TASK 17 The emount of energey required to increase the temprature of one gram of a material by one degree Celsius is the material's specific heat capacity, C. The total amount of energy, q, required to raise m grams of a material by (delta t) degress Celsius can be computed using the formula q= mCdelta t Write a program that reads the mass of some water and the temprature change from the user. Your program should display the total amount of energy that must be added or removed to achieve the desired temprature change

In [13]:
```
WATER_HEAT_CAPACITY = 4.186
ELECTRECITY_PRICE = 8.9
J_TO_KWH = 2.777*10**-7
volume = float(input("Amount to water in milliliters: "))
d_temp = float(input("Temperature increase (degrees Celsius): "))

# FORMULA FOR TOTAL AMOUNT OF ENERGY: q= mCdelta t.   MILLILITER IS EQUAL TO GRAM I
# GRAM HAS BEEN USED IN THIS CODE


q = volume * d_temp * WATER_HEAT_CAPACITY

print("That will require %d Joules of energy." %q)
# compute the cost
kwh = q * J_TO_KWH
cost = kwh * ELECTRECITY_PRICE
print("The much energy will cost %.2f cents." % cost)
```

```
Amount to water in milliliters: 121
Temperature increase (degrees Celsius): 12
That will require 6078 Joules of energy.
The much energy will cost 0.02 cents.
```

Task 18

The volume of a cylinder can be computed by multiplying the eara of its circular base by its height. Write a program that reads the radius of the cylinder, along with its height, from the user and computes its volume. Display the result to one decimal place

In [12]:
```
PI = 3.14159246

r = int(input("Enter the radius of cylinder:  "))
erea = (PI * r**2)
print(f"The erea of the base is : {erea}")

h = int(input("Enter the height of cylinder: "))
volume = (2*erea * h)
print(f"The volume of cylinder is {volume}" )
```

```
Enter the radius of cylinder:  2
The erea of the base is : 12.56636984
Enter the height of cylinder: 2
The volume of cylinder is 50.26547936
```

In [10]:
```
PI = 3.14159246

r = int(input("Enter the radius of cylinder:  "))
erea = (PI * r**2)
print("The erea of the base is :erea %.2f"%erea)

h = int(input("Enter the height of cylinder: "))
```

```
volume = (2*erea * h)
print("The volume of cylinder is volume % .2f"%volume )
```

```
Enter the radius of cylinder:  2
The erea of the base is :erea 12.57
Enter the height of cylinder: 2
The volume of cylinder is volume  50.27
```

Task 19 Creat a program that determines how quickly an object is travelling when it hits the ground. The user will enter the height from which the object is dropped its initial speed is 0 m/s. Assume that acceleration due to gravity is 9.8m/sec**2. You can use the formula vf = seq(vi**2 + 2ad) to compute the final speed vf, when the initial speed vi, acceleration, a, and distance , d, are known.

In [16]:
```
# Compute the speed of an object when it hits the ground.    FORMULA:  vf = seq(vi*
from math import sqrt
Gravity = 9.8
d = float(input("Height (in meters): "))
# Vi is zero then its not neccessary to included in
vf = sqrt(2*Gravity *d )
print("It will hit the ground at %.2f m/s."%vf)
```

```
Height (in meters): 18
It will hit the ground at 18.78 m/s.
```

Task 20 The ideal gas law is a mathematical approximation of the behavior of gasses as pressure, volume and temprature change. It is usually stated as: PV = nRT where P is the pressure in Pascals, V is the volume in liters, n is the amount of substance in moles, R is the ideal gas constant, equal to 8.314 J/ mol Kalvin , and T is the temprature in degrees Kalvin.

Write a program that compute the amount of gas in moles when the user supplies the pressure, volume and temprature. Test your program by determining the number of moles of gas in a SCUBA tank. A typical SCUBA tank holds 12 liters of gas at a pressure of 20,000,000 Pascals(approximately 3,000 PSI). Room temprature is approximately 20 degree Celsius or 68 degree Fahrenheit.

In [ ]:
```
```

In [19]:
```
# Define the ideal gas constant
R = 8.314 # J/mol K

# Get input from the user
P = float(input("Enter the pressure in Pascals: "))
V = float(input("Enter the volume in liters: "))
T = float(input("Enter the temperature in degrees Kelvin: "))

# Calculate the number of moles of gas
n = (P * V) / (R * T)

# Print the result
print("The amount of gas in moles is:", n)
```

```
Enter the pressure in Pascals: 20000000
Enter the volume in liters: 12
Enter the temperature in degrees Kelvin: 293.75
The amount of gas in moles is: 98270.54084625265
```

Task 21

The area of a tringle can be camputed using (area = b*h/2)where b stands as length of base and h stands as height. Write a program that allows the user to enter values for b and h. The progaram shold then compute and display the area of traingle with base and height

In [21]:
```
b = float(input("Enter the length of base of triangel in centimere: "))
h = float(input("Enter the height of a triangle in centimeter: "))
area = (b*h)/2
print("The erea of a triangle is %.2f centimeter square"%area)
```

```
Enter the length of base of triangel in centimere: 2
Enter the height of a triangle in centimeter: 12
The erea of a triangle is 12.00 centimeter square
```

Task 22 Compute the area of a triangle by using three sides such as s1, s2, s3. Using this formula: area= sqrt(s(s-s1)*(s-s2)*(s-s3) where s stands as: s=(s1 + s2 + s3)/2

In [28]:
```
from math import sqrt
s1 = int(input("Enter the first side of triangle: "))
s2 = int(input("Enter the second side of triangle: "))
s3 = int(input("Enter the third side of triangle: "))
s = float((s1+s2+s3)/2)
area = float(sqrt(s*(s-s1)*(s-s2)*(s-s3)))
print("The area of a triangle is: %.2f"%area)
```

```
Enter the first side of triangle: 12
Enter the second side of triangle: 13
Enter the third side of triangle: 10
The area of a triangle is: 57.00
```

Task 23 (Area of a Regular Ploygon) write a program to calculate the area of regular polygon with the following formula: area = (n*s**2) / 4* tan(pi/n)

In [31]:
```
from math import tan, pi


s = int(input("Enter the length of one side of regular polygon."))
n = int(input("Enter the number of sides in polygon"))
area = (n*s**2) / 4* tan(pi/n)
print("The area of Polygon is: %.2f"%area)
```

```
Enter the length of one side of regular polygon.12
Enter the number of sides in polygon10
The area of Polygon is: 116.97
```

Task 24-25 Units of time Creat a program that reads a duration form the user as a number of days, hours, minutes and seconds.Compute and display the total number of seconds represented by this duration.

In [37]:
```
D = int(input("Enter the number of days: "))
H = int(input("Enter the number of hours: "))
M = int(input("Enter the number of minutes: "))
S = int(input("Enter the number of second: "))
M_S = 60
H_S = 3600
D_S = 86400
total_duration =S+ (M_S)*M+(H_S)*3600 +(D_S)*86400
print(f"The total time in second is {total_duration} seconds ")
```

```
Enter the number of days: 2
Enter the number of hours: 10
Enter the number of minutes: 33
Enter the number of second: 10
The total time in second is 7477921990 seconds
```

Task 26: Python's time module includes several time-related functinos. One of these is hte asctime function which reads the current time from the computer's internal clock and returns it in a human-readable formate. Use this function to write a program that displays the current time and date. Your program will not require any inpute from the user.

In [38]:
```
import time

# Get the current time as a struct_time object
current_time = time.localtime()

# Convert the struct_time object to a string
time_string = time.asctime(current_time)

# Print the result
print("The current time and date is:", time_string)
```

```
The current time and date is: Mon Sep 11 13:10:23 2023
```

Task 27 so long!!!!!

Task 28: BMI challenge Calculate the BMI as user inter the height and weight if height in foot and weight in pound then use: BMI = (weight * 703)/height**2 if height is given in meter and weight is given in kilogram then use this formula: BMI = (weight)/height**2

In [41]:
```
h = float(input("Enter your height in meter: "))
w = float(input("Enter your weight in kilogram: "))
BMI = w/h**2
print("Your BMI is %.2f"%BMI)
```

```
Enter your height in meter: 1.77
Enter your weight in kilogram: 77
Your BMI is 24.58
```

Task 29 (Wind Chill) Calculate the wind chill based on the following formaula:

WCI = 13.12 + 0.6215 *T* - *11.37* V**0.16 + 0.3965*T* V0.16

In [42]:
```
T = float(input("Enter the temprature less than 10 degree of cilsius: "))
V = float(input("Enter the volacity of wind more than 4.8 kilometer per hour: "))
WCI = 13.12 + 0.6215 * T - 11.37* V**0.16 + 0.3965*T* V**0.16
print(round(WCI))
```

```
Enter the temprature less than 10 degree of cilsius: 5
Enter the volacity of wind more than 4.8 kilometer per hour: 6
4
```

Task 30 Celsius to Fahenheit and Kelvin Write a program that begins by reading a temprature from the user in degrees Celsius. Then your program should display the equivalent temprature in degrees Fahrenheit and degrees Kelvin. The calculation needed to convert between different unites of temprature can be found in internet.

In [46]:
```
Selection = input("from Celsius to Fahrenheit type 'cf', from celsius to kelvin typ
```

```
if Selection == 'cf':
    C = float(input("Enter degree in celsius: "))
    F = (C * 9/5) + 32
    print("The desired degree in Fahrenheit is: %.2f"% F)
elif Selection == 'ck':
    C = float(input("Enter degree in celsius: "))
    K = C + 273.15
    print("The desired degree in Kelvin is: %.2f"%K)
elif Selection == 'fk':
    F = float(input("Enter degree in Fahrenheit: "))
    K = (F - 32) * 5/9 + 273.15
    print("The desire degree in Kelvin is : %.2f"%K)
elif Selection == 'fc':
    F = float(input("Enter degree in Fahrenheit: "))
    C = (F - 32) * 5/9
    print("The desired degree in celsius is: %.2f"%C)
elif Selection == 'kc':
    K = float(input("Enter degree in kelvin: "))
    C = K - 273.15
    print("The desired degree from kelvin to celsius is: %.2f "%C)
elif Selection == 'kf':
    K = float(input("Enter degree in kelvin: "))
    F = (K - 273.15) * 9/5 + 32
    print(f"The desired degree from Kelvin to Fahrenheit is: %.2f"%F)
```

```
from Celsius to Fahrenheit type 'cf', from celsius to kelvin type 'ck'
from Fahrenheit to kelvin type'fk', from Fahrenheit to Celsius type 'fc'
from Kelvin to Celsius type  'kc', from Kelvin to Fahrenheit type 'kf' kc
Enter degree in kelvin: 293.15
The desired degree from kelvin to celsius is: 20.00
```

Task 31 Creat a program that reads a pressure from the user in kilo-pascals. Once the pressure has been read your program should report the equivalent pressure in pound per squre inch, millimeters of mercury and atomspheres. Use your research skills to determine the conversion factors between units.

In [47]:
```
kpa = float(input("Enter the pressure in kilopascals: "))

# Convert to pounds per square inch
psi = kpa * 0.145038

# Convert to millimeters of mercury
mmhg = kpa * 7.50062

# Convert to atmospheres
atm = kpa * 0.0098692382432766

# Print the results
print("The pressure in pounds per square inch is:", psi)
print("The pressure in millimeters of mercury is:", mmhg)
print("The pressure in atmospheres is:", atm)
```

```
Enter the pressure in kilopascals: 232
The pressure in pounds per square inch is: 33.648816
The pressure in millimeters of mercury is: 1740.14384
The pressure in atmospheres is: 2.289663272440171
```

Task 32 (Sum of the Digits in an Integer) Develop a program that reads a four digit integer from the user and displays the sum of its digits. For example, if the users enters 3141 then your program should display 3+1+4+1 = 9.

In [60]:
```
num = input("Enter a four-digit integer: ")
sum = int(num[0]) + int(num[1]) + int(num[2]) + int(num[3])
print(f" The sum of the digit is: {sum}")
```

```
Enter a four-digit integer: 9879
 The sum of the digit is: 33
```

In [61]:
```
num = input("Enter a four-digit integer: ")
sum = int(num[0]) + int(num[1]) + int(num[2]) + int(num[3])

# Create a string with the digits separated by '+'
digits = num[0] + "+" + num[1] + "+" + num[2] + "+" + num[3]

print(digits + "=" + str(sum))
```

```
Enter a four-digit integer: 9879
9+8+7+9=33
```

In [66]:
```
num = input("Enter a five-digit integer: ")
sum = int(num[0]) + int(num[1]) + int(num[2]) + int(num[3])+int(num[4])


digits =(num[0]) +"+"+ (num[1])+ "+"+ (num[2]) +"+" +(num[3]+ "+" + (num[4]))


print(digits + "=" + str(sum))
```

```
Enter a five-digit integer: 98799
9+8+7+9+9=42
```

Task 33: Create a program that reads three integers from the user and displays them in sorted order(from smallest to largest). Use the min and max functions to find the smallest and largest values. The middle value can be found by computing the sum of all three values, and then subtracting the minimum value and the maximum value.

In [68]:
```
a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))
c = int(input("Enter the third number: "))
mn = min(a,b,c)
mx = max(a,b,c)
md = a+b+c -mn -mx
print("The numbers in sorted order are: ")
print(" ",mn)
print(" ",md)
print(" ",mx)
```

```
Enter the first number: 23
Enter the second number: 99
Enter the third number: 2323
The numbers in sorted order are:
  23
  99
  2323
```

Task 34 (Day Old Bread) A bakery sells loaves of bread for 3.49$ each. Day old bread is discounted by 60 percent. Write a program that begins by reading the number of loaves of day old bread being purchased from the user. Then your program should display the regular price for the bread, the discount because it is a day old, and the total price. Each of these amounts should be displayed on its own line with an appropriate label. All of the values should be displayed using two decimal places, and the decimal points in all of the numbers should be aligned when reasonable values are entered by the user.

