requested_topping in available_toppings:
    print(f"adding {requested_topping}.")
  else:
    print(f"Sorry,we dont have {requested_topping}.")
print("Finish making pizza ")
```

```
adding mushrooms.
Sorry,we dont have french fries.
```

```
adding extra cheese.
Finish making pizza
```

```
#dictionary

#key and value



favorite_languages = {'jen': 'python','sarah': 'c','edward': 'ruby','phil': 'python'}
```

```
for key, value in favorite_languages.items():
 print(f"\nKey: {key}")
 print(f"Value: {value}")
```

```

Key: jen
Value: python

Key: sarah
Value: c

Key: edward
Value: ruby

Key: phil
Value: python
```

```
for name, language in favorite_languages.items():
 print(f"{name.title()}'s favorite language is {language.title()}.")
```

```
Jen's favorite language is Python.
Sarah's favorite language is C.
Edward's favorite language is Ruby.
Phil's favorite language is Python.
```

```
#Looping Through All the Keys in a Dictionary

for name in favorite_languages.keys():
  print(name.title())
```

```
Jen
Sarah
Edward
Phil
```

```
#all the value in dictonary
for language in favorite_languages.values():
  print(language.title())
```

```
Python
C
Ruby
Python
```

```
 friends = ['phil', 'sarah']
for name in favorite_languages.keys():
  print(name.title())

 if name in friends:
  language = favorite_languages[name].title()
  print(f"\t{name.title()}, I see you love {language}!")
```

```
Jen
Sarah
Edward
Phil
        Phil, I see you love Python!
```

```
if 'erin' not in favorite_languages:
  print("Erin, Please take our poll")
```

```
Erin, Please take our poll
```

```
#Looping Through a Dictionary’s Keys in a Particular Order

favorite_languages
```

```
{'edward': 'ruby', 'jen': 'python', 'phil': 'python', 'sarah': 'c'}
```

```
for name in sorted(favorite_languages.keys()):
  print(f'{name.title()},Thank you for the taking poll.')
```

```
Edward,Thank you for the taking poll.
Jen,Thank you for the taking poll.
Phil,Thank you for the taking poll.
Sarah,Thank you for the taking poll.
```

```
#Nesting: we can do for all list and tuple and dictionary

alien_0 = {'color': 'green', 'points': 5}
alien_1 = {'color': 'yellow', 'points': 10}
alien_2 = {'color': 'red', 'points': 15}
```

```
#merge all list

aliens = [alien_0, alien_1, alien_2]
```

```
for alien in aliens:
 print(alien)
```

```
{'color': 'green', 'points': 5}
{'color': 'yellow', 'points': 10}
{'color': 'red', 'points': 15}
```

```
#use range() to create a fleet of 30 aliens



# Make an empty list for storing aliens.


aliens = []

# Make 30 green aliens.
for alien_number in range(30):
  new_alien = {'color': 'green', 'points': 5, 'speed': 'slow'}
  aliens.append(new_alien)
```

```
# Show the first 5 aliens.
for alien in aliens[:5]:
 print(alien)
print("...")
```

```
{'color': 'green', 'points': 5, 'speed': 'slow'}
{'color': 'green', 'points': 5, 'speed': 'slow'}
{'color': 'green', 'points': 5, 'speed': 'slow'}
{'color': 'green', 'points': 5, 'speed': 'slow'}
{'color': 'green', 'points': 5, 'speed': 'slow'}
...
```

```
# Show how many aliens have been created.
print(f"Total number of aliens: {len(aliens)}")
```

```
Total number of aliens: 30
```

```
for alien in aliens[:3]:
 if alien['color'] == 'green':
  alien['color'] = 'yellow'
  alien['speed'] = 'medium'
  alien['points'] = 10
```

These aliens all have the same characteristics, but Python considers each one a separate object, which allows us to modify each alien individually. How might you work with a group of aliens like this? Imagine that one aspect of a game has some aliens changing color and moving faster as the game progresses. When it’s time to change colors, we can use a for loop and an if statement to change the color of aliens. For example, to change the first three aliens to yellow, medium-speed aliens worth 10 points each, we could do this:

```
# Show the first 5 aliens.
for alien in aliens[:5]:
 print(alien)
print("...")
```