In [72]:
```
BREAD_PRICE = 3.49
DISCOUNT_RATE = 0.60
num_loaves = int(input("Enter the number of day old loaves: "))
regular_price = num_loaves * BREAD_PRICE
discount = regular_price * DISCOUNT_RATE
total = regular_price - discount
print("Regular price :  %5.2f" % regular_price)
print("Discount:  %5.2f"%discount)
print("Total:  %5.2f"% total)
```

```
Enter the number of day old loaves: 12
Regular price :  41.88
Discount:  25.13
Total:  16.75
```

In [75]:
```
# When you buy both fresh and day old bread then the program goes like this:

BREAD_PRICE = 3.49
DISCOUNT_RATE = 0.60
num_loaves_day_old = int(input("Enter the number of day old loaves: "))
num_loaves_fresh = int(input("Enter the number of fresh loaves: "))


price_with_discount = num_loaves_day_old * BREAD_PRICE
discount = price_with_discount * DISCOUNT_RATE


regular_price = price_with_discount+ num_loaves_fresh *BREAD_PRICE


total = regular_price - discount
print("Regular price :  %5.2f" % regular_price)
print("Discount:  %5.2f"%discount)
print("Total:  %5.2f"% total)
```

```
Enter the number of day old loaves: 12
Enter the number of fresh loaves: 10
Regular price :  76.78
Discount:  25.13
Total:  51.65
```

Task 35 Even or Odd?

Write a program that reads an integer from the user. Then your program should display a message indicating whether the integer is even or odd.

In [79]:
```
num =  int(input("Enter any integer number:  "))
if num % 2 == 0:
    print("The given number is even")
else:
    print("The given number is odd")
```

```
Enter any integer number:  432435421
The given number is odd
```

Task 36 Dog Year It is commonly said that one human year is seven dog years. However this simple conversion fails to recognize that dogs reach adulthood in approximately two years. As a result, some people believe that it is better to count each of the first two human years as 10.5 dog years, and then count each additional human years as 4 dog years. Write a program that implements the conversion from human years to dog years described in the previos paragraph. Ensure that your program works correctly for conversion fo less than two

human years adn for conversions of two or more human years. Your program should display an appropriate error message if the user enters a negative number.

In [ ]:
```
```

Task 37: Creat a program that reads the letter from users if the user enter letters like 'a','e','i','o','u' the program should give comment that entered letter is vowel and if the user enter letter 'y' your program should give comments that 'y' is semivowel.

In [4]:
```
letters = input("Enter letter: ")
if letters == 'a'or'e'or'i'or'o'or'u':
    print("Entered letter is vowel")
elif letters == 'y':
    print("The entered letter is semivowel")
else:
    print("The entered letter is cnosonant")
print(letters)
```

```
Enter letter: i
Entered letter is vowel
i
```

In [6]:
```
letters = input("Enter a letter: ")
if letters in ['a', 'e', 'i', 'o', 'u']:
    print("Entered letter is vowel")
elif letters == 'y':
    print("The entered letter is semivowel")
else:
    print("The entered letter is consonant")
```

```
Enter a letter: p
The entered letter is consonant
```

In [7]:
```
letters = input("Enter a letter: ")
if letters in ['a', 'e', 'i', 'o', 'u']:
    print("Entered letter is vowel")
elif letters == 'y':
    print("The entered letter is semivowel")
else:
    print("The entered letter is consonant")
```

```
Enter a letter: u
Entered letter is vowel
```

In [8]:
```
letters = input("Enter a letter: ")
if letters in ['a', 'e', 'i', 'o', 'u']:
    print("Entered letter is vowel")
elif letters == 'y':
    print("The entered letter is semivowel")
else:
    print("The entered letter is consonant")
```

```
Enter a letter: y
The entered letter is semivowel
```

Task 38: Write a program that determines the name of a shape from its number of sides. Read the number of sides form the user and then report the appropriate name as part of the meaningful message. Your program should support shapes with anywhere from 3 then your program should display an appropriate error message.

In [16]:
```
nsides = int(input("Enter the numbe of the sides: "))
# The empty string is being used as a sentinel value. if the number of sides entere
# range then the name will remain empty, causing an error message to be displayed l
name = " "
if nsides == 3:
    name ="Triangle"
elif nsides == 4:
    name ="Quardrilateral"
elif nsides ==5:
    name ="Pentagonal"
elif nsides ==6:
    name =("Hexagonal")
elif nsides ==7:
    name =("Heptagonal")
elif nsides ==8:
    name =("Octagonal")
elif nsides ==9:
    name =("Nanogonal")
elif nsides ==10:
    name =("Decagonal")

if name == " ":
    print("That number of sides is not supported by this program")
else:
    print("That's a", name)
```

```
Enter the numbe of the sides: 12
That number of sides is not supported by this program
```

In [17]:
```
nsides = int(input("Enter the numbe of the sides: "))
# The empty string is being used as a sentinel value. if the number of sides entere
# range then the name will remain empty, causing an error message to be displayed l
name = " "
if nsides == 3:
    name ="Triangle"
elif nsides == 4:
    name ="Quardrilateral"
elif nsides ==5:
    name ="Pentagonal"
elif nsides ==6:
    name =("Hexagonal")
elif nsides ==7:
    name =("Heptagonal")
elif nsides ==8:
    name =("Octagonal")
elif nsides ==9:
    name =("Nanogonal")
elif nsides ==10:
    name =("Decagonal")

if name == " ":
    print("That number of sides is not supported by this program")
else:
    print("That's a", name)
```

```
Enter the numbe of the sides: 8
That's a Octagonal
```

Task 39:Month Name to Number of Days The number of days in a months varies from 28 to 31days. In this exercise you will creat a program that reads the name of month from user as a string. Then your program should display the number of month. Display " 28 or 29 days" for february so that leap years are addressed.

In [18]:
```
month = input("Enter the name of a month: ")
days = 31
if month in ['April','June','September','November']:
    days = 30
elif month =="February":
    days = "28" or "29"
print(month, "has",days, "days in it")
```

```
Enter the name of a month: April
April has 30 days in it
```

In [19]:
```
month = input("Enter the name of a month: ")
days = 31
if month in ['April','June','September','November']:
    days = 30
elif month =="February":
    days = "28" or "29"
print(month, "has",days, "days in it")
```

```
Enter the name of a month: agust
agust has 31 days in it
```

In [20]:
```
month = input("Enter the name of a month: ")
days = 31
if month in ['April','June','September','November']:
    days = 30
elif month =="February":
    days = "28" or "29"
print(month, "has",days, "days in it")
```

```
Enter the name of a month: February
February has 28 days in it
```

Task 40 Write a program that reads a sound level in decibels form the user. if the user enters a decible level that matches one of the noises in the table then your program should display a message containing only that noise. If the user enters a number of decible between the noise listed then your program should display a massege indicating which noise the value is between. Ensure that your program also generates reasonable output for a value smaller than the qiuetest noise in the table, and for a value larger than the loudest noise in the table.

In [22]:
```
Decible = int(input("Enter the level of noise: "))
name = ""
if Decible == 130:
    name = "Jackhammer the loaudest level"
elif Decible == 106:
    name = "Gas Lawnmower"
elif Decible == 70:
    name = "Alarm Clock"
elif Decible == 40:
    name = "Quiet Room the quietest level"
elif Decible >40 and Decible <70:
    name = "Between Quiet level and Alarm Clock level"
elif Decible >70 and Decible <130:
    name = "Between Alarm Clock and Laudest level"


if name == "":
    print("The entered number is not supported by this programm")

print("The entered number is indicating level of ", name)
```

```
Enter the level of noise: 99
The entered number is indicating level of  Between Alarm Clock and Laudest level
```

In [25]:
```
Decible = int(input("Enter the level of noise: "))
name = ""
if Decible == 130:
    name = "Jackhammer the loaudest level"
elif Decible == 106:
    name = "Gas Lawnmower"
elif Decible == 70:
    name = "Alarm Clock"
elif Decible == 40:
    name = "Quiet Room the quietest level"
elif Decible >40 and Decible <70:
    name = "Between Quiet level and Alarm Clock level"
elif Decible >70 and Decible <130:
    name = "Between Alarm Clock and Laudest level"


if name == "":
    print("The entered number is not supported by this programm")
else:
    print("The entered number is indicating level of ", name)
```

```
Enter the level of noise: 134
The entered number is not supported by this programm
```

Task 41: Write a program the read the length of three side of triangle and display its type wheather its An isosceles or scalene or equilateral.

In [31]:
```
a = float(input("Enter first side of triangle: "))
b = float(input("Enter first second of triangle: "))
c = float(input("Enter first third of triangle: "))
name = ""
if a==b==c:
    name = "Equilateral"
elif a == b or a == c or c == b:
    name = "Isosceles"
else:
    name = "Scalene"

print("The triangle type is: ",name)
```

```
Enter first side of triangle: 12
Enter first second of triangle: 12
Enter first third of triangle: 12
The triangle type is:  Equilateral
```

In [32]:
```
a = float(input("Enter first side of triangle: "))
b = float(input("Enter first second of triangle: "))
c = float(input("Enter first third of triangle: "))
name = ""
if a==b==c:
    name = "Equilateral"
elif a == b or a == c or c == b:
    name = "Isosceles"
else:
    name = "Scalene"

print("The triangle type is: ",name)
```

```
Enter first side of triangle: 12
Enter first second of triangle: 22
Enter first third of triangle: 12
The triangle type is:  Isosceles
```

In [33]:
```
a = float(input("Enter first side of triangle: "))
b = float(input("Enter first second of triangle: "))
c = float(input("Enter first third of triangle: "))
name = ""
if a==b==c:
    name = "Equilateral"
elif a == b or a == c or c == b:
    name = "Isosceles"
else:
    name = "Scalene"

print("The triangle type is: ",name)
```

```
Enter first side of triangle: 12
Enter first second of triangle: 13
Enter first third of triangle: 15
The triangle type is:  Scalene
```

Task 42: Note to Freqeuncy

In [41]:
```
C4_FREQ =261.63
D4_FREQ =293.66
E4_FREQ =329.63
F4_FREQ =349.23
G4_FREQ =392.00
A4_FREQ =440.00
B4_FREQ =493.88
name = input("Enter the two character note name, such as C4: ")
# Store the note and its octave in seprate variable
note = name[0]
octave =int(name[1])


if note == "C":
    freq = C4_FREQ
elif note == "D":
    freq = D4_FREQ
elif note == "E":
    freq =E4_FREQ
elif note == "F":
    freq = F4_FREQ
elif note == "G":
    freq =G4_FREQ
elif note == "A":
    freq = A4_FREQ
elif note == "B":
    freq = B4_FREQ

freq = freq / 2**(4-octave)

print("The frequency of", name, "is", freq)
```

```
Enter the two character note name, such as C4: C4
The frequency of C4 is 261.63
```

Task 43: In the previous question you converted from a note's name to its frequency.In this question you will write a program that reserves that process. Begin by reading a frequency form the user. If the frequency is within on Hertz of a value listed in table in the previous

question then report the name of the corresponding note. Otherwise report that the frequency does not correspond to a known note. In this exercise you only need to consider the notes listed in the table. There is no need to consider notes from other octaves.

In [42]:
```
C4_FREQ =261.63
D4_FREQ =293.66
E4_FREQ =329.63
F4_FREQ =349.23
G4_FREQ =392.00
A4_FREQ =440.00
B4_FREQ =493.88
LIMIT = 1
# Read the frequency from the user
freq = float(input("Enter a frequency (Hz): "))
# Determine the note that corresponds to the entered frequency. Set note equal to t
if freq >= C4_FREQ - LIMIT and freq <= C4_FREQ + LIMIT:
    note = "C4"
elif freq >= D4_FREQ - LIMIT and freq <= D4_FREQ +LIMIT:
    note = "D4"
elif freq >= E4_FREQ - LIMIT and freq <= E4_FREQ +LIMIT:
    note = "E4"
elif freq >= F4_FREQ - LIMIT and freq <= F4_FREQ +LIMIT:
    note = "F4"
elif freq >= G4_FREQ - LIMIT and freq <= G4_FREQ +LIMIT:
    note = "G4"
elif freq >= A4_FREQ - LIMIT and freq <= A4_FREQ +LIMIT:
    note = "A4"
elif freq >= B4_FREQ - LIMIT and freq <= B4_FREQ +LIMIT:
    note = "B4"
else:
    note = ""

# Display the result, or an appropriate error message.
if note=="":
    print("There are no note that corresponds to that frequency.")
else:
    print("That frequency is", note)
```

```
Enter a frequency (Hz): 349.23
That frequency is F4
```

In [43]:
```
C4_FREQ =261.63
D4_FREQ =293.66
E4_FREQ =329.63
F4_FREQ =349.23
G4_FREQ =392.00
A4_FREQ =440.00
B4_FREQ =493.88
LIMIT = 1
# Read the frequency from the user
freq = float(input("Enter a frequency (Hz): "))
# Determine the note that corresponds to the entered frequency. Set note equal to t
if freq >= C4_FREQ - LIMIT and freq <= C4_FREQ + LIMIT:
    note = "C4"
elif freq >= D4_FREQ - LIMIT and freq <= D4_FREQ +LIMIT:
    note = "D4"
elif freq >= E4_FREQ - LIMIT and freq <= E4_FREQ +LIMIT:
    note = "E4"
elif freq >= F4_FREQ - LIMIT and freq <= F4_FREQ +LIMIT:
    note = "F4"
elif freq >= G4_FREQ - LIMIT and freq <= G4_FREQ +LIMIT:
    note = "G4"
elif freq >= A4_FREQ - LIMIT and freq <= A4_FREQ +LIMIT:
```

```
    note = "A4"
elif freq >= B4_FREQ - LIMIT and freq <= B4_FREQ +LIMIT:
    note = "B4"
else:
    note = ""

# Display the result, or an appropriate error message.
if note=="":
    print("There are no note that corresponds to that frequency.")
else:
    print("That frequency is", note)
```

```
Enter a frequency (Hz): 395
There are no note that corresponds to that frequency.
```

Task 44: Faces on Money

In [45]:
```
Amount =  int(input("Enter the number on banknote: "))
face = ""
if Amount == 1:
    face = "George Washington"
elif Amount ==2:
    face = "Thomas Jefferson"
elif Amount ==5:
    face = "Abraham Lincoln"
elif Amount ==10:
    face = "Alexander Hamilton"
elif Amount == 20:
    face = "Andrew Jackson"
elif Amount == 50:
    face = "Ulysses S. Grant"
elif Amount == 100:
    face = "Benjamin Franklin"
if face == "":
    print("There are no such a banknote existed")
else:
    print("The given number indicate", face)
```

```
Enter the number on banknote: 20
The given number indicate Andrew Jackson
```

In [46]:
```
Amount =  int(input("Enter the number on banknote: "))
face = ""
if Amount == 1:
    face = "George Washington"
elif Amount ==2:
    face = "Thomas Jefferson"
elif Amount ==5:
    face = "Abraham Lincoln"
elif Amount ==10:
    face = "Alexander Hamilton"
elif Amount == 20:
    face = "Andrew Jackson"
elif Amount == 50:
    face = "Ulysses S. Grant"
elif Amount == 100:
    face = "Benjamin Franklin"
if face == "":
    print("There are no such a banknote existed")
else:
    print("The given number indicate", face)
```

```
Enter the number on banknote: 45
There are no such a banknote existed
```

Task 45: Date on Holiday name Canada has three national holidays which fall on the same dates each year. Write a program that reads a month and day from the user. If the month and day match one of the holidays listed previosly then your program should display the holiday's name. Otherwise your program should indicate that the entered month and day do not correspond to a fixed date holiday.