```
{'color': 'yellow', 'points': 10, 'speed': 'medium'}
{'color': 'yellow', 'points': 10, 'speed': 'medium'}
{'color': 'yellow', 'points': 10, 'speed': 'medium'}
{'color': 'green', 'points': 5, 'speed': 'slow'}
{'color': 'green', 'points': 5, 'speed': 'slow'}
...
```

You could expand this loop by adding an elif block that turns yellow aliens into red, fast-moving ones worth 15 points each. Without showing the entire program again, that loop would look like this:

```
for alien in aliens[0:3]:
 if alien['color'] == 'green':
  alien['color'] = 'yellow'
  alien['speed'] = 'medium'
  alien['points'] = 10
 elif alien['color'] == 'yellow':
  alien['color'] = 'red'
  alien['speed'] = 'fast'
  alien['points'] = 15
```

```
for alien in aliens[:5]:
 print(alien)
print("...")
```

```
{'color': 'red', 'points': 15, 'speed': 'fast'}
{'color': 'red', 'points': 15, 'speed': 'fast'}
{'color': 'red', 'points': 15, 'speed': 'fast'}
{'color': 'green', 'points': 5, 'speed': 'slow'}
{'color': 'green', 'points': 5, 'speed': 'slow'}
...
```

```
#A List in a Dictionary
```

```
## Store information about a pizza being ordered.
pizza = {'crust': 'thick','toppings': ['mushrooms', 'extra cheese']}
```

Rather than putting a dictionary inside a list, it's sometimes useful to put a list inside a dictionary. For example, consider how you might describe a pizza that someone is ordering. If you were to use only a list, all you could really store is a list of the pizza's toppings. With a dictionary, a list of toppings can be just one aspect of the pizza you're describing. In the following example, two kinds of information are stored for each pizza: a type of crust and a list of toppings. The list of toppings is a value associated with the key 'toppings'. To use the items in the list, we give the name of the dictionary and the key 'toppings', as we would any value in the dictionary. Instead of returning a single value, we get a list of toppings:

```
# Summarize the order.
print(f"You ordered a {pizza['crust']}-crust pizza "
 "with the following toppings:")
```

```
You ordered a thick-crust pizza with the following toppings:
```

```
for topping in pizza['toppings']:
 print("\t" + topping)
```

```
        mushrooms
        extra cheese
```

```
users = {'aeinstein': {'first': 'albert','last': 'einstein','location': 'princeton',},'mcur
```

We first define a dictionary called users with two keys: one each for the usernames 'aeinstein' and 'mcurie'. The value associated with each key is a dictionary that includes each user's first name, last name, and location. At u we loop through the users dictionary. Python assigns each key to the variable username, and the dictionary associated with each username is assigned to the variable user_info. Once inside the main dictionary loop, we print the username at v. At w we start accessing the inner dictionary. The variable user_info, which contains the dictionary of user information, has three keys: 'first', 'last', and 'location'. We use each key to generate a neatly formatted full name and location for each person, and then print a summary of what we know about each user x:

```
for username, user_info in users.items():
  print(f"\nUsername: {username}")
  full_name = f"{user_info['first']} {user_info['last']}"
  location = user_info['location']
  print(f"\tFull name: {full_name.title()}")
  print(f"\tLocation: {location.title()}")
```

```
Username: aeinstein
        Full name: Albert Einstein
        Location: Princeton

Username: mcurie
        Full name: Marie Curie
        Location: Paris
```

WHILE LOOP

Input

```
#How the input() Function Works
message = input("Tell me something, and I will repeat it back to you: ")
print(message)
```

```
Tell me something, and I will repeat it back to you: hellow everyone
hellow everyone
```

```
name = input("Please enter your name: ")
print(f"\nHello, {name}!")
```

```
Please enter your name: satyam

Hello, satyam!
```

```
age = input("how old you are :")
print("age : ",age)
```

```
how old you are :21
age :  21
```

```
height=input("What is your hight ?")
height=int(height)


print("\n Height :",height)
if height>48:
  print("\n You are tall enough to ride.")
else:
   print("\n You will able to ride when you will be little older")
```

```
What is your hight ?46

 Height : 46

 You will able to ride when you will be little older
```

```
#The Modulo Operator

4 % 3,5 % 3,6 % 3,7 % 3
```

```
(1, 2, 0, 1)
```

```
number = input("Enter a number, and I'll tell you if it's even or odd: ")
number = int(number)
if number % 2 == 0:
 print(f"\nThe number {number} is even.")
else:
 print(f"\nThe number {number} is odd.")
```