In [47]:
```
date = input("Enter the month and date: ")
if date == "January 1":
    print("New Year's Day")
elif date == "July 1":
    print("Canada Day")
elif date == " December 25":
    print("Christmas Day")
else:
    print("There are no holyday in this date")
```

```
Enter the month and date: January 1
New Year's Day
```

Task 46 Positions on a chess board are indentified by a letter and a number. The letter indentifies the column, while the number identifies the row. Write a program that reads a position form a user. Use an if statement to determine if the column begins with black square or white square. Then use modulare arithmatic to report the color of square in that row. For example, if the user enters 'a1' then your program should report that the square is black. If the user entere 'd5' then your program should report that the square is white. Your program may assume that a valid position will always be entered. It does not need to perform any error.

In [1]:
```
position = input("Enter a position on the chessboard (e.g. a1): ")
column = position[0]
row = int(position[1])

if (ord(column) + row) % 2 == 0:
    print("The square is black.")
else:
    print("The square is white.")
```

```
Enter a position on the chessboard (e.g. a1): f1
The square is white.
```

In [2]:
```
position=input("Enter a position on the chessboard(e.g.a1):")
column = position[0]
row = int(position[1])
if (ord(column)+ row) / 2 == 0:
    print("The square is black.")
else:
    print("The square is white.")
```

```
Enter a position on the chessboard(e.g.a1):e3
The square is white.
```

Commnet about 46 In the previous code, ord(column) returns the ASCII value of the column character. The ASCII value of the letter 'a' is 97, 'b' is 98, and so on. The ord() function is used to convert the column character to its corresponding ASCII value. The expression (ord(column) + row) % 2 == 0 checks if the sum of the ASCII value of the column character

and the row number is even or odd. If the sum is even, then the square is black, otherwise, it is white. For example, if the user enters 'a1', then the ASCII value of 'a' is 97, and the row number is 1. So, (97 + 1) % 2 is equal to 0, which means the square is black. If the user enters 'd5', then the ASCII value of 'd' is 100, and the row number is 5. So, (100 + 5) % 2 is equal to 1, which means the square is white.

Task 47: The year is divided in four seasons: spring, summer, fall and winter. While the exact date that season chage vary a litter bit form year to year because of the way that the calender is constracted, we will use the following dates for this task.

season First dat Spring March 20 Summer June 21 Fall September 22 Winter December 21 Create a program that reads a month and day form the user. The user will enter the name of the month as a string, followed by the day within the month as an integer. Then your program should display the season associated with the date that was entered.

In [8]:
```
month = input("Enter the name of the month: ")
day = int(input("Enter the day number: "))

if month=="January" or month == "February":
    season = "Winter"
elif month =="March":
    if day < 20:
        season = "Winter"
    else:
        season = "Spring"

elif month =="April"or month == "May":
    season = "Spring"
elif month == "June":
    if day < 21:
        season = "Spring"
    else:
        season ="Summer"

elif month=="July"or month =="August":
    season == "Summer"
elif month =="September":
    if day <22:
        season = "Summer"
    else:
        season ="Fall"


elif month =="October"or month == "November":
    season = " Winter"
elif month == "December":
    if day < 21:
        season = "Winter"
    else:
        season ="Spring"

print(month, day, "is in", season)
```

```
Enter the name of the month: September
Enter the day number: 9
September 9 is in Summer
```

In [9]:
```
month = input("Enter the name of the month: ")
day = int(input("Enter the day number: "))

if month=="January" or month == "February":
    season = "Winter"
elif month =="March":
    if day < 20:
        season = "Winter"
    else:
        season = "Spring"

elif month =="April"or month == "May":
    season = "Spring"
elif month == "June":
    if day < 21:
        season = "Spring"
    else:
        season ="Summer"

elif month=="July"or month =="August":
    season == "Summer"
elif month =="September":
    if day <22:
        season = "Summer"
    else:
        season ="Fall"


elif month =="October"or month == "November":
    season = " Winter"
elif month == "December":
    if day < 21:
        season = "Winter"
    else:
        season ="Spring"

print(month, day, "is in", season)
```

```
Enter the name of the month: October
Enter the day number: 14
October 14 is in  Winter
```

In [14]:
```
month = input("Enter the month: ")
day = int(input("Enter the day: "))

if month =="January" or month == "February":
    season = "Winter"
elif month == "March":
    if day < 20:
        season ="Winter"
    else:
        season = "Spirng"

elif month == "April" or month == "May":
    season = "Spring"
elif month =="June":
    if day < 21:
        season = "Spring"
    else:
        season ="Summer"

elif month == "July" or month =="August":
    seaon = "Summer"
```

```
elif month =="September":
    if day<22:
        season = "Summer"
    else:
        season = "Fall"

elif month == "October" or month == "November":
    season = "Fall"
elif month == December:
    if day < 21:
        season ="Fall"
    else:
        season = "Winter"
print(month, day, "is", season, "season")
```

```
Enter the month: September
Enter the day: 9
September 9 is Summer season
```

Task 48: The horoscopes commonly reported in newspapers use the position of the sun at the time of one's birth to try and predict the future. This system of astrology divides the year into twelve zodiac signs, as outline in table: Capricorn December 22 to January 19 Aquarius January 20 to February 18 Pisces February 19 to March 20 Aries March 21 to April 20 Taurus April 21 to May 20 Gemini May 21 to June 20 Cancer June 21 to July 22 Leo July 23 to August22 Virgo Ausust 23 to September 22 Libra September 23 to October 21 Scorpio October 23 to Novemer 21 Sagittarius November 22 to December 21

In [21]:
```
month = input("Enter a month: ")
day = int(input("Enter a day:" ))

if month == "December":
    signs = "Capricon"
    if day > 22:
        signs = "Capricon"
    else:
        signs = Safittarius
elif month == "January":
    if day < 19:
        signs = "Capricon"
    else:
        signs = "Aquarius"



elif month =="January":
    if day >20:
        signes = "Aquarius"
elif month == "February":
    if day <18:
        signs = "Aquarius"
    else:
        signs = "Pisces"

elif month == "February":
    if day >19:
        signs = "Pisces"
elif month == "March":
    if day < 20:
        signs = "Pisces"
    else:
        signs = "Aries"
```

```
elif month == "March":
    if day >= 21:
        signs = "Aries"
elif month == "April":
    if day < 19:
        signs = "Aries"
    else:
        signs = "Taurus"

elif month == "April":
    if day >=20:
        signs = "Taurus"
elif month == "May":
    if day < 20:
        signs = "Taurus"


elif month =="May":
    if day>= 21:
        signs = "Gemini"
elif month =="June":
    if day < 20:
        signs = "Gemini"

elif month == "June":
    if day >= 21:
        signs = "Cancer"
elif month == "July":
    if day < 20:
        signs = "Cancer"


elif month == "July":
    if day >=20:
        signs = "Leo"
elif month == "August":
    if day < 22:
        signs = "Leo"

elif month == "August":
    if day >=23:
        signs ="Virgo"
elif month =="September":
    if day < 22:
        signs = "Virgo"


elif month == "September":
    if day >=23:
        signs ="Liba"
elif month == "October":
    if day < 22:
        signs = "Libra"

elif month == "October":
    if day > 23:
        signs = "Scorpio"
elif month == " November":
    if day < 21:
        signs = "Scorpio"

elif month =="November":
```

```
    if day >22:
        signs ="Sagittarius"
elif month == "December":
    if day < 21:
        signs ="Sagittarius"


print(month, day, "is", signs)
```

```
Enter a month: July
Enter a day:5
July 5 is Cancer
```

This code is too much long The follwoing code is shorter and has the following explianation

Explanation: We create a dictionary zodiac_signs that maps each zodiac sign to a list of month and day ranges. We loop through the items of the dictionary using the items() method. For each item, we check if the input month and day fall within the month and day ranges for that zodiac sign. If there is a match, we assign the zodiac sign to the variable zodiac_sign and exit the loop using the break statement. Finally, we print the input month, day, and zodiac sign. This code is shorter and more readable than the original code, and it can be easily extended to include more zodiac signs or different month and day ranges.

In [28]:
```
import calendar
month = input("Enter a month: ")
day = int(input("Enter a day: "))

zodiac_signs = {
    "Capricon": [(1, 19), (12, 22)],
    "Aquarius": [(1, 20), (2, 18)],
    "Pisces": [(2, 19), (3, 20)],
    "Aries": [(3, 21), (4, 19)],
    "Taurus": [(4, 20), (5, 20)],
    "Gemini": [(5, 21), (6, 20)],
    "Cancer": [(6, 21), (7, 22)],
    "Leo": [(7, 23), (8, 22)],
    "Virgo": [(8, 23), (9, 22)],
    "Libra": [(9, 23), (10, 22)],
    "Scorpio": [(10, 23), (11, 21)],
    "Sagittarius": [(11, 22), (12, 21)]
}

for sign, ranges in zodiac_signs.items():
    if (month == calendar.month_name[ranges[0][0]] and day >= ranges[0][1]) or \
       (month == calendar.month_name[ranges[1][0]] and day <= ranges[1][1]):
        zodiac_sign = sign
        break

print(month, day, "is", zodiac_sign)
```

```
Enter a month: July
Enter a day: 5
July 5 is Cancer
```

Task 49 The Chinese zodiac assigns animals to year in a 12 year cycle. One 12 cycle is shown in the table below. The pattern repeats from there, with 2012 being another year of the dragon, and 1999 being year of the Hare. Year Animal 2000 Dragon 2001 Snake 2002 Horse 2003 Sheep 2004 Monkey 2005 Rooster 2006 Dog 2007 Pig 2008 Rat 2009 Ox 2010 Tiger 2011 Hare

Write a porgram that reads a year from ther user and displays the animal associated with that year. Your program should work correctly for any year greater than or equal to zero, not just the ones listed in the table.

In [33]:
```
year = int(input("Enter a year: "))
if year %12 ==8:
    animal = "Dragon"
elif year % 12 == 9:
    animal ="Snake"
elif year % 12 ==10:
    animal = "Horse"
elif year % 12 == 11:
    animal = "Sheep"
elif year % 12 == 0:
    animal = "Monkey"
elif year %12 == 1:
    animal = "Rooster"
elif year % 12 ==2:
    animal = "Dog"
elif year % 12 ==3:
    animal = "Pig"
elif year % 12 ==4:
    animal = "Rat"
elif year %12==5:
    animal = "Ox"
elif year %12 ==6:
    animal = "Tiger"
elif year %12 == 7:
    animal ="Hare"

print("The inserted", year,"is",animal)
```

```
Enter a year: 1994
The inserted 1994 is Dog
```

The two following codes are the shorter formate of my code but its output is not the same as my code. You have to find the problem where is the probelms and how it works.

Hint: Explanation: We create a list zodiac_animals that contains the zodiac animals in order. We calculate the index of the animal based on the input year using the formula (year - 4) % 12. This formula works because the zodiac cycle starts with the year of the Rat, which is 4 in the modulo 12 system. So we subtract 4 from the input year to align it with the zodiac cycle, and then take the modulo 12 to get the index of the animal in the list. We assign the zodiac animal to the variable animal using the calculated index. Finally, we print the input year and zodiac animal. This code is shorter and more efficient than the original code, and it can be easily extended to include more zodiac animals or different starting years.

In [38]:
```
year = int(input("Enter a year: "))

zodiac_animals = ["Monkey", "Rooster", "Dog", "Pig", "Rat", "Ox", "Tiger", "Hare",
animal_index = (year - 4) % 12
animal = zodiac_animals[animal_index]

print("The inserted", year, "is", animal)
```

```
Enter a year: 1994
The inserted 1994 is Horse
```

In [37]:
```
year = int(input("Enter a year: "))
zodiac_signs = {
    0: "Monkey", 1: "Rooster", 2: "Dog", 3: "Pig", 4: "Rat", 5: "Ox",
    6: "Tiger", 7: "Rabbit", 8: "Dragon", 9: "Snake", 10: "Horse", 11: "Sheep"
}
animal = zodiac_signs[(year - 4) % 12]
print(f"The inserted {year} is {animal}")
```

```
Enter a year: 1994
The inserted 1994 is Horse
```

In [2]:
```
year = int(input("Enter a year: "))

zodiac_animals = ["Monkey", "Rooster", "Dog", "Pig", "Rat", "Ox", "Tiger", "Hare",
animal_index = (year) % 12
animal = zodiac_animals[animal_index]

print("The inserted", year, "is", animal)
```

```
Enter a year: 1994
The inserted 1994 is Dog
```

In [3]:
```
year = int(input("Enter a year: "))
zodiac_signs = {
    0: "Monkey", 1: "Rooster", 2: "Dog", 3: "Pig", 4: "Rat", 5: "Ox",
    6: "Tiger", 7: "Rabbit", 8: "Dragon", 9: "Snake", 10: "Horse", 11: "Sheep"
}
animal = zodiac_signs[year % 12]
print(f"The inserted {year} is {animal}")
```

```
Enter a year: 1994
The inserted 1994 is Dog
```

Task 50: The following table contains earthquack magnitude ranges on the Richter scale and their descriptors: Magnitude Descriptor Less than 2.0 Micro 2.0 to less than 3.0 Very Minor 3.0 to less than 4.0 Minor 4.0 to less than 5.0 Light 5.0 to less than 6.0 Moderate 6.0 to less than 7.0 Strong 7.0 to less than 8.0 Majore 8.0 to less than 10.0 Great 10.0 or more Meteoric

Write a program that reads a magnitude from the user and displays the appropriate descriptor as part of a meaningful message. For example, if the user enters 5.5 then your program should display that a magnitude 5.5 earthquack is considered to be a moderate earthquack.

In [11]:
```
magnitude = float(input("Enter the earthquack magnitude: "))
if 0.0<magnitude<2.0:
    print("A magnitude", magnitude,"is considered as Micro")
elif 2.0<= magnitude<3.0:
    print("A magnitude", magnitude,"is considered as Very Minor")
elif 3.0<=magnitude<4.0:
    print("A magnitude", magnitude,"is considered as Minor")
elif 4.0<= magnitude <5.0:
    print("A magnitude", magnitude,"is considered as Light")
elif 5.0<= magnitude <6.0:
    print("A magnitude", magnitude,"is considered as Moderate")
elif 6.0<=magnitude<7.0:
    print("A magnitude", magnitude,"is considered as Strong")
elif 7.0<=magnitude <8.0:
    print("A magnitude", magnitude,"is considered as Major")
elif 8.0<=magnitude < 10.0:
    print("A magnitude", magnitude,"is considered as Great")
```

```
elif magnitude>=10:
    print("A magnitude", magnitude,"is considered as Miteoric")
```

```
Enter the earthquack magnitude: 4
A magnitude 4.0 is considered as Light
```

In [9]:
```
magnitude = float(input("Enter the earthquack magnitude: "))
if 0.0<magnitude<2.0:
    print("A magnitude", magnitude,"is considered as Micro")
elif 2.0<= magnitude<3.0:
    print("A magnitude", magnitude,"is considered as Very Minor")
elif 3.0<=magnitude<4.0:
    print("A magnitude", magnitude,"is considered as Minor")
elif 4.0<= magnitude <5.0:
    print("A magnitude", magnitude,"is considered as Light")
elif 5.0<= magnitude <6.0:
    print("A magnitude", magnitude,"is considered as Moderate")
elif 6.0<=magnitude<7.0:
    print("A magnitude", magnitude,"is considered as Strong")
elif 7.0<=magnitude <8.0:
    print("A magnitude", magnitude,"is considered as Major")
elif 8.0<=magnitude < 10.0:
    print("A magnitude", magnitude,"is considered as Great")
elif magnitude>=10:
    print("A magnitude", magnitude,"is considered as Miteoric")
```

```
Enter the earthquack magnitude: 3
A magnitude 3.0 is considered as Minor
```

Task 51:Roots of a quadratic function Write a program that computes the real roots of a quadratic function. Your program should begin by prompting the user for the values of a , b , and, c. Then it should display a message indicating the number of real roots, along with the values of the real roots(if any).

$$f(x) = ax**2 + bx + c$$

$$root1 = (-b + sqrt(b**2 - 4ac))/2a$$

$$root2 = (-b - sqrt(b**2 - 4ac))/2a$$

In [13]:
```
from math import sqrt
a = float(input("Enter a non zero value for a: "))
b = float(input("Enter a non zero value for b: "))
c = float(input("Enter a non zero value for c: "))
d = b**2 - 4*a*c
root1 = (-b + sqrt(b**2-4*a*c))/2*a
if d ==0:
    print("The quadratic function has one root which is: ", root1)
elif d >0:
    root2 = (-b - sqrt(b**2-4*a*c))/2*a
    print("The quadratic function has one root which is: ", root1,root2)
elif d < 0 :
    print("This quadratic function has no real root")
```

```
Enter a non zero value for a: 1
Enter a non zero value for b: 7
Enter a non zero value for c: 12
The quadratic function has one root which is:  -3.0 -4.0
```

In [18]:
```
from math import sqrt
a = float(input("Enter a non zero value for a: "))
b = float(input("Enter a non zero value for b: "))
```

```
c = float(input("Enter a non zero value for c: "))
d = b**2 - 4*a*c
root1 = (-b + sqrt(b**2-4*a*c))/2*a
if d ==0:
    print("The quadratic function has one root which is: ", root1)
elif d >0:
    root2 = (-b - sqrt(b**2-4*a*c))/2*a
    print("The quadratic function has tow roots:","x1=" ,root1," ""x2=",root2)
elif d < 0 :
    print("This quadratic function has no real root")
# When the discriminant is negative then you will face with this error to solve it
```

```
Enter a non zero value for a: 6
Enter a non zero value for b: 8
Enter a non zero value for c: 12
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[18], line 6
      4 c = float(input("Enter a non zero value for c: "))
      5 d = b**2 - 4*a*c
----> 6 root1 = (-b + sqrt(b**2-4*a*c))/2*a
      7 if d ==0:
      8     print("The quadratic function has one root which is: ", root1)

ValueError: math domain error
```

In [25]:
```
a = float(input("Enter a non zero value for a: "))
b = float(input("Enter a non zero value for b: "))
c = float(input("Enter a non zero value for c: "))
d = b**2 - 4*a*c
if d <0:
    print("The quadratic function has no real roots")
else:
    root1 = ((-b+sqrt(d))/(2*a))
    root2 = ((-b-sqrt(d))/(2*a))
    if d == 0:
        print("The quadratic function has one root which is: ", root1)
    else:
        print("The quadratic function has two roots which are: ", root1, "and", roo
```

```
Enter a non zero value for a: 1
Enter a non zero value for b: 7
Enter a non zero value for c: 12
The quadratic function has two roots which are:  -3.0 and -4.0
```

Task 52: Latte Grade Points At a particular university, letter grades are mapped to grade points in the following manner. Letter Grade Points A+ 4.0 A 4.0 A- 3.7 B+ 3.3 B 3.0 B- 2.7 C+ 2.3 C 2.0 C- 1.7 D+ 1.3 D 1.0 F 0 Write a program that begins by reading aletter grade from user. Then your program should compute and display the equivalent number of grade points. Ensure that your program generates an appropriate error message if the user enters an invalid letter grade.

In [2]:
```
latter = input("Enter a latter: ")
if latter == 'A+' or latter =='A':
    print("Your corresponding number is 4")
elif latter =='A-':
    print("Your corresponding number is 3.7")
elif latter =='B+':
    print("Your corresponding number is 3.3")
elif latter =='B':
    print("Your corresponding number is 3.0")
elif latter =='B-':
```

```
    print("Your corresponding number is 2.7")
elif latter =='C+':
    print("Your corresponding number is 2.3")
elif latter =='C':
    print("Your corresponding number is 2.0")
elif latter =='C-':
    print("Your corresponding number is 1.7")
elif latter =='D+':
    print("Your corresponding number is 1.3")
elif latter =='D':
    print("Your corresponding number is 1")
elif latter =='F':
    print("Your corresponding number is 0")
else:
    print("The entered latter does not correspond the system try again.")
```

```
Enter a latter: C-
Your corresponding number is 1.7
```

In [3]:
```
# THE SHORTER FORM OF THIS CODE BY USING DICTIONARY COULD BE LIKE THIS:
grades = {
    'A+': 4.0, 'A': 4.0, 'A-': 3.7, 'B+': 3.3, 'B': 3.0, 'B-': 2.7, 'C+': 2.3,
          'C': 2.0, 'C-': 1.7, 'D+': 1.3, 'D': 1.0, 'F': 0.0}

letter = input("Enter a letter: ")
if letter in grades:
    print(f"Your corresponding number is {grades[letter]}")
else:
    print("The entered letter does not correspond to the system. Please try again.'
```

```
Enter a letter: A-
Your corresponding number is 3.7
```

In [9]:
```
grades = {
    'A+': 4.0, 'A':4.0, 'A-':3.7 , 'B+': 3.3 , 'B':3.0,
    'B-': 2.7, 'C+': 2.3,
          'C': 2.0, 'C-': 1.7, 'D+': 1.3, 'D': 1.0, 'F': 0.0

}
letter = input("Enter a letter: ").upper()
if letter in grades:
    print(f"Your corresponding number is {grades[letter]}")
else:
    print("The entered letter does not correspond to the system.")
```

```
Enter a letter: C
Your corresponding number is 2.0
```

Exercise 53: Grade Points to Letter Grade

In the previous exercise you created a program that converted a letter grade into the equivalent number of grade points. In this exercise you will create a program that reverses the process and converts from a grade point value entered by the user to a letter grade. Ensure that your program handles grade point values that fall between letter grades. These should be rounded to the closest letter grade. Your program should report A+ if the value entered by the user is 4.0 or more.

In [12]:
```
number = float(input("Enter a latter: "))
if number==0.0:
    print("Your corresponding letter is F")
elif 0<number <=1.0:
```

```
    print("Your corresponding number is D")
elif 1<number <=1.3:
    print("Your corresponding number is D+")
elif 1.3<number <=1.7:
    print("Your corresponding number is C-")
elif 1.7<number <=2:
    print("Your corresponding number is C")
elif 2<number <=2.3:
    print("Your corresponding number is C+")
elif 2.3<number <=2.7:
    print("Your corresponding number is B-")
elif 2.7<number <=3.0:
    print("Your corresponding number is B")
elif 3.0<number <=3.3:
    print("Your corresponding number is B+")
elif 3.3<number <=3.7:
    print("Your corresponding number is A-")
elif 3.7<number <=4.0:
    print("Your corresponding number is A OR A+")
else:
    print("The entered latter does not correspond the system try again.")
```

```
Enter a latter: 1.99
Your corresponding number is C
```

Task 54:

At a particular company, employees are rated at the end of each year. The rating scale begins at 0.0, with higher values indicating better performance and resulting in larger raises. The value awarded to an employee is either 0.0, 0.4, or 0.6 or more. Values between 0.0 and 0.4, and between 0.4 and 0.6 are never used. The meaning associated with each rating is shown in the following table. The amount of an employee's raise is $2,400.00 multiplied by their rating.

Rating Meaning 0.0 Unacceptable Performance 0.4 Acceptable Performance 0.6 or more Meritorious Performance

Write a program that reads a rating from the user and indicates whether the performance for that rating is unacceptable, acceptable or meritorious. The amount of the employee's raise should also be reported. Your program should display an appropriate error message if an invalid rating is entered.

In [16]:
```
Rating = float(input("Enter rating the employees: "))
Raising = Rating * 2400
if Rating ==0.0:
    print("The mentioned employee performance is unacceptable")
elif Rating ==0.4:
    print("The mentioned employee performance is acceptable")
    print(f"The amount of raising is, {Raising}")
elif Rating >=0.6:
    print("The mentioned employee performance is meritorious")
    print(f"The amount of raising is, {Raising}")
```

```
Enter rating the employees: 0.9
The mentioned employee performance is meritorious
The amount of raising is, 2160.0
```

In [29]:
```
Rating = float(input("Enter rating the employees: "))
Raising = (Rating * 2400)
```

```
if Rating ==0.0:
    print("The mentioned employee performance is unacceptable")
elif Rating ==0.4:
    print("The mentioned employee performance is acceptable")
    print(f"The amount of raising is, {Raising}")
elif Rating >=0.6:
    print("The mentioned employee performance is meritorious")
    print("The amount of raising is: %5.2f $" %Raising)
else:
    print("That was not a valid rating")
```

```
Enter rating the employees: 0.987
The mentioned employee performance is meritorious
The amount of raising is: 2368.80 $
```

In [30]:
```
Rating = float(input("Enter rating the employees: "))
Raising = (Rating * 2400)
if Rating ==0.0:
    print("The mentioned employee performance is unacceptable")
elif Rating ==0.4:
    print("The mentioned employee performance is acceptable")
    print(f"The amount of raising is, {Raising}")
elif Rating >=0.6:
    print("The mentioned employee performance is meritorious")
    print("The amount of raising is: %5.2f $" %Raising)
else:
    print("That was not a valid rating")
```

```
Enter rating the employees: 0.234
That was not a valid rating
```

Task55:Wavelengths of Visible Light

The wavelength of visible light ranges from 380 to 750 nanometers (nm). While the spectrum is continuous, it is often divided into 6 colors as shown below: Color Wavelength (nm) Violet 380 to less than 450 Blue 450 to less than 495 Green 495 to less than 570 Yellow 570 to less than 590 Orange 590 to less than 620 Red 620 to 750 Write a program that reads a wavelength from the user and reports its color. Display an appropriate error message if the wavelength entered by the user is outside of the visible spectrum.