```
Enter a number, and I'll tell you if it's even or odd: 48

The number 48 is even.
```

7-1. Rental Car: Write a program that asks the user what kind of rental car they would like. Print a message about that car, such as “Let me see if I can find you a Subaru.”

7-2. Restaurant Seating: Write a program that asks the user how many people are in their dinner group. If the answer is more than eight, print a message saying they’ll have to wait for a table. Otherwise, report that their table is ready.

7-3. Multiples of Ten: Ask the user for a number, and then report whether the number is a multiple of 10 or not.

```
people =input("How many people are in Dinner Group")
people=int(people)

if people>8:
  print("\n Please wait for the table Otherwise.\nreport that their table is ready.")
else:
  print("\nPlease take Tabel.\nTable is Ready for the Dinner.")
```

```
How many people are in Dinner Group4

Please take Tabel.
Table is Ready for the Dinner.
```

Double-click (or enter) to edit

```
5
```

while

The for loop takes a collection of items and executes a block of code once for each item in the collection. In contrast, the while loop runs as long as, or while, a certain condition is true.

```
#You can use a while loop to count up through a series of numbers. For example, the followi
current_number = 0
while current_number <= 5:
 print("current num =",current_number*5)
 current_number += 1
```

```
current num = 0
current num = 5
current num = 10
current num = 15
current num = 20
current num = 25
```

Letting the User Choose When to Quit

```
prompt = "\nTell me something, and I will repeat it back to you:"
prompt += "\nEnter 'quit' to end the program. "
```

```
message = ""
while message != 'quit':
 message = input(prompt)
 print(message)
```

```
Tell me something, and I will repeat it back to you:
Enter 'quit' to end the program. e
e

Tell me something, and I will repeat it back to you:
Enter 'quit' to end the program. quit
quit
```

```
prompt = "\nTell me something, and I will repeat it back to you:"
prompt += "\nEnter 'quit' to end the program. "

message = ""
while message != 'quit':
 message = input(prompt)

 if message != 'quit':
  print(message)
```

```
   Tell me something, and I will repeat it back to you:
   Enter 'quit' to end the program. hellow
   hellow

   Tell me something, and I will repeat it back to you:
   Enter 'quit' to end the program. hi
   hi

   Tell me something, and I will repeat it back to you:
   Enter 'quit' to end the program. quit '
   quit '

   Tell me something, and I will repeat it back to you:
   Enter 'quit' to end the program. quit
```

Using a Flag

In the previous example, we had the program perform certain tasks while a given condition was true. But what about more complicated programs in which many different events could cause the program to stop running? For example, in a game, several different events can end the game. When the player runs out of ships, their time runs out, or the cities they were supposed to protect are all destroyed, the game should end. It needs to end if any one of these events happens. If many possible events might occur to stop the program, trying to test all these conditions in one while statement becomes complicated and difficult. For a program that should run only as long as many conditions are true, you can define one variable that determines whether or not the entire program is active. This variable, called a flag, acts as a signal to the program. We can write our programs so they run while the flag is set to True and stop running when any of several events sets the value of the flag to False. As a result, our overall while statement needs to check only one condition: whether or not the flag is currently True. Then, all our other tests (to see if an event has occurred that should set the flag to False) can be neatly organized in the rest of the program. Let's add a flag to parrot.py from the previous section. This flag, which we'll call active (though you can call it anything), will monitor whether or not the program should continue running:

```
prompt = "\nTell me something, and I will repeat it back to you:"
prompt = "\nTell me something, and I will repeat it back to you:
prompt += "\nEnter 'quit' to end the program. "
```

```
#flag

active=True
while active:
  message=input(prompt)
  if message =='quit':
    active=False
  else:
    print(message)
```

```
Tell me something, and I will repeat it back to you:
Enter 'quit' to end the program. hellow
hellow

Tell me something, and I will repeat it back to you:
Enter 'quit' to end the program. quit
```

```
#Using break to Exit a Loop

prompt="\n Please Enter The City You Visited"
prompt+="\n(Enter 'quit' when you are finished.) "

while True:
  city=input(prompt)
  if city == 'quit':
    break
  else:
    print(f"I'd love to go to {city.title()}!")
```

```
 Please Enter The City You Visited
(Enter 'quit' when you are finished.) New York
I'd love to go to New York!