In [31]:
```
Wavelength_in_nm = int(input("Enter a latter: "))
if 380<=Wavelength_in_nm <450:
    print("The wave color is Violet")
elif 450<=number <495:
    print("The wave color is Blue")
elif 495<=number <570:
    print("The wave color is Green")
elif 570<=number <590:
    print("The wave color is Yellow")
elif 590<=number <620:
    print("The wave color is Orange")
elif 620<=number <750:
    print("The wave color is Red")
else:
    print("The entered latter does not correspond the system try again.")
```

```
Enter a latter: 400
The wave color is Violet
```

In [33]:
```
# Shorter version
```

```
wavelength = int(input("Enter the wavelength in nm: "))

if 380 <= wavelength < 450:
    color = "Violet"
elif 450 <= wavelength < 495:
    color = "Blue"
elif 495 <= wavelength < 570:
    color = "Green"
elif 570 <= wavelength < 590:
    color = "Yellow"
elif 590 <= wavelength < 620:
    color = "Orange"
elif 620 <= wavelength < 750:
    color = "Red"
else:
    print("The entered wavelength does not correspond to the system. Please try aga
    exit()

print(f"The wave color is {color}.")
```

```
Enter the wavelength in nm: 400
The wave color is Violet.
```

Task 56 Electromagnetic radiation can be classified into one of 7 categories according to its frequency, as shown in the table below:

Name Frequency Range (Hz) Radio Waves Less than 3 × 109 Microwaves 3 × 109 to less than 3 × 1012 Infrared Light 3 × 1012 to less than 4.3 × 1014 Visible Light 4.3 × 1014 to less than 7.5 × 1014 Ultraviolet Light 7.5 × 1014 to less than 3 × 1017 X-Rays 3 × 1017 to less than 3 × 1019 Gamma Rays 3 × 1019 or more

Write a program that reads the frequency of some radiation from the user and displays name of the radiation as part of an appropriate message.

In [32]:
```
frequency = float(input("Enter the frequency of the radiation: "))
# 3*10**9 = 3e9
if frequency < 3e9:
    name = "Radio Waves"
elif frequency < 3e12:
    name = "Microwaves"
elif frequency < 4.3e14:
    name = "Infrared Light"
elif frequency < 7.5e14:
    name = "Visible Light"
elif frequency < 3e17:
    name = "Ultraviolet Light"
elif frequency < 3e19:
    name = "X-Rays"
else:
    name = "Gamma Rays"

print(f"The radiation is classified as {name}.")
```

```
Enter the frequency of the radiation: 3e11
The radiation is classified as Microwaves.
```

Exercise 57: Cell Phone Bill A particular cell phone plan includes 50 minutes of air time and 50 text messages for $15.00 a month. Each additional minute of air time costs 0.25, while additional text messages cost

$0.15 each. All cell phone bills include an additional charge of $0.44 to support 911 call centers, and the entire bill (including the 911 charge) is subject to 5 percent sales tax.

Write a program that reads the number of minutes and text messages used in a month from the user. Display the base charge, additional minutes charge (if any), additional text message charge (if any), the 911 fee, tax and total bill amount. Only display the additional minute and text message charges if the user incurred costs in these categories. Ensure that all of the charges are displayed using 2 decimal places.

In [1]:
```
# Constants
BASE_CHARGE = 15.00
AIR_TIME_INCLUDED = 50
AIR_TIME_CHARGE = 0.25
TEXT_MESSAGES_INCLUDED = 50
TEXT_MESSAGE_CHARGE = 0.15
SUPPORT_CHARGE = 0.44
TAX_RATE = 0.05

# Input
air_time_used = int(input("Enter the number of minutes used: "))
text_messages_used = int(input("Enter the number of text messages used: "))

# Calculations
air_time_charge = 0
if air_time_used > AIR_TIME_INCLUDED:
    air_time_charge = (air_time_used - AIR_TIME_INCLUDED) * AIR_TIME_CHARGE

text_message_charge = 0
if text_messages_used > TEXT_MESSAGES_INCLUDED:
    text_message_charge = (text_messages_used - TEXT_MESSAGES_INCLUDED) * TEXT_MESS

subtotal = BASE_CHARGE + air_time_charge + text_message_charge + SUPPORT_CHARGE
tax = subtotal * TAX_RATE
total = subtotal + tax

# Output
print("Base charge: ${:.2f}".format(BASE_CHARGE))
if air_time_charge > 0:
    print("Additional air time charge: ${:.2f}".format(air_time_charge))
if text_message_charge > 0:
    print("Additional text message charge: ${:.2f}".format(text_message_charge))
print("911 support charge: ${:.2f}".format(SUPPORT_CHARGE))
print("Tax: ${:.2f}".format(tax))
print("Total bill: ${:.2f}".format(total))
```

```
Enter the number of minutes used: 56
Enter the number of text messages used: 67
Base charge: $15.00
Additional air time charge: $1.50
Additional text message charge: $2.55
911 support charge: $0.44
Tax: $0.97
Total bill: $20.46
```

In [36]:
```
minutes = int(input("Enter the number of minutes used: "))
texts = int(input("Enter the number of text messages used: "))

base_charge = 15.00
additional_minutes_charge = max(0, minutes - 50) * 0.25
additional_texts_charge = max(0, texts - 50) * 0.15
call_center_fee = 0.44
```

```
tax_rate = 0.05

total_charge = base_charge + additional_minutes_charge + additional_texts_charge +
tax = total_charge * tax_rate
total_bill = total_charge + tax

print(f"Base charge: ${base_charge:.2f}")
if additional_minutes_charge > 0:
    print(f"Additional minutes charge: ${additional_minutes_charge:.2f}")
if additional_texts_charge > 0:
    print(f"Additional texts charge: ${additional_texts_charge:.2f}")
print(f"call_center_fee: ${call_center_fee:.2f}")
print(f"Tax: ${tax:.2f}")
print(f"Total bill: ${total_bill:.2f}")
```

```
Enter the number of minutes used: 56
Enter the number of text messages used: 78
Base charge: $15.00
Additional minutes charge: $1.50
Additional texts charge: $4.20
call_center_fee: $0.44
Tax: $1.06
Total bill: $22.20
```

Exercise 58: Is It a Leap Year?

Most years have 365 days. However, the time required for the Earth to orbit the Sun is actually slightly more than that. As a result, an extra day, February 29, is included in some years to correct for this difference. Such years are referred to as leap years. The rules for determining whether or not a year is a leap year follow: • Any year that is divisible by 400 is a leap year. • Of the remaining years, any year that is divisible by 100 is not a leap year. • Of the remaining years, any year that is divisible by 4 is a leap year. • All other years are not leap years. Write a program that reads a year from the user and displays a message indicating whether or not it is a leap year.

In [3]:
```
year = int(input("Enter a year: "))
# Determine if the inserted year is a leap year.
if year % 400==0:
    Is_Leap_Year = True
elif year % 100 ==0:
    Is_Leap_Year = False
elif year % 4 ==0:
    Is_Leap_Year =True
else:
    Is_Leap_Year = False
# Displaying the result.
if Is_Leap_Year:
    print(year,"is a leap year.")
else:
    print(year,"is not leap year")
```

```
Enter a year: 2023
2023 is not leap year
```

Exercise 59: Next Day

Write a program that reads a date from the user and computes its immediate successor. For example, if the user enters values that represent 2019-11-18 then your program should display a message indicating that the day immediately after 2019-11-18 is 2019-11-19. If the user enters values that represent 2019-11-30 then the program should indicate that the next

day is 2019-12-01. If the user enters values that represent 2019-12-31 then the program should indicate that the next day is 2020-01-01. The date will be entered in numeric form with three separate input statements; one for the year, one for the month, and one for the day. Ensure that your program works correctly for leap years.

In [5]:
```
import datetime

# Get the date from the user
year = int(input("Enter the year (yyyy): "))
month = int(input("Enter the month (mm): "))
day = int(input("Enter the day (dd): "))

# Create a datetime object with the user's input
date = datetime.date(year, month, day)

# Compute the next day
# The line next_day = date + datetime.timedelta(days=1)
# in the Python program computes the next day of the date entered by the user.

next_day = date + datetime.timedelta(days=1)

# Print the result
print("The day immediately after {} is {}".format(date, next_day))
```

```
Enter the year (yyyy): 2023
Enter the month (mm): 09
Enter the day (dd): 16
The day immediately after 2023-09-16 is 2023-09-17
```

Exercise 60:What Day of the Week Is January 1? (32 Lines) The following formula can be used to determine the day of the week for January 1 in a given year: day_of_the_week = (year + floor((year − 1) / 4) − floor((year − 1) / 100) + floor((year − 1) / 400)) % 7 The result calculated by this formula is an integer that represents the day of the week. Sunday is represented by 0. The remaining days of the week following in sequence through to Saturday, which is represented by 6. Use the formula above to write a program that reads a year from the user and reports the day of the week for January 1 of that year. The output from your program should include the full name of the day of the week, not just the integer returned by the formula.

In [2]:
```
import math

year = int(input("Enter a year: "))

day_of_the_week = (year + math.floor((year - 1) / 4) - math.floor((year - 1) / 100)

days_of_week = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", '

print("January 1 of", year, "falls on a", days_of_week[day_of_the_week] + ".")
```

```
Enter a year: 2024
January 1 of 2024 falls on a Monday.
```

Task 61: Is a License Plate Valid?

In a particular jurisdiction, older license plates consist of three uppercase letters followed by three digits. When all of the license plates following that pattern had been used, the format was changed to four digits followed by three uppercase letters. Write a program that begins by reading a string of characters from the user. Then your program should display a

message indicating whether the characters are valid for an older style license plate or a newer style license plate. Your program should display an appropriate message if the string entered by the user is not valid for either style of license plate

In [5]:
```
# Read the license plate from the user
license_plate = input("Enter a license plate: ")

# Check if the license plate is valid for an older style
if len(license_plate) == 6 and license_plate[:3].isalpha() and license_plate[3:].is
    print("The license plate is valid for an older style.")

# Check if the license plate is valid for a newer style
elif len(license_plate) == 7 and license_plate[:4].isdigit() and license_plate[4:]
    print("The license plate is valid for a newer style.")

# If the license plate is not valid for either style
else:
    print("The license plate is not valid for either style.")
```

```
Enter a license plate: 121ASXZ
The license plate is not valid for either style.
```

In [23]:
```
plate = input("Enter the license plate: ")
if len(plate) ==6 and \
     plate[0]>= "A" and plate[0] <= "Z" and \
     plate[1]>= "A" and plate[1] <= "Z" and \
     plate[2]>= "A" and plate[2] <= "Z" and \
     plate[3]>= "0" and plate[3] <= "9" and \
     plate[4]>= "0" and plate[4] <= "9" and \
     plate[5]>= "0" and plate[5] <= "9":
    print("The plate is a valid older style plate. ")
elif len(plate)==7 and \
     plate[0]>= "0" and plate[0] <= "9" and \
     plate[1]>= "0" and plate[1] <= "9" and \
     plate[2]>= "0" and plate[2] <= "9" and \
     plate[3]>= "0" and plate[3] <= "9" and \
     plate[4]>= "A" and plate[4] <= "Z" and \
     plate[5]>= "A" and plate[5] <= "Z" and \
     plate[6]>= "A" and plate[6] <= "Z":
    print("The plate is a valid newer style plate.")
else:
    print("The plate is not valid")
```

```
Enter the license plate: QWE123
The plate is a valid older style plate.
```

In [25]:
```
plate = input("Enter the license plate: ")
if len(plate) ==6 and \
     plate[0]>= "A" and plate[0] <= "Z" and \
     plate[1]>= "A" and plate[1] <= "Z" and \
     plate[2]>= "A" and plate[2] <= "Z" and \
     plate[3]>= "0" and plate[3] <= "9" and \
     plate[4]>= "0" and plate[4] <= "9" and \
     plate[5]>= "0" and plate[5] <= "9":
    print("The plate is a valid older style plate. ")
elif len(plate)==7 and \
     plate[0]>= "0" and plate[0] <= "9" and \
     plate[1]>= "0" and plate[1] <= "9" and \
     plate[2]>= "0" and plate[2] <= "9" and \
     plate[3]>= "0" and plate[3] <= "9" and \
     plate[4]>= "A" and plate[4] <= "Z" and \
     plate[5]>= "A" and plate[5] <= "Z" and \
     plate[6]>= "A" and plate[6] <= "Z":
```

```
    print("The plate is a valid newer style plate.")
else:
    print("The plate is not valid")
```

```
Enter the license plate: 0988AXX
The plate is a valid newer style plate.
```

In [26]:
```
plate = input("Enter the license plate: ")
if len(plate) ==6 and \
     plate[0]>= "A" and plate[0] <= "Z" and \
     plate[1]>= "A" and plate[1] <= "Z" and \
     plate[2]>= "A" and plate[2] <= "Z" and \
     plate[3]>= "0" and plate[3] <= "9" and \
     plate[4]>= "0" and plate[4] <= "9" and \
     plate[5]>= "0" and plate[5] <= "9":
    print("The plate is a valid older style plate. ")
elif len(plate)==7 and \
     plate[0]>= "0" and plate[0] <= "9" and \
     plate[1]>= "0" and plate[1] <= "9" and \
     plate[2]>= "0" and plate[2] <= "9" and \
     plate[3]>= "0" and plate[3] <= "9" and \
     plate[4]>= "A" and plate[4] <= "Z" and \
     plate[5]>= "A" and plate[5] <= "Z" and \
     plate[6]>= "A" and plate[6] <= "Z":
    print("The plate is a valid newer style plate.")
else:
    print("The plate is not valid")
```

```
Enter the license plate: 5343DFD
The plate is a valid newer style plate.
```

Task 62: Roulette Payouts

A roulette wheel has 38 spaces on it. Of these spaces, 18 are black, 18 are red, and two are green. The green spaces are numbered 0 and 00. The red spaces are numbered 1, 3, 5, 7, 9, 12, 14, 16, 18, 19, 21, 23, 25, 27, 30 32, 34 and 36. The remaining integers between 1 and 36 are used to number the black spaces. Many different bets can be placed in roulette. We will only consider the following subset of them in this exercise: • Single number (1 to 36, 0, or 00) • Red versus Black • Odd versus Even (Note that 0 and 00 do not pay out for even) • 1 to 18 versus 19 to 36 Write a program that simulates a spin of a roulette wheel by using Python's random number generator. Display the number that was selected and all of the bets that must be payed. For example, if 13 is selected then your program should display: The spin resulted in 13... Pay 13 Pay Black Pay Odd Pay 1 to 18 If the simulation results in 0 or 00 then your program should display Pay 0 or Pay 00 without any further output

In [41]:
```
import random

# Define the spaces on the roulette wheel
green_spaces = [0, 00]
red_spaces = [i for i in range(1, 36, 2)] + [i for i in range(12, 30, 2)]
black_spaces = [i for i in range(2, 36, 2)] + [i for i in range(11, 30, 2)]

# Simulate a spin of the roulette wheel
spin = random.randint(0, 37)

# Display the selected number
if spin in green_spaces:
    print("The spin resulted in", spin, "...")
    print("Pay", spin)
    print("Pay Green")
```

```
elif spin in red_spaces:
    print("The spin resulted in", spin, "...")
    print("Pay", spin)
    print("Pay Red")
    if spin % 2 == 1:
        print("Pay Odd")
    else:
        print("Pay Even")
    if spin <= 18:
        print("Pay 1 to 18")
    else:
        print("Pay 19 to 36")
else:
    print("The spin resulted in", spin, "...")
    print("Pay", spin)
    print("Pay Black")
    if spin % 2 == 0:
        print("Pay Even")
    else:
        print("Pay Odd")
    if spin <= 18:
        print("Pay 1 to 18")
    else:
        print("Pay 19 to 36")
```

```
The spin resulted in 0 ...
Pay 0
Pay Green
```

In [42]:

```
from random import randrange
# Simulate spinning wheel, using 37 to represent 00
value = randrange(0,38)
if value ==37:
    print("The spin resulted in 00...")
else:
    print("The spin resulted in %d..." %value)
# Display the payout for a singe number
if value ==37:
    print("Pay 00")
else:
    print("Pay", value)

# Display the color payout
# The first line in the condition checks for 1, 3, 5, 7 and 9
# The second line in the condition checks for 12, 14, 16 and 18
# The third line in the condition checks for 19, 21, 23, 25 and 27
# The fourth line in the condition checks for 30, 32, 34 and 36
if value % 2 == 1 and value >= 1 and value <= 9 or \
value % 2 == 0 and value >= 12 and value <= 18 or \
value % 2 == 1 and value >= 19 and value <= 27 or \
value % 2 == 0 and value >= 30 and value <= 36:


    print("Pay Red")
elif value == 0 or value == 37:
    pass
else:
    print("Pay Black")
# Display the odd vs. even payout is no work to be performed.
if value >= 1 and value <= 36:
    if value % 2 == 1:
        print("Pay Odd")
    else:
        print("Pay Even")
```

```
# Display the lower numbers vs. upper numbers payout
if value >= 1 and value <= 18:
          print("Pay 1 to 18")
elif value >= 19 and value <= 36:
          print("Pay 19 to 36")
```

```
The spin resulted in 14...
Pay 14
Pay Red
Pay Even
Pay 1 to 18
```

In [1]:

```
import random

# Define the spaces on the roulette wheel
green_spaces = [0, 00]
red_spaces = [1, 3, 5, 7, 9, 12, 14, 16, 18, 19, 21, 23, 25, 27, 30, 32, 34, 36]
black_spaces = [i for i in range(2, 36) if i not in red_spaces]

# Simulate a spin of the roulette wheel
spin = random.randint(0, 37)

# Display the selected number and all of the bets that must be paid
if spin in green_spaces:
    print("The spin resulted in", spin, "...")
    print("Pay", spin)
    if spin == 0:
        print("Pay 0")
    else:
        print("Pay 00")
elif spin in red_spaces:
    print("The spin resulted in", spin, "...")
    print("Pay", spin)
    print("Pay Red")
    if spin % 2 == 1:
        print("Pay Odd")
    else:
        print("Pay Even")
    if spin <= 18:
        print("Pay 1 to 18")
    else:
        print("Pay 19 to 36")
else:
    print("The spin resulted in", spin, "...")
    print("Pay", spin)
    print("Pay Black")
    if spin % 2 == 0:
        print("Pay Even")
    else:
        print("Pay Odd")
    if spin <= 18:
        print("Pay 1 to 18")
    else:
        print("Pay 19 to 36")
```

```
The spin resulted in 22 ...
Pay 22
Pay Black
Pay Even
Pay 19 to 36
```

Task 63: Average

In this exercise you will create a program that computes the average of a collection of values entered by the user. The user will enter 0 as a sentinel value to indicate that no further values will be provided. Your program should display an appropriate error message if the first value entered by the user is 0

In [12]:
```
count = 0
total = 0
value = float(input("Enter a value (0 to quit): "))
while value != 0:
    count += 1
    total += value
    value = float(input("Enter a value (0 to quit): "))
    if count == 1 and value == 0:
        print("Error: No values were entered.")
        break
if count > 0:
    average = total / count
    print("The average is:", average)
```

```
Enter a value (0 to quit): 576.88888888888888888
Enter a value (0 to quit): 87687.9879868
Enter a value (0 to quit): 6787.67687687687
Enter a value (0 to quit): 0
The average is: 31684.184584188584
```

In [13]:
```
count = 0
total = 0
value = input("Enter a value (0 to quit): ")
while value != "0":
    if value == "":
        print("Error: No values were entered.")
        break
    count += 1
    total += float(value)
    value = input("Enter a value (0 to quit): ")
if count > 0:
    average = total / count
    print("The average is:", average)
```

```
Enter a value (0 to quit): 4566.8798
Enter a value (0 to quit):
Error: No values were entered.
The average is: 4566.8798
```

Task 64: Discount Table

A particular retailer is having a 60 percent off sale on a variety of discontinued products. The retailer would like to help its customers determine the reduced price of the merchandise by having a printed discount table on the shelf that shows the original prices and the prices after the discount has been applied. Write a program that uses a loop to generate this table, showing the original price, the discount amount, and the new price for purchases of 4.95, 9.95, 14.95,19.95 and $24.95. Ensure that the discount amounts and the new prices are rounded to 2 decimal places when they are displayed

In [2]:
```
prices = [4.95, 9.95, 14.95, 19.95, 24.95]
discount = 0.6

print("Original Price\tDiscount Amount\tNew Price")
for price in prices:
    discount_amount = round(price * discount, 2)
```

```
    new_price = round(price - discount_amount, 2)
    print("${}\t\t${}\t\t${}".format(price, discount_amount, new_price))
```

```
Original Price  Discount Amount New Price
$4.95           $2.97           $1.98
$9.95           $5.97           $3.98
$14.95          $8.97           $5.98
$19.95          $11.97          $7.98
$24.95          $14.97          $9.98
```

In [14]:
```
prices = [23.89, 90.89, 778.34, 11.23]
discount = 0.6
print("Original Price\t Discount Amount\t New Price")
for price in prices:
    discount_amount = round(price * discount, 2)
    new_price = round(price- discount_amount, 2)
    print("${}\t ${}\t ${}\t". format(price, discount_amount, new_price))
```

```
Original Price   Discount Amount         New Price
$23.89   $14.33  $9.56
$90.89   $54.53  $36.36
$778.34  $467.0  $311.34
$11.23   $6.74   $4.49
```

Task 65: Temperature Conversion Table Write a program that displays a temperature conversion table for degrees Celsius and degrees Fahrenheit. The table should include rows for all temperatures between 0 and 100 degrees Celsius that are multiples of 10 degrees Celsius. Include appropriate headings on your columns. The formula for converting between degrees Celsius and degrees Fahrenheit can be found on the Internet.

In [3]:
```
print("Celsius\tFahrenheit")
for celsius in range(0, 101, 10):
    fahrenheit = (celsius * 9/5) + 32
    print("{}\t{}".format(celsius, fahrenheit))
```

```
Celsius Fahrenheit
0       32.0
10      50.0
20      68.0
30      86.0
40      104.0
50      122.0
60      140.0
70      158.0
80      176.0
90      194.0
100     212.0
```

In [20]:
```
print("Celsius\t Fahrenheit")
degree_of_Celsius = [21, 34, 43, 78,90, 22, 56, 88]
for celsius in degree_of_Celsius:
    fahrenheit = (celsius * 9/5) + 32
    print("{}\t    {}".format(celsius ,fahrenheit))
```

```
Celsius  Fahrenheit
21          69.8
34          93.2
43          109.4
78          172.4
90          194.0
22          71.6
56          132.8
88          190.4
```

In [ ]:

Task 66

February 4, 2013 was the last day that pennies were distributed by the Royal Canadian Mint. Now that pennies have been phased out retailers must adjust totals so that they are multiples of 5 cents when they are paid for with cash (credit card and debit card transactions continue to be charged to the penny). While retailers have some freedom in how they do this, most choose to round to the closest nickel. Write a program that reads prices from the user until a blank line is entered. Display the total cost of all the entered items on one line, followed by the amount due if the customer pays with cash on a second line. The amount due for a cash payment should be rounded to the nearest nickel. One way to compute the cash payment amount is to begin by determining how many pennies would be needed to pay the total. Then compute the remainder when this number of pennies is divided by 5. Finally, adjust the total down if the remainder is less than 2.5. Otherwise adjust the total up

In [10]:
```
total_cost = 0
while True:
    price = input("Enter the price of an item (press enter to quit): ")
    if price == "":
        break
    total_cost += float(price)

print("Total cost: ${:.2f}".format(total_cost))
cash_payment = round(total_cost / 0.05) * 0.05
print("Amount due (cash payment): ${:.2f}".format(cash_payment))
```

```
Enter the price of an item (press enter to quit): 6.8908978776
Enter the price of an item (press enter to quit): 6.879879867866
Enter the price of an item (press enter to quit): 9.76876564354354
Enter the price of an item (press enter to quit): 0.897864676546546546543433424328
98979878
Enter the price of an item (press enter to quit):
Total cost: $24.44
Amount due (cash payment): $24.45
```

In [15]:
```
total_cost = 0
while True:
    price = input("Enter the price of an item( press enter to quit):")
    if price =="":
        break
    total_cost +=float(price)
print("Total cost: ${:.2f}". format(total_cost))
cash_payment = round(total_cost)
print("Amount due (cash payment): ${:.2f}".format(cash_payment))
```

```
Enter the price of an item( press enter to quit):32.99999999999999
Enter the price of an item( press enter to quit):789.77777777777777777777777
Enter the price of an item( press enter to quit):0.9987879899
Enter the price of an item( press enter to quit):
Total cost: $823.78
Amount due (cash payment): $823.80
```

Task 67 Compute the Perimeter of a Polygon

Write a program that computes the perimeter of a polygon. Begin by reading the x and y coordinates for the first point on the perimeter of the polygon from the user. Then continue reading pairs of values until the user enters a blank line for the x-coordinate. Each time you

read an additional coordinate you should compute the distance to the previous point and add it to the perimeter. When a blank line is entered for the x-coordinate your program should add the distance from the last point back to the first point to the perimeter. Then the perimeter should be displayed. Sample input and output values are shown below. The input values entered by the user are shown in bold

In [15]:
```
import math

# Read the first point
x1, y1 = map(float, input("Enter the x and y coordinates of the first point: ").spl

# Initialize the perimeter and previous point
perimeter = 0
x_prev, y_prev = x1, y1

# Read the remaining points
while True:
    # Read the next point
    point = input("Enter the x and y coordinates of the next point (blank line to q
    if point == "":
        break
    x, y = map(float, point.split())

    # Compute the distance to the previous point and add it to the perimeter
    distance = math.sqrt((x - x_prev) ** 2 + (y - y_prev) ** 2)
    perimeter += distance

    # Update the previous point
    x_prev, y_prev = x, y

# Compute the distance from the last point back to the first point and add it to th
distance = math.sqrt((x1 - x_prev) ** 2 + (y1 - y_prev) ** 2)
perimeter += distance

# Display the perimeter
print("The perimeter of the polygon is:", perimeter)
```

```
Enter the x and y coordinates of the first point: -3 4
Enter the x and y coordinates of the next point (blank line to quit): 0 0
Enter the x and y coordinates of the next point (blank line to quit): -3 0
Enter the x and y coordinates of the next point (blank line to quit):
The perimeter of the polygon is: 12.0
```

In [28]:
```
from math import sqrt

# Store the perimeter of the polygon
perimeter = 0

# Read the first coordinate of the polygon
first_x = float(input("Enter the first x-coordinate: "))
first_y = float(input("Enter the first y-coordinate: "))

# Provide initial value for prev_x and prev_y
prev_x = first_x
prev_y = first_y

# Read the remaining coordinates
line = input("Enter the next x-coordinate (blank to quit): ")
while line != "":
    # Convert the x-coordinate to a number and read the y-coordinate
    x = float(line)
    y = float(input("Enter the next y-coordinate: "))
```

```
    # Compute the distance between the current and previous points
    dist = sqrt((prev_x - x) ** 2 + (prev_y - y) ** 2)
    perimeter += dist

    # Set up prev_x and prev_y for the next loop iteration
    prev_x = x
    prev_y = y

    # Read the next x-coordinate
    line = input("Enter the next x-coordinate (blank to quit): ")



# Compute the distance from the last point back to the first point
dist = sqrt((first_x - x) ** 2 + (first_y - y) ** 2)
perimeter += dist

# Display the perimeter of the polygon
print("The perimeter of the polygon is:", perimeter)
```

```
Enter the first x-coordinate: -3
Enter the first y-coordinate: -4
Enter the next x-coordinate (blank to quit): 0
Enter the next y-coordinate: 4
Enter the next x-coordinate (blank to quit):
The perimeter of the polygon is: 17.08800749063506
```