 Please Enter The City You Visited
(Enter 'quit' when you are finished.) India
I'd love to go to India!

 Please Enter The City You Visited
(Enter 'quit' when you are finished.) quit
I'd love to go to Quit !

 Please Enter The City You Visited
(Enter 'quit' when you are finished.) quit
```

```
#Using continue in a Loop
```

Rather than breaking out of a loop entirely without executing the rest of its code, you can use the continue statement to return to the beginning of the loop based on the result of a conditional test. For example, consider a loop that counts from 1 to 10 but prints only the odd numbers in that range:

```
#if i need to print only the odd number

current_number =0
while current_number<10:
  current_number +=1
  if current_number % 2==0:
    continue
  print(current_number)
```

```
1
3
5
7
9
```

Movie Tickets: A movie theater charges different ticket prices depending on a person’s age. If a person is under the age of 3, the ticket is free; if they are

between 3 and 12, the ticket is $10; and if they are over age 12, the ticket is $15. Write a loop in which you ask users their age, and then tell them the cost of their movie ticket.

```
age =input("Please Enter the age of the Person")
age=int(age)

if age<3 :
  print("The Ticket is free!")

elif age <=12:
  print("The Ticket price is : $10")

elif age>12:
  print('The ticket Price is : $15')
```

```
Please Enter the age of the Person12
The Ticket price is : $10
```

```
#Using a while Loop with Lists and Dictionaries
```

So far, we've worked with only one piece of user information at a time. We received the user's input and then printed the input or a response to it. The next time through the while loop, we'd receive another input value and respond to that. But to keep track of many users and pieces of information, we'll need to use lists and dictionaries with our while loops. A for loop is effective for looping through a list, but you shouldn't modify a list inside a for loop because Python will have trouble keeping track of the items in the list. To modify a list as you work through it, use a while loop. Using while loops with lists and dictionaries allows you to collect, store, and organize lots of input to examine and report on later.

Moving Items from One List to Another

Consider a list of newly registered but unverified users of a website. After we verify these users, how can we move them to a separate list of confirmed users? One way would be to use a while loop to pull users from the list of unconfirmed users as we verify them and then add them to a separate list of confirmed users. Here's what that code might look like:

```
# Start with users that need to be verified,
#users.py # and an empty list to hold confirmed users.
unconfirmed_users = ['alice', 'brian', 'candace']
confirmed_users = []

#verify the user until the unconfirmed user no more

#Move the verify user the to the confirmed_user list

while unconfirmed_users:
  current_user=unconfirmed_users.pop()

  print(f"Verifying User :{current_user.title()}")
  confirmed_users.append(current_user)
```

```
Verifying User :Candace
Verifying User :Brian
Verifying User :Alice
```

```
# Display all confirmed users.
```

```
print("\nThe following users have been confirmed:")
for confirmed_user in confirmed_users:
 print(confirmed_user.title())
```

```
The following users have been confirmed:
Candace
Brian
Alice
```

```
#@Removing All Instances of Specific Values from a List

pets = ['dog', 'cat', 'dog', 'goldfish', 'cat', 'rabbit', 'cat']

print(pets)

while 'cat' in pets:
 pets.remove('cat')

print(pets)
```

```
['dog', 'cat', 'dog', 'goldfish', 'cat', 'rabbit', 'cat']
['dog', 'dog', 'goldfish', 'rabbit']
```

```
responses = {}


# Set a flag to indicate that polling is active.
polling_active = True

while polling_active:
 # Prompt for the person's name and response.
 name = input("\nWhat is your name? ")
 response = input("Which mountain would you like to climb someday? ")


 #Store the response in the dictionary.
 responses[name] = response

 # Find out if anyone else is going to take the poll.
 repeat = input("Would you like to let another person respond? (yes/ no / Yes / No) ")
 if repeat == 'no' or "No":
  polling_active = False




# Polling is complete. Show the results.
print("\n--- Poll Results ---")

for name, response in responses.items():
    print(f"{name} would like to climb {response}.")
```

```
What is your name? satyam
Which mountain would you like to climb someday? everest
Would you like to let another person respond? (yes/ no / Yes / No) No

--- Poll Results ---
satyam would like to climb everest.
```