In [25]:

```
from math import sqrt

# Store the perimeter of the polygon
perimeter = 0

# Read the first coordinate of the polygon
first_x = float(input("Enter the first x-coordinate: "))
first_y = float(input("Enter the first y-coordinate: "))

# Provide initial value for prev_x and prev_y
prev_x = first_x
prev_y = first_y

# Read the remaining coordinates
line = input("Enter the next x-coordinate (blank to quit): ")
while line != "":
    # Convert the x-coordinate to a number and read the y-coordinate
    x = float(line)
    y = float(input("Enter the next y-coordinate: "))

    # Compute the distance between the current and previous points
    dist = sqrt((prev_x - x) ** 2 + (prev_y - y) ** 2)
    perimeter += dist

    # Set up prev_x and prev_y for the next loop iteration
    prev_x = x
    prev_y = y

    # Read the next x-coordinate
    line = input("Enter the next x-coordinate (blank to quit): ")

# Compute the distance from the last point back to the first point
dist = sqrt((first_x - x) ** 2 + (first_y - y) ** 2)
perimeter += dist
```

```
# Display the perimeter of the polygon
print("The perimeter of the polygon is:", perimeter)
```

```
Enter the first x-coordinate: -4
Enter the first y-coordinate: -3
Enter the next x-coordinate (blank to quit): 0
Enter the next y-coordinate: -3
Enter the next x-coordinate (blank to quit):
The perimeter of the polygon is: 8.0
```

Task 68: Compute a Grade Point Average

Exercise 52 includes a table that shows the conversion from letter grades to grade points at a particular academic institution. In this exercise you will compute the grade point average of an arbitrary number of letter grades entered by the user. user will enter a blank line to indicate that all of the grades have been provided. For example, if the user enters A, followed by C+, followed by B, followed by a blank line then your program should report a grade point average of 3.1. You may find your solution to Exercise 52 helpful when completing this exercise. Your program does not need to do any error checking. It can assume that each value entered by the user will always be a valid letter grade or a blank line.

In [29]:
```
# Create a dictionary to map letter grades to grade points
grade_points = {'A+': 4.0, 'A': 4.0, 'A-': 3.7, 'B+': 3.3, 'B': 3.0, 'B-': 2.7, 'C-

# Initialize the total grade points and number of grades
total_grade_points = 0
num_grades = 0

# Read the letter grades from the user
letter_grade = input("Enter a letter grade (blank line to quit): ")
while letter_grade != "":
    # Convert the letter grade to grade points
    if letter_grade in grade_points:
        grade_point = grade_points[letter_grade]

#         In the provided code, grade_point = grade_points[letter_grade]
# is used to retrieve the grade point value from the grade_points dictionary
# that corresponds to the letter grade entered by the user.
# The square brackets [] are used to access the value
# associated with the key letter_grade in the dictionary grade_points.
# This value is then assigned to the variable grade_point.

        total_grade_points += grade_point
        num_grades += 1
    else:
        print("Invalid letter grade")

    # Read the next letter grade from the user
    letter_grade = input("Enter a letter grade (blank line to quit): ")

# Calculate the grade point average
if num_grades > 0:
    gpa = total_grade_points / num_grades
    print("The grade point average is:", gpa)
else:
    print("No valid grades entered")
```

```
Enter a letter grade (blank line to quit): A-
Enter a letter grade (blank line to quit): A
Enter a letter grade (blank line to quit): B
Enter a letter grade (blank line to quit): D
Enter a letter grade (blank line to quit): A+
Enter a letter grade (blank line to quit):
The grade point average is: 3.1399999999999997
```

Task 69: A particular zoo determines the price of admission based on the age of the guest. Guests 2 years of age and less are admitted without charge. Children between 3 and 12 years of age cost $14.00. Senior aged 65 years and over cost 18.00. Admission for all other guests is $23.00. Create a program that begins by reading the ages of all of the guests in a group from the user, with one age entered on each line. The user will enter a blank line to indicate that there are no more guests in the group. Then your program should display the admission cost for the group with an appropriate message. The cost should be displayed using two decimal places

In [30]:
```
# Initialize the admission cost to 0
admission_cost = 0

# Read the age of each guest from the user
while True:
    age_str = input("Enter the age of a guest (blank to quit): ")
    if age_str == "":
        break
    age = int(age_str)

    # Calculate the admission cost based on the age
    if age <= 2:
        admission_cost += 0
    elif age <= 12:
        admission_cost += 14
    elif age >= 65:
        admission_cost += 18
    else:
        admission_cost += 23

# Display the admission cost for the group
print("The admission cost for the group is: ${:.2f}".format(admission_cost))
```

```
Enter the age of a guest (blank to quit): 1
Enter the age of a guest (blank to quit): 27
Enter the age of a guest (blank to quit): 31
Enter the age of a guest (blank to quit): 45
Enter the age of a guest (blank to quit): 34
Enter the age of a guest (blank to quit):
The admission cost for the group is: $92.00
```

Task 70: Parity Bits

A parity bit is a simple mechanism for detecting errors in data transmitted over an unreliable connection such as a telephone line. The basic idea is that an additional bit is transmitted after each group of 8 bits so that a single bit error in the transmission can be detected. Parity bits can be computed for either even parity or odd parity. If even parity is selected then the parity bit that is transmitted is chosen so that the total number of one bits transmitted (8 bits of data plus the parity bit) is even. When odd parity is selected the parity bit is chosen so that the total number of one bits transmitted is odd. Write a program that computes the parity bit for groups of 8 bits entered by the user using even parity. Your

program should read strings containing 8 bits until the user enters a blank line. After each string is entered by the user your program should display a clear message indicating whether the parity bit should be 0 or 1. Display an appropriate error message if the user enters something other than 8 bits. Hint: You should read the input from the user as a string. Then you can use the count method to help you determine the number of zeros and ones in the string. Information about the count method is available online.

In [31]:
```
# Read the 8-bit strings from the user
while True:
    bits = input("Enter 8 bits (blank to quit): ")
    if bits == "":
        break

    # Check that the input is valid
    if len(bits) != 8 or not all(bit in "01" for bit in bits):
        print("Invalid input")
        continue

    # Count the number of ones in the string
    num_ones = bits.count("1")

    # Determine the parity bit
    if num_ones % 2 == 0:
        parity_bit = 0
    else:
        parity_bit = 1

    # Display the parity bit
    print("The parity bit should be:", parity_bit)
```

```
Enter 8 bits (blank to quit): 87987078
Invalid input
Enter 8 bits (blank to quit): d
Invalid input
Enter 8 bits (blank to quit): 0000000
Invalid input
Enter 8 bits (blank to quit): 11111111
The parity bit should be: 0
Enter 8 bits (blank to quit): 00000000
The parity bit should be: 0
Enter 8 bits (blank to quit): 11111111
The parity bit should be: 0
Enter 8 bits (blank to quit):
```

Task 71: Approximate π

The value of π can be approximated by the following infinite series: $π ≈ 3 + 4/ 2 × 3 × 4

4/ 4 × 5 × 6 + 4/ 6 × 7 × 8

4/ 8 × 9 × 10 + 4/ 10 × 11 × 12 − · · ·$ Write a program that displays 15 approximations of π. The first approximation should make use of only the first term from the infinite series. Each additional approximation displayed by your program should include one more term in the series, making it a better approximation of π than any of the approximations displayed previously

In [23]:
```
pi = 3
denominator = 2
for i in range(15):
```

```
    pi += 4 / (denominator * (denominator + 1) * (denominator + 2))
    denominator += 2
    print("Approximation", i+1, ":", pi)
```

```
Approximation 1 : 3.1666666666666665
Approximation 2 : 3.1999999999999997
Approximation 3 : 3.2119047619047616
Approximation 4 : 3.217460317460317
Approximation 5 : 3.22049062049062
Approximation 6 : 3.222322122322122
Approximation 7 : 3.223512598512598
Approximation 8 : 3.2243295919766504
Approximation 9 : 3.2249143872982877
Approximation 10 : 3.225347287731188
Approximation 11 : 3.2256766684953515
Approximation 12 : 3.225933078751762
Approximation 13 : 3.2261365789552623
Approximation 14 : 3.226300782567742
Approximation 15 : 3.2264351911698923
```

Task72: Fizz-Buzz

Fizz-Buzz is a game that is sometimes played by children to help them learn about division. The players are commonly arranged in a circle so that the game can progress from player to player continually. The starting player begins by saying one, and then play passes to the player to the left. Each subsequent player is responsible for the next integer in sequence before play passes to the following player. On a player's turn they must either say their number or one of following substitutions: • If the player's number is divisible by 3 then the player says fizz instead of their number. • If the player's number is divisible by 5 then the player says buzz instead of their number. A player must say both fizz and buzz for numbers that are divisible by both 3 and 5. Any player that fails to perform the correct substitution or hesitates before answering is eliminated from the game. The last player remaining is the winner. Write a program that displays the answers for the first 100 numbers in the FizzBuzz game. Each answer should be displayed on its own line.

In [27]:
```
for number in range(1,101):
    if i % 3==0 and i % 5==0:
        print("Fizz Buzz")
    elif i % 3==0:
        print("Fizz")
    elif i % 5==0:
        print("Buzz")
    else:
        print(i)
```

56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56

```
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
56
```

In [26]:

```python
number = int(input("Enter any number smaller than 100: "))
output = ""
for i in range(1, number+1):
    if i % 3 == 0 and i % 5 == 0:
        output += "FizzBuzz\n"
    elif i % 3 == 0:
        output += "Fizz\n"
    elif i % 5 == 0:
        output += "Buzz\n"
    else:
        output += str(i) + "\n"
print(output)
```

```
Enter any number smaller than 100: 56
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
19
Buzz
Fizz
22
23
Fizz
Buzz
26
Fizz
28
29
FizzBuzz
31
32
Fizz
34
Buzz
Fizz
37
38
Fizz
Buzz
41
Fizz
43
44
FizzBuzz
46
47
Fizz
49
Buzz
Fizz
52
53
Fizz
Buzz
56
```

Task 73: Caesar Cipher

One of the first known examples of encryption was used by Julius Caesar. Caesar needed to provide written instructions to his generals, but he didn't want his enemies to learn his plans

if the message slipped into their hands. As a result, he developed what later became known as the Caesar cipher. The idea behind this cipher is simple (and as such, it provides no protection against modern code breaking techniques). Each letter in the original message is shifted by 3 places. As a result, A becomes D, B becomes E, C becomes F, D becomes G, etc. The last three letters in the alphabet are wrapped around to the beginning: X becomes A, Y becomes B and Z becomes C. Non-letter characters are not modified by the cipher. Write a program that implements a Caesar cipher. Allow the user to supply the message and the shift amount, and then display the shifted message. Ensure that your program encodes both uppercase and lowercase letters. Your program should also support negative shift values so that it can be used both to encode messages and decode messages.

In [28]:
```
def caesar_cipher(message, shift):
    result = ""
    for char in message:
        if char.isalpha():
            if char.isupper():
                result += chr((ord(char) + shift - 65) % 26 + 65)
            else:
                result += chr((ord(char) + shift - 97) % 26 + 97)
        else:
            result += char
    return result

message = input("Enter the message: ")
shift = int(input("Enter the shift amount: "))
encrypted_message = caesar_cipher(message, shift)
print("The encrypted message is:", encrypted_message)
```

```
Enter the message: Hello hero
Enter the shift amount: 5
The encrypted message is: Mjqqt mjwt
```

In [3]:
```
message = input("Enter the message: ")
shift = int(input("Enter the shift value: "))
new_message = ""
for ch in message:
    if ch >= "a" and ch <= "z":
        pos = ord(ch) - ord("a")
        pos = (pos +shift ) % 26
        new_char = chr(pos + ord("a"))
        new_message = new_message + new_char
    elif ch >= "A" and ch <= "Z":
        pos = ord(ch) - ord("A")
        pos = (pos +shift ) % 26
        new_char = chr(pos + ord("A"))
        new_message = new_message + new_char
    else:
        new_message = new_message + ch
    print("The shifted message is", new_message)
```

```
Enter the message: Hello Hero
Enter the shift value: 5
The shifted message is M
The shifted message is Mj
The shifted message is Mjq
The shifted message is Mjqq
The shifted message is Mjqqt
The shifted message is Mjqqt
The shifted message is Mjqqt M
The shifted message is Mjqqt Mj
The shifted message is Mjqqt Mjw
The shifted message is Mjqqt Mjwt
```

In [5]:
```
message = input("Enter the message: ")
shift = int(input("Enter the shift value: "))
new_message = ""
for ch in message:
    if ch >= "a" and ch <= "z":
        pos = ord(ch) - ord("a")
        pos = (pos +shift ) % 26
        new_char = chr(pos + ord("a"))
        new_message = new_message + new_char
    elif ch >= "A" and ch <= "Z":
        pos = ord(ch) - ord("A")
        pos = (pos +shift ) % 26
        new_char = chr(pos + ord("A"))
        new_message = new_message + new_char
    else:
        new_message = new_message + ch
print("The shifted message is: ", new_message)
```

```
Enter the message: Hello Hero
Enter the shift value: 5
The shifted message is:  Mjqqt Mjwt
```

Task 74: Write a program that implements Newton's method to compute and display the square root of a number, x, entered by the user. The algorithm for Newton's method follows: Read x from the user Initialize guess to x/2 While guess is not good enough do Update guess to be the average of guess and x/guess When this algorithm completes, guess contains an approximation of the square root of x. The quality of the approximation depends on how you define "good enough". In the author's solution, guess was considered good enough when the absolute value of the difference between guess * guess and x was less than or equal to 10**(−12)

In [8]:
```
def newton_sqrt(x):
    guess = x / 2
    while abs(guess * guess - x) > 10**(-12):
        guess = (guess + x / guess) / 2
    return guess

x = float(input("Enter a number: "))
sqrt_x = newton_sqrt(x)
print("The square root of", x, "is", sqrt_x)
```

```
Enter a number: 37
The square root of 37.0 is 6.08276253029822
```

In [9]:
```
def newton_sqrt(x):
    guess = x /2
    while abs(guess*guess - x)> 10**(-12):
```

```
        guess = (guess + x/guess)/2
    return guess
x = float(input("Enter a number: "))
sqrt_x = newton_sqrt(x)
print("The square root of", x , "is", sqrt_x)
```

```
Enter a number: 45677565456.6476459999999999
The square root of 45677565456.647644 is 213723.10463926833
```

Task 75: Is a string a Palindrome?

A string is a palindrome if it is identical forward and backward. For example “anna”, “civic”, “level” and “hannah” are all examples of palindromic words.Write a program that reads a string from the user and uses a loop to determine whether or not it is a palindrome. Display the result, including a meaningful output message

In [12]:
```
line = input("Enter a string: ")
is_palindrome = True
i =0
while i < len(line) /2 and is_palindrome:
    if line[i] !=line[len(line)-i-1]:
        is_palindrome = False
    i = i + 1

if is_palindrome:
    print(line,"is a palindrome")
else:
    print(line,"is not a palindrome")
```

```
Enter a string: anna
anna is a palindrome
```

Task 76 Multiple Word Palindromes There are numerous phrases that are palindromes when spacing is ignored. Examples include “go dog”, “flee to me remote elf” and “some men interpret nine memos”, among many others. Extend your solution to Exercise 75 so that it ignores spacing while determiningwhether or not a string is a palindrome. For an additional challenge, further extend your solution so that is also ignores punctuation marks and treats uppercase and lowercase letters as equivalent.

In [3]:
```
def is_palindrome(string):
    # Remove whitespace and convert to lowercase
    new = ''.join(string.split()).lower()
    # Check if the resulting string is a palindrome
    return new == new[::-1]
print(is_palindrome("go dog"))  # True
print(is_palindrome("flee to me remote elf"))  # True
print(is_palindrome("some men interpret nine memos"))  # True
print(is_palindrome("A man, a plan, a canal, Panama!"))  # True
print(is_palindrome("not a palindrome"))  # False
```

```
True
True
True
False
False
```

Task 77 In this exercise you will create a program that displays a multiplication table that shows the products of all combinations of integers from 1 times 1 up to and including 10 times 10. Your multiplication table should include a row of labels across the top of it containing the numbers 1 through 10. It should also include labels down the left side

consisting of the numbers 1 through 10. The expected output from the program is shown below: When completing this exercise you will probably find it helpful to be able to print out a value without moving down to the next line. This can be accomplished by added end="" as the last argument to your print statement. For example, print("A") will display the letter A and then move down to the next line. The statement print("A", end="") will display the letter A without moving down to the next line, causing the next print statement to display its result on the same line as the letter A.

In [4]:
```
# Print the column labels
print("   ", end="")
for i in range(1, 11):
    print(f"{i:4}", end="")  # Use a field width of 4 for alignment
print()

# Print the table
for i in range(1, 11):
    # Print the row label
    print(f"{i:2}", end="")

    # Print the products
    for j in range(1, 11):
        print(f"{i * j:4}", end="")  # Use a field width of 4 for alignment

    # Move to the next line
    print()
```

```
     1   2   3   4   5   6   7   8   9  10
 1   1   2   3   4   5   6   7   8   9  10
 2   2   4   6   8  10  12  14  16  18  20
 3   3   6   9  12  15  18  21  24  27  30
 4   4   8  12  16  20  24  28  32  36  40
 5   5  10  15  20  25  30  35  40  45  50
 6   6  12  18  24  30  36  42  48  54  60
 7   7  14  21  28  35  42  49  56  63  70
 8   8  16  24  32  40  48  56  64  72  80
 9   9  18  27  36  45  54  63  72  81  90
10  10  20  30  40  50  60  70  80  90 100
```

Task 78:The Collatz Conjecture

Consider a sequence of integers that is constructed in the following manner: Start with any positive integer as the only term in the sequence While the last term in the sequence is not equal to 1 do If the last term is even then Add another term to the sequence by dividing the last term by 2 using floor division Else Add another term to the sequence by multiplying the last term by 3 and adding 1 The Collatz conjecture states that this sequence will eventually end with one when it begins with any positive integer. Although this conjecture has never been proved, it appears to be true. Create a program that reads an integer, n, from the user and reports all of the values in the sequence starting with n and ending with one. Your program should allow the user to continue entering new n values (and your program should continue displaying the sequences) until the user enters a value for n that is less than or equal to zero. The Collatz conjecture is an example of an open problem in mathematics. While many people have tried to prove that it is true, no one has been able to do so. Information on other open problems in mathematics can be found on the Internet.

In [ ]:
```
```

Task 79: Greatest Common Divisor

The greatest common divisor of two positive integers, n and m, is the largest number, d, which divides evenly into both n and m. There are several algorithms that can be used to solve this problem, including: Initialize d to the smaller of m and n. While d does not evenly divide m or d does not evenly divide n do Decrease the value of d by 1 Report d as the greatest common divisor of n and m Write a program that reads two positive integers from the user and uses this algorithm to determine and report their greatest common divisor

In [ ]:

Task 80 The prime factorization of an integer, n, can be determined using the following steps: Initialize factor to 2 While factor is less than or equal to n do If n is evenly divisible by factor then Conclude that factor is a factor of n Divide n by factor using floor division Else Increase factor by 1 Write a program that reads an integer from the user. If the value entered by the user is less than 2 then your program should display an appropriate error message. Otherwise your program should display the prime numbers that can be multiplied together to compute n, with one factor appearing on each line

In [ ]:

Task 81: Binary to Decimal

Write a program that converts a binary (base 2) number to decimal (base 10). Your program should begin by reading the binary number from the user as a string. Then it should compute the equivalent decimal number by processing each digit in the binary number. Finally, your program should display the equivalent decimal number with an appropriate message.

In [ ]:

Task 82 Exercise 82: Decimal to Binary (Solved, 27 Lines) Write a program that converts a decimal (base 10) number to binary (base 2). Read the decimal number from the user as an integer and then use the division algorithm shown below to perform the conversion. When the algorithm completes, result contains the binary representation of the number. Display the result, along with an appropriate message Let result be an empty string Let q represent the number to convert repeat Set r equal to the remainder when q is divided by 2 Convert r to a string and add it to the beginning of result Divide q by 2, discarding any remainder, and store the result back into q until q is 0

In [ ]:

Task83: Maximum Integer

This exercise examines the process of identifying the maximum value in a collection of integers. Each of the integers will be randomly selected from the numbers between 1 and 100. The collection of integers may contain duplicate values, and some of the integers between 1 and 100 may not be present. Take a moment and think about how you would solve this problem on paper. Many people would check each integer in sequence and ask

themself if the number that they are currently considering is larger than the largest number that they have seen previously. If it is, then they forget the previous maximum number and remember the current number as the new maximum number. This is a reasonable approach, and will result in the correct answer when the process is performed carefully. If you were performing this task, how many times would you expect to need to update the maximum value and remember a new number? While we can answer the question posed at the end of the previous paragraph using probability theory, we are going to explore it by simulating the situation. Create a program that begins by selecting a random integer between 1 and 100. Save this integer as the maximum number encountered so far. After the initial integer has been selected, generate 99 additional random integers between 1 and 100. Check each integer as it is generated to see if it is larger than the maximum number encountered so far. If it is then your program should update the maximum number encountered and count the fact that you performed an update. Display each integer after you generate it. Include a notation with those integers which represent a new maximum. After you have displayed 100 integers your program should display the maximum value encountered, along with the number of times the maximum value was updated during the process. Partial output for the program is shown below, with... representing the remaining integers that your program will display. Run your program several times. Is the number of updates performed on the maximum value what you expected?

In [ ]:

```
```

Task84: Coin Flip Simulation

What's the minimum number of times you have to flip a coin before you can have three consecutive flips that result in the same outcome (either all three are heads or all three are tails)? What's the maximum number of flips that might be needed? How many flips are needed on average? In this exercise we will explore these questions by creating a program that simulates several series of coin flips. Create a program that uses Python's random number generator to simulate flipping a coin several times. The simulated coin should be fair, meaning that the probability of heads is equal to the probability of tails. Your program should flip simulated coins until either 3 consecutive heads of 3 consecutive tails occur. Display an H each time the outcome is heads, and a T each time the outcome is tails, with all of the outcomes for one simulation on the same line. Then display the number of flips that were needed to reach 3 consecutive occurrences of the same outcome. When your program is run it should perform the simulation 10 times and report the average number of flips needed. Sample output is shown below:

In [ ]:

```
```

Print the numbers 0-20, one number per line

In [16]:

```
for i in range(21):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

In [17]:
```
# print only ODD values form 3-29, one number per line.
for i in range(3,30,2):
    print(i)
```

```
3
5
7
9
11
13
15
17
19
21
23
25
27
29
```

In [18]:
```
# Print the EVEN numbers 12 down to -14 in descending order, one number per line
for i in range(12,-15,-2):
    print(i)
```

```
12
10
8
6
4
2
0
-2
-4
-6
-8
-10
-12
-14
```

In [19]:
```
# print the numbers 50 down to 20 in decsending order, but only if the numbers are
for i in range(50,19,-1):
    if i %3 ==0:
        print(i)
```

```
48
45
42
39
36
33
30
27
24
21
```

In [28]:
```
# Initialize two variables to hold the string 'LaunchCode' and the array [1, 5, 'LC

# Print each element of the array to a new line.

# Print each character of the string---in reverse order---to a new line.
arr = [1,5,'LC101','blue',42]
for i in arr:
    print(i)
```

```
1
5
LC101
blue
42
```

In [21]:
```
# Initialize two variables to hold the string 'LaunchCode'
# and the array [1, 5, 'LC101', 'blue', 42], then construct for
# loops to accomplish the following tasks:
# Print each character of the string---in reverse order---to
# a new line.
str = 'LaunchCode'
for i in range(len(str)-1,-1,-1):
    print(str[i])
```

```
e
d
o
C
h
c
n
u
a
L
```

In [26]:
```
# Construct a for loop that sorts the array[2,3,13,18,-5,38,-10,11,0,104] in to two

arr = [2,3,13,18,-5,38,-10,11,0,104]
evens =[]
odds = []
for i in arr:
    if i %2 ==0:
        evens.append(i)
    else:
        odds.append(i)
print("Evens:",evens)
print("Odds:",odds)
```

```
Evens: [2, 18, 38, -10, 0, 104]
Odds: [3, 13, -5, 11]
```

In [4]:
```
# Construct while loops to do the following:

# Prompt the user to enter the starting fuel level.
```

```
# The loop should continue until the user enters a positive value
# greater than 5000 but less than 30000.

fuel_level = 0
while fuel_level <=0 or fuel_level >=30000 or fuel_level <=5000:
    fuel_level= int(input("Enter the starting fuel(between 5000 and 30000)"))
```

```
Enter the starting fuel(between 5000 and 30000)6778987
Enter the starting fuel(between 5000 and 30000)22500
```

In [1]:
```
# Use a second loop to query the user for the number of
# astronauts (up to a maximum of 7). Validate the entry by having
# the loop continue until the user enters an integer from 1 - 7.
num_astronauts = 0
while num_astronauts < 1 or num_astronauts >7:
    num_astronauts = int(input("Enter the number of astronauts(1-7): "))
```

```
Enter the number of astronauts(1-7): 4
```

In [3]:
```
fuel_level = 0
while fuel_level <= 0 or fuel_level >= 30000 or fuel_level <= 5000:
    fuel_level = int(input("Enter the starting fuel level (between 5000 and 30000):
```

```
Enter the starting fuel level (between 5000 and 30000): 6700
```

In [5]:
```
# Use a final loop to monitor the fuel status and the altitude
# of the shuttle. Each iteration, decrease the fuel level by 100 units
# for each astronaut aboard. Also, increase the altitude by
# 50 kilometers. (Hint: The loop should end when there is not enough
#                  fuel to boost the
# crew another 50 km, so the fuel level might not reach 0).
fuel_level = int(input("Enter the starting fuel level: "))
num_astronauts = int(input("Enter the number of astronauts:"))
altitude =0
while fuel_level >=num_astronauts *100 and fuel_level >0:
    fuel_level -= num_astronauts*100
    altitude +=50

print("Final altitude: ", altitude)
print("Remaining fuel: ", fuel_level)
```

```
Enter the starting fuel level: 22500
Enter the number of astronauts:6
Final altitude:  1850
Remaining fuel:  300
```

In [6]:
```
# After the loops complete, output the result with the phrase, The shuttle gained a

# If the altitude is 2000 km or higher, add "Orbit achieved!"

# Otherwise add, "Failed to reach orbit."
fuel_level = int(input("Enter the starting fuel level: "))
num_astronauts = int(input("Enter the number of astronauts: "))
altitude = 0
while fuel_level >= num_astronauts * 100 and fuel_level > 0:
    fuel_level -= num_astronauts * 100
    altitude += 50

if altitude >= 2000:
    print("The shuttle gained an altitude of", altitude, "km. Orbit achieved!")
else:
    print("The shuttle gained an altitude of", altitude, "km. Failed to reach orbit
```

```
Enter the starting fuel level: 21000
Enter the number of astronauts: 5
The shuttle gained an altitude of 2100 km. Orbit achieved!
```

In [ ]: