 # Python Programming

Python is a powerful multipurpose programming language created by *Guido van Rossum*. It has a simple and easy-to-use syntax, making it a popular first-choice programming language for beginners. This is a comprehensive guide that explores the reasons you should consider learning Python and the ways you can get started with Python.

 # Interpreter Vs Compiler : Differences Between Interpreter and Compiler

In this class, you will learn the differences between interpreters and compilers. We generally write a computer program using a high-level language. A high-level language is one that is understandable by us, humans. This is called **source code**.However, a computer does not understand high-level language. It only understands the program written in **0**'s and **1**'s in binary, called the **machine code**.To convert source code into machine code, we use either a **compiler** or an **interpreter**.

What is Interpreter?

An interpreter is a computer program, which coverts each high-level program statement into the machine code. This includes source code, pre-compiled code, and scripts. Both compiler and interpreters do the same job which is converting higher level programming language to machine code. However, a compiler will convert the code into machine code (create an exe) before program run. Interpreters convert code into machine code when the program is run.

What is Compiler?

A compiler is a computer program that transforms code written in a high-level programming language into the machine code. It is a program which translates the human-readable code to a language a computer processor understands (binary 1 and 0 bits). The computer processes the machine code to perform the corresponding tasks.A compiler should comply with the syntax rule of that programming language in which it is written. However, the compiler is only a program and cannot fix errors found in that program. So, if you make a mistake, you need to make changes in the syntax of your program. Otherwise, it will not compile.

```
[ ] ## Average through for loop and append
    print ("calculate the average for the number ")
    count = int(input())
```

```
calculate the average for the number
3
```

```
num_list = [] #empty list
for i in range (count):
  print("enter the value of the position", i)
  x = int(input())
  num_list.append(x)
```

```
enter the value of the position 0
66
enter the value of the position 1
7
enter the value of the position 2
77
```

```
average = sum(num_list)/count
print(average)
```

```
50.0
```

```
### Average
print("enter the number")
a = int(input())
print("enter the number")
b = int(input())
print("enter the number")
c = int(input())
print("enter the number")
d = int(input())
print("enter the number")
e = int(input())
avg = (a+b+c+d+e)/5
print("the average is ...> ", avg )
```

```
enter the number
5
enter the number
3
enter the number
2
enter the number
6
enter the number
7
the average is ...>  4.6
```

```python
# int number printing
print("first number is")
Name = int(input(""))
```

```
first number is
1
```

```python
## Type casting
print("integer to float")
a = 33
type(a)
```

```
integer to float
int
```

```python
x = (float(a))
print(x)
```

```
33.0
```

```python
type(a)
```

```
int
```

```python
# Example:

a = 37
b = 19.16
c = 3 + 27j

#converting float to int
print (int(b))

#converting int to float
print (float(a))

#converting int to complex
print (complex(a))

#converting float to complex
print (complex(b))

#converting to complex
print (complex(a, b))
```

```
19
37.0
(37+0j)
(19.16+0j)
(37+19.16j)
```

```python
# Example:

#REMEMBER  True == 1
 #         False == 0

x = (1 == True)
y = (1 == False)
a = True + 6
b = False + 90

print("x is", x)
print("y is", y)
print("a:", a)
print("b:", b)
```

```
x is True
y is False
a: 7
b: 90
```

```python
# Example:

fruits1 = ("Banana", "Apple", "Strawberry")             # tuple ()
fruits2 = ["Banana", "Apple", "Strawberry"]             # list []
fruits3 = {"Banana", "Apple", "Strawberry"}             # set {}
fruits4 = {"1":"Banana", "2":"Apple", "3":"Strawberry"} # dictionary {"Key":"Value"}

print(fruits1)
print(fruits2)
print(fruits3)
print(fruits4)
```

```
('Banana', 'Apple', 'Strawberry')
['Banana', 'Apple', 'Strawberry']
{'Apple', 'Strawberry', 'Banana'}
{'1': 'Banana', '2': 'Apple', '3': 'Strawberry'}
```

```python
# Example:

fruits = ["apple", "mango", "orange"] #list
numbers = (1, 2, 3) #tuple
alphabets = {'a':'apple', 'b':'ball', 'c':'cat'} #dictionary
vowels = {'a', 'e', 'i' , 'o', 'u'} #set

print(fruits)
print(numbers)
print(alphabets)
print(vowels)
```

```
['apple', 'mango', 'orange']
(1, 2, 3)
{'a': 'apple', 'b': 'ball', 'c': 'cat'}
{'u', 'e', 'o', 'a', 'i'}
```

```
"""# tuple is immutable, cant be changed , Once cretaed ,
cant be changed. fixed ..
# difference between list and tuple
1. list are mutable but tuple are not
2. tuple can't be changed or modifed and they are faster in processing
and the are not heavy. if u think that ur list are not gonna changed
then u can simply utilize tuple. List are slower in processing.
3. list ur using 3rd bracket but tuple we are using 1st bracket
4. List we can change positional argument but tuple we cant
"""
```

```
l1=[1,2,3,4,5,6,7]  # list
t1=(1,2,3,4,5,6,7)  # tuple
print(l1,t1)

l1[2] = 100   # slicing
print(l1)
```

```
[1, 2, 3, 4, 5, 6, 7] (1, 2, 3, 4, 5, 6, 7)
[1, 2, 100, 4, 5, 6, 7]
```

```
l1[2:5]
```

```
[100, 4, 5]
```

```
len(l1)
```

```
7
```

```
t1[3:5]  # tuple
```

```
(4, 5)
```

```
for j in range(1,11):
  print (j)
  if j < 5:
      print ("less than 5")
  elif j > 8:
      print ("greater than 8")
  elif j > 6 and i < 8:
       print ("aaaaaa")
  else:
      print ("nothing")
```

```
1
less than 5
2
less than 5
3
less than 5
4
less than 5
5
nothing
6
nothing
7
aaaaaa
8
aaaaaa
9
greater than 8
10
greater than 8
```

```
## greater , lessthan
for i in range (1,11):
  print (i)
  if i == 6:
    break
```

```
1
2
3
4
5
6
```

```
a = [5, 10, 15, 20, 25, 30, 35, 40]  # Total elemnets is 8
#   [0   1   2   3   4   5   6   7]  ← Index forward
#   [-8 -7  -6  -5  -4  -3  -2  -1]  → Index backward

# index '0' is element '1' = 5,
# index '1' is element '2' = 10,
# index '2' is element '3' = 15,
# .
# .
# .
# index '7' is element '8' = 40,

a[1] # To access the elements in the list

# a[2] = 15
print("a[2] = ", a[2])
```

```
# change the element of the list
a = [1,2,3]
#   0,1,2 index forward
print("original list", a)
a[1] = 5
print ("Alternative list", a)
```

```
original list [1, 2, 3]
Alternative list [1, 5, 3]
```

```
list1 =  [9, 'apple', 3 + 6j]  # list
tuple1 = (9, 'apple', 3 + 6j)  # tuple

list1[1] = 'banana'  # List is mutable
print(list1)         # No error
tuple1[1]= 'banana'  # Tuple is immutable
print(tuple1)        # error
```

```
[9, 'banana', (3+6j)]
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-1-e32e417070a1> in <cell line: 6>()
      4 list1[1] = 'banana'  # List is mutable
      5 print(list1)         # No error
----> 6 tuple1[1]= 'banana'  # Tuple is immutable
      7 print(tuple1)        # error

TypeError: 'tuple' object does not support item assignment
```

```
[ ] list1 = [1,4,5] ## list - mutable

    tuple1 = (1,4,5) ## tuple - immutable

    list1[1] = 17
    # tuple1[1] = 17
    print (list1,tuple1)
```

```
[1, 17, 5] (1, 4, 5)
```

```
# Tuple 't' have 3 elements
t = (6,'program', 1+3j)
#   (0        1      2) → Index forward

# index '0' is element '1'= 6
# index '1' is element '2'= program
# index '2' is elemtnt '3'= 1+3j

# t[1] = 'program'
print("t[1] = ", t[1])

# t[0:3] = (6, 'program', (1+3j))
print("t[0:3] = ", t[0:4])

# Generates error
# Tuples are immutable
# t[0] = 10  # trying to change element 0 from '6' to '10'
```

```
t[1] =  program
t[0:3] =  (6, 'program', (1+3j))
```

```
[ ] d = {1: 'Apple', 2: 'Cat', 3: 'Food'}  # 'Apple' is element and 1 is the key of element.
    print(d, type(d))
    d[3]
```

```
{1: 'Apple', 2: 'Cat', 3: 'Food'} <class 'dict'>
'Food'
```

```
d[1]
```

```
'Apple'
```

```
[ ] d = {1:'value','key':2} # d is my variable, 'value' and 'key' are the element and 1 and 2 are the key.
    type(d)
```

```
dict
```

```
d['key']
```

```
2
```

```
[ ] d = {1:"A",2:["B","C"],3:("D","E"),4:{"F","G"},5:{8:"H",9:"I"}}
    d
```

```
{1: 'A', 2: ['B', 'C'], 3: ('D', 'E'), 4: {'F', 'G'}, 5: {8: 'H', 9: 'I'}}
```

```
[ ] d[5],type(d[5])
```

```
({8: 'H', 9: 'I'}, dict)
```

```
[ ] d[5][8]
```

```
'H'
```

```
# Example 3: program to check if a num1 is less than num2

num1, num2 = 6, 5
if (num1 < num2):
    print("num1 is less than num2")
else:
    print("num2 is less than num1")
```

```
num2 is less than num1
```

```
# Example 4: ## optional

def password_check(password):
    if password == "Python@99>":
        print("Correct password")
    else:
        print("Incorrect Password")

password_check("Python@99>")
# Output Correct password

password_check("Python99")
# Output Incorrect Password
```

```
Correct password
Incorrect Password
```

```
[ ]  x = 10.5
     if 10 < x < 11:
         print("hello")
     else:
         print("world")
```

```
hello
```

```
'''
x = 1

if x > 3:
    print ("Case 1")
if x <= 3:
    print ("Case 2")
'''

x = 1

if x > 3:
    print ("Case 1")
else:
    print ("Case 2")
```

```
Case 2
```

```
[ ]  x = 2

     if x > 4:
         print ("Correct")
     else:
         print("Incorrect")
```

```
Incorrect
```

Shortcut of `if else` (Short Hand if ... else or One line `if else`)

If you have **only one statement each for if and else, then they can be put in the sar** vn below

```
hungry = False
x = 'Feed the bear now!' if hungry else 'Do not feed the bear.'
print(x)
```

```
Do not feed the bear.
```

+ Code + Text

```
[ ]  hungry = False
     if hungry:
         x = 'Feed the bear now!'
     else:
         x = 'Do not feed the bear.'
     print (x)
```

```
Do not feed the bear.
```

```
[ ]  a = 3
     print('A is positive') if a > 0 else print('A is negative') # first condition met, 'A is positive' will be printed
```

```
A is positive
```

```
[ ]  num1, num2 = 6, 5
     print("num1 is less than num2") if (num1 < num2) else print("num2 is less than num1")
```

```
num2 is less than num1
```

```
[ ]  number = 96
     if number > 0: print("positive")
     else: print("negative")
```

positive

```
# Example 1:

'''In this program, we check if the
numberis positive or negative or zero
and display an appropriate message'''

num = -3

# Try these two variations as well:
# num = 0
# num = -4.5

if num > 0:
    print("Positive number")
elif num == 0:
    print(" value is Zero")
else:
    print("Negative number")
```

```
Negative number
```

```
# Example 4:
grade = 30
if grade >= 90:
    print("A grade")
#     print("something")
elif grade >=80:
    print("B grade")
elif grade >=70:
    print("C grade")
elif grade >= 65:
    print("D grade")
else:
    print("Failing grade")
```

```
Failing grade
```

```
# Example 5: ## optional
def user_check(choice):
    if choice == 1:
        print("Admin")
    elif choice == 2:
        print("Editor")
    elif choice == 3:
        print("Guest")
    else:
        print("Wrong entry")

user_check(1)  # Admin
user_check(2)  # Editor
user_check(3)  # Guest
user_check(4)  # Wrong entry
```

```
Admin
Editor
Guest
Wrong entry
```

Explanation:

When variable `num` is positive, `Positive number` is printed.

If `num` is equal to 0, `Zero` is printed.

If `num` is negative, `Negative number` is printed.

```
# Example 2:

num1, num2 = 5, 5
if(num1 > num2):
    print("num1 is greater than num2")
elif(num1 == num2):
    print("num1 is equal to num2")
else:
    print("num1 is less than num2")
```

```
num1 is equal to num2
```

```
# Example 3:

x = 10
y = 10
if x > y:
    print("x>y")
elif x < y:
    print("x<y")
else:
    print("x=y")
```

```
x=y
```

```
a = 3
if a > 0 and a % 2 == 0:
        print('A is an even and positive integer')
elif a > 0 and a % 2 !=  0:
     print('A is a positive integer')
elif a == 0:
    print('A is zero')
else:
    print('A is negative')
```

```
A is a positive integer
```

```
user = 'Arthur'
access_level = 4
if user == 'admin' or access_level >= 4:
        print('Access granted!')
else:
    print('Access denied!')
```

```
Access granted!
```

Nested If Statement Page:

```
# Example 1:

a=16
if a>=20:  # Condition FALSE
    print ("Condition is True")
else:  # Code will go to ELSE body
    if a>=15:  # Condition FALSE
        print ("Checking second value")
    else:  # Code will go to ELSE body
        print ("All Conditions are false")
```

```
Checking second value
```

```
# Example 2:

x = 11
y = 11
if x > y:
    print( "x>y")
elif x < y:
    print( "x<y")
    if x==10:
        print ("x=10")
    else:
        print ("invalid")
else:
    print ("x=y")
```

```
x=y
```

```
# Example 3:

num1 = -1
if (num1 != 0):  # For zero condition is FALSE
    if(num1 > 0):
        print("num1 is a positive number")
    else:
        print("num1 is a negative number")
else:  # For zero condition is TRUE
    print("num1 is neither positive nor negative")
```

```
num1 is a negative number
```

```
# Example 4:

'''In this program, we input a number check if the number is
positive or negative or zero and display an appropriate message.
This time we use nested if statement'''

num = float(input("Enter a number: "))
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```

```
Enter a number: 0
Zero
```

```
# Example 5:

def number_arithmetic(num1, num2):
    if num1 >= num2:
        if num1 == num2:
            print(f'{num1} and {num2} are equal')
        else:
            print(f'{num1} is greater than {num2}')
    else:
        print(f'{num1} is smaller than {num2}')

number_arithmetic(96, 66)
# Output 96 is greater than 66
number_arithmetic(96, 96)
# Output 56 and 56 are equal
```

```
96 is greater than 66
96 and 96 are equal
```

```python
# Example 4:
for i in range (2, 12, 2):  # beginning 2 with distance of 2 and stop before 12
    print (i)
```

```
2
4
6
8
10
```

```python
# Example 5:
num=2

for a in range (1,6):  # range (1,6) means numbers from 1 to 5, i.e., (1,2,3,4,5)
    print (num * a)
```

```
2
4
6
8
10
```

```python
# Example 1: How range works in Python?

# empty range
print(list(range(0)))

# using range(stop)
print(list(range(10)))

# using range(start, stop)
print(list(range(1, 10)))
```

```
[]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```python
# Example 1: For loop

words = ['one', 'two', 'three', 'four', 'five']

for i in words:
    print(i)
```

```
one
two
three
four
five
```

```python
# Example 2:

for num in range(10):
    print(num)
```

```
0
1
2
3
4
5
6
7
8
9
```

```python
# Example 3:
for i in range(1, 11):
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

```python
# Example 2: Calculate the average of list of numbers

numbers = [10, 20, 30, 40, 50]

# definite iteration
# run loop 5 times because list contains 5 items
sum = 0
for i in numbers:
    sum = sum + i
list_size = len(numbers)
average = sum / list_size
print(average)
```

```
30.0
```

```python
# Example 6: Find Sum of 10 Numbers

sum=0
for n in range(1,11):  # range (1,11) means numbers from 1 to 5, i.e., (1,2,3,4,5,6,7,8,9,10)
    sum+=n
    print (sum)

'''
0+1 = 1
1+2 = 3
3+3 = 6
6+4 = 10
10+5 =15
21
28
36
45
45+10 = 55
'''
```

```
1
3
6
10
15
21
28
36
45
55
'\n0+1 = 1\n1+2 = 3\n3+3 = 6\n6+4 = 10\n10+5 =15\n21\n28\n36\n45\n45+10 = 55\n'
```

```python
# Example 1:

digits = [0, 1, 5]

for i in digits:
    print(i)
else:
    print("No items left.")
```

```
0
1
5
No items left.
```

```python
# Example 1: Print all even and odd numbers

for i in range(1, 11):
    if i % 2 == 0:
        print('Even Number:', i)
    else:
        print('Odd Number:', i)
```

```
Odd Number: 1
Even Number: 2
Odd Number: 3
Even Number: 4
Odd Number: 5
Even Number: 6
Odd Number: 7
Even Number: 8
Odd Number: 9
Even Number: 10
```

```python
# Example 7: printing a series of numbers using for and range

print("Case 1:")
for i in range(5):  # Print numbers from 0 to 4
    print (i)

print("Case 2:")
for i in range(5, 10):  # Print numbers from 5 to 9
    print (i)

print("Case 3:")
for i in range(5, 10, 2):  # Print numbers from 5 with distace 2 and stop before 10
    print (i)
```

```
Case 1:
0
1
2
3
4
Case 2:
5
6
7
8
9
Case 3:
5
7
9
```

```python
# Example 2:

for number in range(11):
    print(number)   # prints 0 to 10, not including 11
else:
    print('The loop stops at', number)
```

```
0
1
2
3
4
5
6
7
8
9
10
The loop stops at 10
```

```python
# Example 3: Else block in for loop

for i in range(1, 6):
    print(i)
else:
    print("Done")
```

```
1
2
3
4
5
Done
```

```python
# Example 4:

student_name = 'Arthur'

marks = {'Alan': 99, 'Bill': 55, 'Cory': 77}

for student in marks:
    if student == student_name:
        print(marks[student])
        break
else:
    print('No entry with that name found.')
```

```
No entry with that name found.
```

```python
# Example 5:

count = 0
for i in range(1, 6):
    count = count + 1
    if count > 2:
        break
    else:
        print(i)
else:
    print("Done")
```

```
1
2
```

```python
# Example 1:

numbers = (0,1,2,3,4,5)
for number in numbers:
    print(number)
    if number == 3:
        break
```

```
0
1
2
3
```

```python
# Example 2:

color = ['Green', 'Pink', 'Blue']
for i in color:
    if(i == 'Pink'):
        break
print (i)
```

```
Pink
```

```python
numbers = [1, 4, 7, 8, 15, 20, 35, 45, 55]
for i in numbers:
    if i > 15:
        # break the loop
        break
    else:
        print(i)
```

```
1
4
7
8
15
```

```
# Example 4:

for i in range(5):
    for j in range(5):
        if j == i:
            break
        print(i, j)
```

```
1 0
2 0
2 1
3 0
3 1
3 2
4 0
4 1
4 2
4 3
```

```
# Example 1:

color = ['Green', 'Pink', 'Blue']
for i in color:
    if(i == 'Pink'):
        continue
print (i)
```

```
Blue
```

```
# Example 3:

first = [3, 6, 9]
second = [3, 6, 9]
for i in first:
    for j in second:
        if i == j:
            continue
        print(i, '*', j, '= ', i * j)
```

```
3 * 6 =  18
3 * 9 =  27
6 * 3 =  18
6 * 9 =  54
9 * 3 =  27
9 * 6 =  54
```

```
# Example 2:

numbers = (0,1,2,3,4,5)
for number in numbers:
    print(number)
    if number == 3:
        continue
    print('Next number should be ', number + 1) if number != 5 else print("loop's end")
  # for short hand conditions need both if and else statements
print('outside the loop')
```

```
0
Next number should be  1
1
Next number should be  2
2
Next number should be  3
3
4
Next number should be  5
5
loop's end
outside the loop
```

```
# Example 4:

name = "mariya mennen"
count = 0
for char in name:
    if char != 'm':
        continue
    else:
        count = count + 1

print('Total number of m is:', count)
```

```
Total number of m is: 2
```

```
# Example 1:

for number in range(6):
    pass
```

```
# Example 2:

num = [1, 4, 5, 3, 7, 8]
for i in num:
    # calculate multiplication in future if required
    pass
```

```
# Example 1: Reversed numbers using `reversed()` function

list1 = [10, 20, 30, 40]
for num in reversed(list1):
    print(num)
```

```
40
30
20
10
```

```
print("Reverse numbers using for loop")
num = 5
# start = 5
# stop = -1
# step = -1
for num in (range(num, -1, -1)):
    print(num)
```

```
Reverse numbers using for loop
5
4
3
2
1
0
```

```
# Example 1: printing a multiplication table of the first ten numbers

# outer loop
for i in range(1, 11):
    # nested loop
    for j in range(1, 11):   # to iterate from 1 to 10
        print(i * j, end=' ')    # print multiplication
    print()
```

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

```
# Example 2: Write a code to add all the prime numbers between 17 to 53 using while loop
# 17, 19, 23, 29, 31, 37, 41, 43, 47, 53

sum=0
for i in range(17,54):
    for j in range(2,i):
        if i%j ==0:
            break
    else:
        sum=sum+i
        print(i)
print(sum)
```

```
17
19
23
29
31
37
41
43
47
53
340
```

```
[ ]  # Example 3: iterating through nested for loops

     color = ['Red', 'Pink']
     element = ['flower', 'watch']
     for i in color:
         for j in element:
             print(i, j)
```

```
Red flower
Red watch
Pink flower
Pink watch
```

```
# Example 5:

numbers = [[1, 2, 3], [4, 5, 6]]

cnt = 0
for i in numbers:
    for j in i:
        print('iteration', cnt, end=': ')
        print(j)
        cnt = cnt + 1
```

```
iteration 0: 1
iteration 1: 2
iteration 2: 3
iteration 3: 4
iteration 4: 5
iteration 5: 6
```

```
# Example 4: A use case of a nested for loop in `list_of_lists` case would be

list_of_lists = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
total=0
for list1 in list_of_lists:
    for i in list1:
        total = total+i
print(total)
```

```
45
```

```
# Example 1: single line `for` loop code

first = [3, 6, 9]
second = [30, 60, 90]
final = [i+j for i in first for j in second]
print(final)
```

```
[33, 63, 93, 36, 66, 96, 39, 69, 99]
```

```
# Example 1: regular `for` loop code

first = [3, 6, 9]
second = [30, 60, 90]
final = []
for i in first:
    for j in second:
        final.append(i+j)
print(final)
```

```
[33, 63, 93, 36, 66, 96, 39, 69, 99]
```

```
# Example 2: Print the even numbers by adding 1 to the odd numbers in the list

odd = [1, 5, 7, 9]
even = [i + 1 for i in odd if i % 2 == 1]
print(even)
```

```
[2, 6, 8, 10]
```

```
# Example 3:

final = [[x, y] for x in [30, 60, 90] for y in [60, 30, 90] if x != y]
print(final)
```

```
[[30, 60], [30, 90], [60, 30], [60, 90], [90, 60], [90, 30]]
```

```python
# Example 1: Print elements of the list with its index number using the `enumerate()` function

#In this program, the for loop iterates through the list and displays the
#elements along with its index number.

numbers = [4, 2, 5, 7, 8]
for i, v in enumerate(numbers):
    print('Numbers[', i, '] =', v)
```

```
Numbers[ 0 ] = 4
Numbers[ 1 ] = 2
Numbers[ 2 ] = 5
Numbers[ 3 ] = 7
Numbers[ 4 ] = 8
```

```python
# Example 2: Printing the elements of the list with its index number using the `range()` function

numbers = [1, 2, 4, 6, 8]
size = len(numbers)
for i in range(size):
    print('Index:', i, " ", 'Value:', numbers[i])
```

```
Index: 0   Value: 1
Index: 1   Value: 2
Index: 2   Value: 4
Index: 3   Value: 6
Index: 4   Value: 8
```

```python
# Example 4: printing the elements of a list using for loop

even_numbers = [2, 4, 6, 8, 10]  # list with 5 elements
for i in even_numbers:
    print(even_numbers)
```

```
[2, 4, 6, 8, 10]
[2, 4, 6, 8, 10]
[2, 4, 6, 8, 10]
[2, 4, 6, 8, 10]
[2, 4, 6, 8, 10]
```

```python
# Example 5: printing the elements of a list using for loop

list = [60, "HelloWorld", 90.96]
for i in list:
    print(i)
```

```
60
HelloWorld
90.96
```

```python
# Example 6: Program to find the sum of all numbers stored in a list

# List of numbers
numbers = [6, 5, 3, 8, 4, 2, 5, 6, 11]  # list with 9 elements

# variable to store the sum
sum = 0

# iterate over the list
for val in numbers:
    sum = sum+val

print("The sum is", sum)
```

```
The sum is 50
```

```python
# Example 1: For loop with string

# Method 1:
language = 'Python'
for letter in language:
    print(letter)

# Method 2: using range() function

for i in range(len(language)):
    print(language[i])
```

```
P
y
t
h
o
n
P
y
t
h
o
```

```python
# Example 3: Access all characters of a string

name = "Alan"
for i in name:
    print(i, end=' ')
```

```
A l a n
```

```python
# Example 4: Iterate string in reverse order

name = "Alan"
for i in name[::-1]:
    print(i, end=' ')
```

```
n a l A
```

```python
# Example 5: Iterate over a particular set of characters in string

name = "Alan Watson"
for char in name[2:7:1]:
    print(char, end=' ')
```

```
a n   W a
```

```
# Example 8: Calculate the average of list of numbers

numbers = [10, 20, 30, 40, 50]

# definite iteration
# run loop 5 times because list contains 5 items
sum = 0
for i in numbers:
    sum = sum + i
list_size = len(numbers)
average = sum / list_size
print(average)
```

```
30.0
```

```
# Example 9: Printing a list using range function

color = ['Green', 'Pink', 'Blue']  # list with total 3 elements
print(len(color))  # print length of color
for i in range(len(color)):
    print(color[i])
```

```
3
Green
Pink
Blue
```

```
# Example 1: For loop with tuple

numbers = (0, 1, 2, 3, 4, 5)
for number in numbers:
    print(number)
```

```
0
1
2
3
4
5
```

```
# Example 6: For loop with set

mix_fruits = {'Banana', 'Apple', 'Mango', 'Orange', 'GUava', 'Kiwi', 'Grape'}
for fruits in mix_fruits:
    print(fruits)
```

```
GUava
Banana
Grape
Kiwi
Orange
Apple
Mango
```

```
[ ]  # Example 1: Access only the keys of the dictionary.

dict1 = {"Antibiotics": "Penicillin", "Inventor": "Fleming", "Year": 1928}
for key in dict1:
    print(key)
```

```
Antibiotics
Inventor
Year
```

```
# Example 2: Iterate keys and values of the dictionary

dict1 = {"Vaccine": "Polio", "Inventor": "Salk", "Year": 1953}
for key in dict1:
    print(key, "->", dict1[key])
```

```
Vaccine -> Polio
Inventor -> Salk
Year -> 1953
```

```
[ ]  # Example 3: Iterate only the values the dictionary

dict1 = {"Vaccine": "Smallpox ", "Inventor": "Jenner", "Year": 1796}
for value in dict1.values():
    print(value)
```

```
Smallpox
Jenner
1796
```

7

```
"""Question:
Consider you have been provided with three variables containing different data types:
integer_value = 42
float_value = 3.14
string_value = "Python"

Your task is to write a Python program that performs the following operations:
  • Convert the integer_value to a float and store it in a new variable called integer_to_float.
  • Convert the float_value to an integer and store it in a new variable called float_to_integer.
  • Concatenate the string_value with the float_to_integer variable, and store the result in a new variable called concatenated_string.
Finally, print the resulting variables in the following format:
Integer to float: {integer_to_float}
Float to integer: {float_to_integer}
Concatenated string: {concatenated_string}

Write the Python code to complete the given task.
"""
```

```
integer_value = 42
float_value = 3.14
string_value = "Python"
```

```
integer_to_float = float(integer_value)
print (integer_to_float), print (integer_value)
```

```
42.0
42
(None, None)
```

```
float_to_integer = int(float_value)
print (float_to_integer), print (float_value)
```

```
3
3.14
(None, None)
```

```
concatenated_string = string_value + str(integer_to_float)
concatenated_string
```

```
'Python42.0'
```

```
42.00000000
```

```
42.0
```

```
42.00004
```

```
42.00004
```

```
x =  42.357689654

round(x,4)
```

In [40]:

```
import matplotlib.pyplot as plt
```

In [41]:

```
import numpy as np
import pandas as pd
```

In [42]:

```
x = np.array([26,47,59,42,68,55,66,77,88,99])
```

In [43]:

```
y = np.array([54,65,37,48,29,22,33,36,76,73])
```

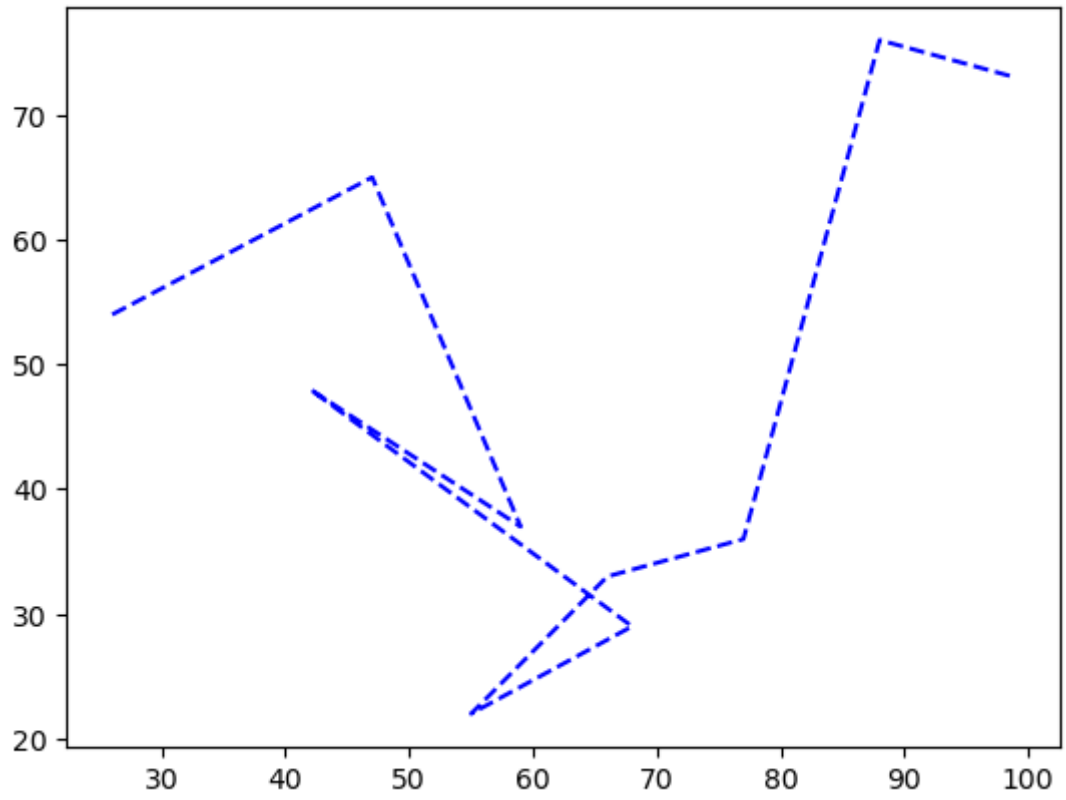
In [44]:

```
z = (x+y)/2
```

In [45]:

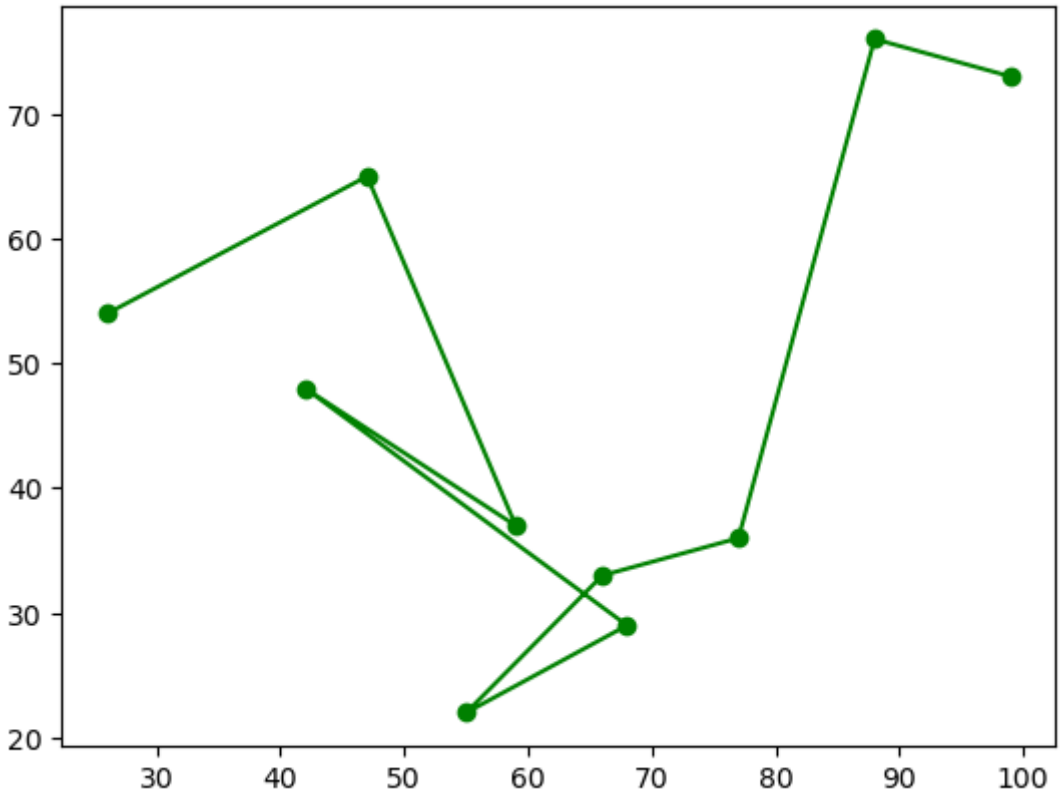
```
plt.plot(x,y,"b--", color="b")
plt.show()
```

```
C:\Users\johnb\AppData\Local\Temp\ipykernel_2504\3146585507.py:1: UserWarn
ing: color is redundantly defined by the 'color' keyword argument and the
fmt string "b--" (-> color='b'). The keyword argument will take precedenc
e.
  plt.plot(x,y,"b--", color="b")
```



In [46]:

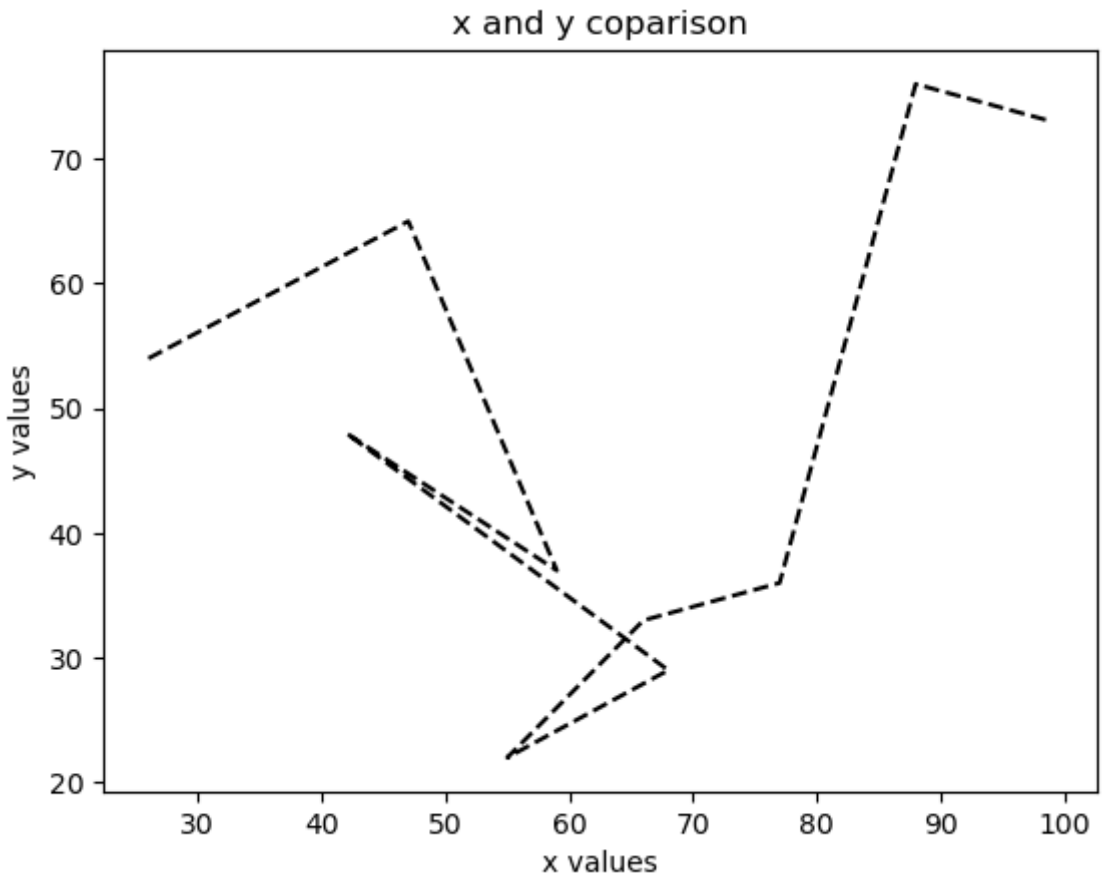
```
plt.plot(x,y,"o-", color="g")   #color g= green, b=blue, k=black, r= red, y=yellow
plt.show()
```



In [47]:

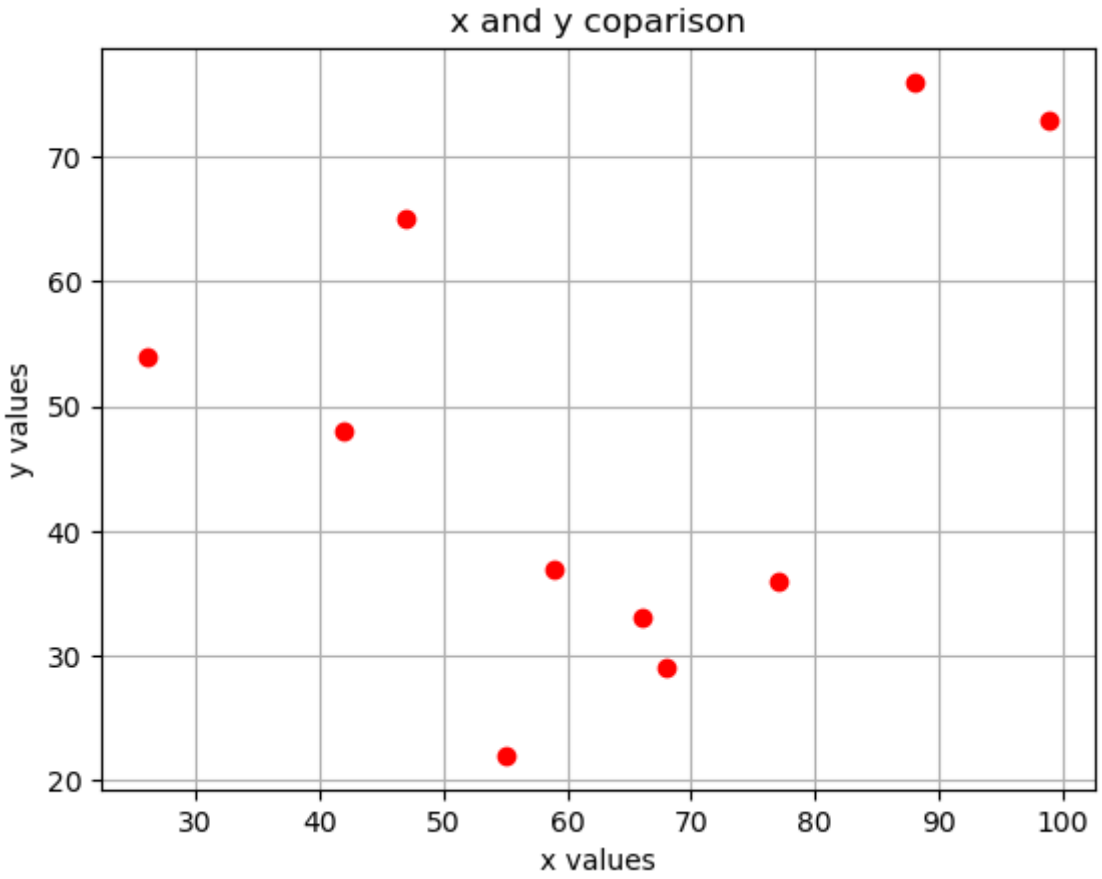
```
plt.plot(x,y,"b--", color="k")   #color g= green, b=blue, k=black, r= red, y=yellow
plt.xlabel("x values")
plt.ylabel("y values")
plt.title("x and y coparison")
plt.show()
```

```
C:\Users\johnb\AppData\Local\Temp\ipykernel_2504\3585746382.py:1: UserWarn
ing: color is redundantly defined by the 'color' keyword argument and the
fmt string "b--" (-> color='b'). The keyword argument will take precedenc
e.
  plt.plot(x,y,"b--", color="k")   #color g= green, b=blue, k=black, r= re
d, y=yellow
```



In [48]:

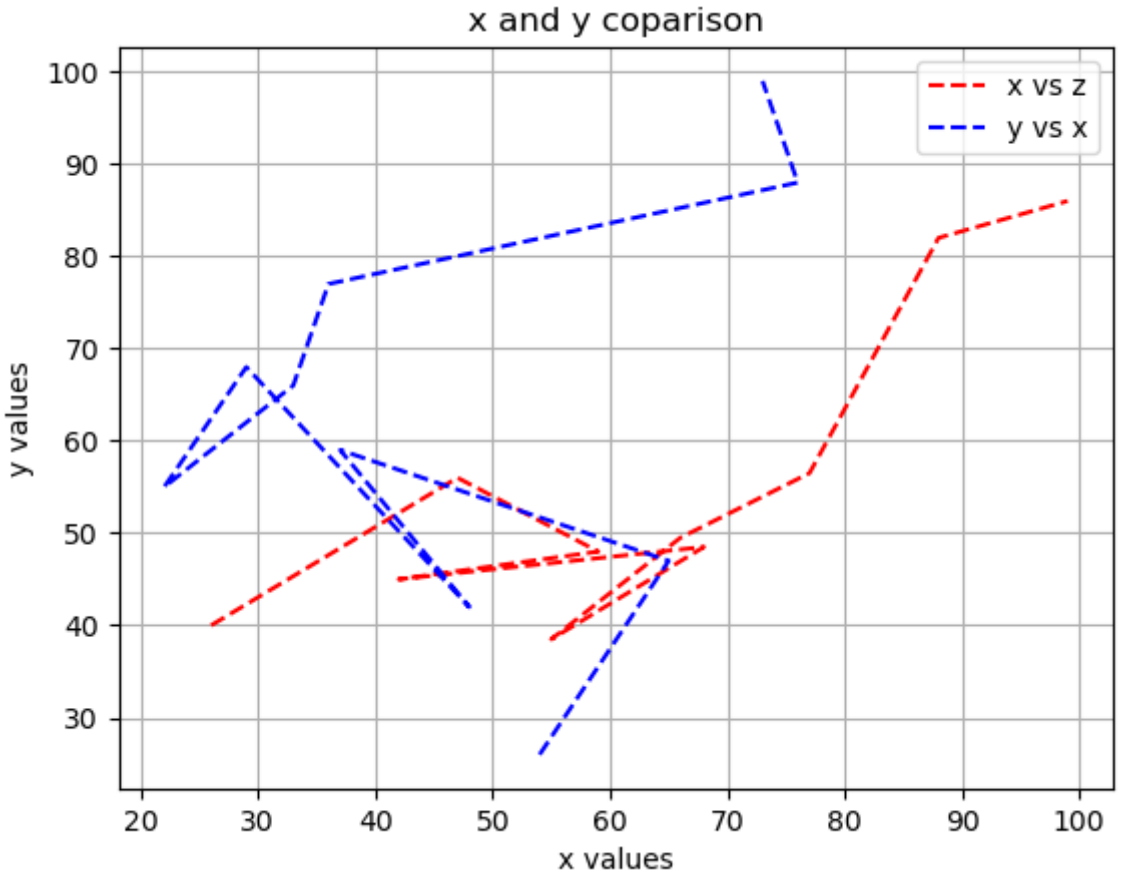
```
plt.scatter(x,y, color="r")   #color g= green, b=blue, k=black, r= red, y=yellow
plt.xlabel("x values")
plt.ylabel("y values")
plt.title("x and y coparison")
plt.grid()
#plt.savefig("D:\fig1.jpg")
plt.show()
```



In [49]:

```
plt.plot(x,z,"b--", label="x vs z", color="r")
plt.plot(y,x,"b--", label="y vs x", color="b")
plt.xlabel("x values")
plt.ylabel("y values")
plt.title("x and y coparison")
plt.legend()
plt.grid()
plt.show()
```

```
C:\Users\johnb\AppData\Local\Temp\ipykernel_2504\382996462.py:1: UserWarni
ng: color is redundantly defined by the 'color' keyword argument and the f
mt string "b--" (-> color='b'). The keyword argument will take precedence.
  plt.plot(x,z,"b--", label="x vs z", color="r")
C:\Users\johnb\AppData\Local\Temp\ipykernel_2504\382996462.py:2: UserWarni
ng: color is redundantly defined by the 'color' keyword argument and the f
mt string "b--" (-> color='b'). The keyword argument will take precedence.
  plt.plot(y,x,"b--", label="y vs x", color="b")
```



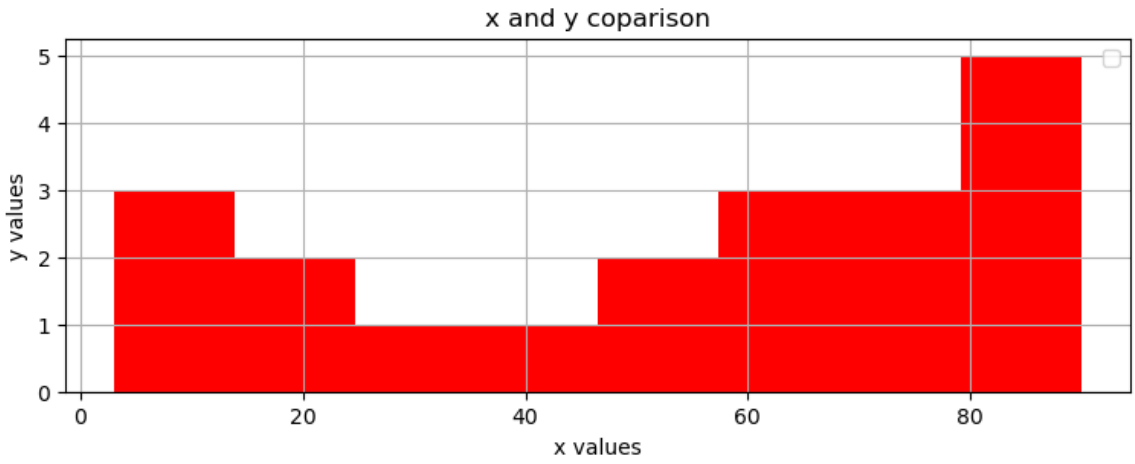
In [50]:

```
 # visualize the frequency of data using histogram

temp = np.array([20,47,63,48,69,72,85,9,3,6,21,43,68,88,89,71,82,90,31,62])
```

In [51]:

```
plt.figure(figsize=(9,3))
plt.hist(temp,bins=8, color="r")
plt.xlabel("x values")
plt.ylabel("y values")
plt.title("x and y coparison")
plt.legend()
plt.grid()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose la
bel start with an underscore are ignored when legend() is called with no a
rgument.
```



In [52]:

```
city = ["noth","south","east","west"]
values = ["2000","2500","1800","3500"]
ex =[0.2,0,0.2,0]
```

In [53]:

```
plt.pie(values,labels=city,explode=ex)
plt.show()
```

In [54]:

```
table_Iris = pd.read_csv("D:\DataScience\DataElement\Iris.csv")
```

In [55]:

```
table_Iris.head(5)
```

Out[55]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [56]:

```
table_Iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [57]:

```
table_Iris.describe()
```

Out[57]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [58]:

```
table_Iris= table_Iris.drop("Id", axis = 1)
```

In [59]:

```
table_Iris
```

Out[59]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

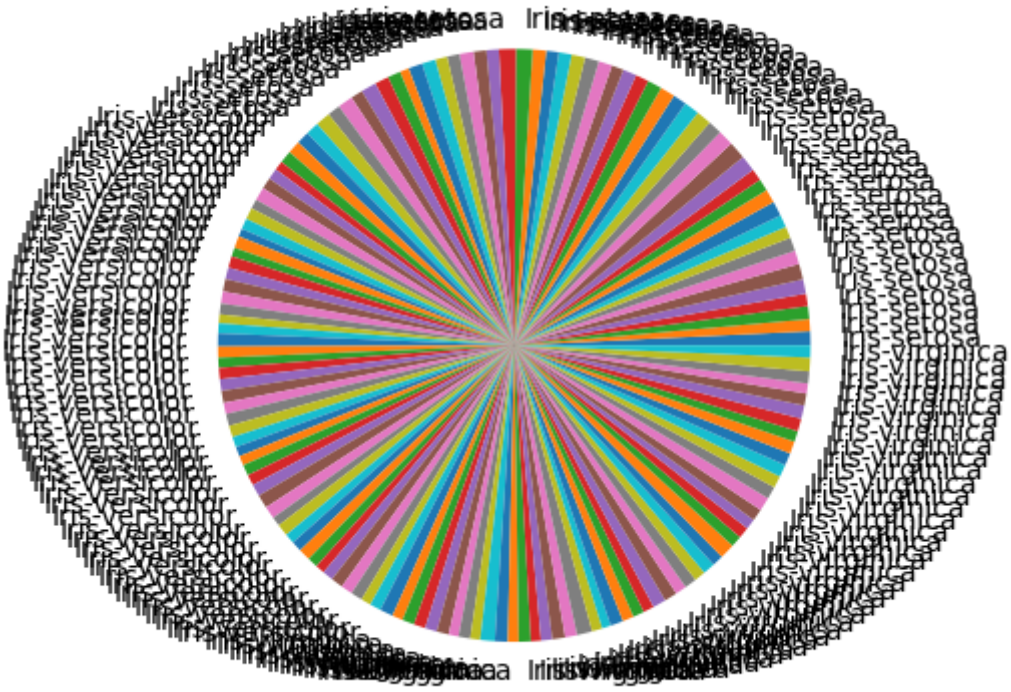
150 rows × 5 columns

In [60]:

```
plt.pie(table_Iris["SepalWidthCm"],labels= table_Iris["Species"] )
plt.show
```

Out[60]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [1]:

```
import numpy as np
arr1=np.random.randn(3,3)
arr1
```

Out[1]:

```
array([[ 0.71317909, -2.27208183,  0.56668485],
       [-0.14207934,  0.93167699,  0.71637899],
       [ 0.96947497,  0.11387776, -0.62662035]])
```

In [2]:

```
ar1=np.array([2,4,6,8])
```

In [3]:

```
ar1
```

Out[3]:

```
array([2, 4, 6, 8])
```

In [5]:

```
ar1.ndim
```

Out[5]:

```
1
```

In [6]:

```
ar2=np.array([[1,2],[3,4]])
```

In [7]:

```
ar2
```

Out[7]:

```
array([[1, 2],
       [3, 4]])
```

In [8]:

```
ar2.ndim
```

Out[8]:

```
2
```

In [9]:

```
a3=np.array([[1,2,3],[4,5,6]])
```

In [10]:

```
a3
```

Out[10]:

```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [11]:

```
a3.ndim
```

Out[11]:

```
2
```

In [13]:

```
b=np.reshape(a3,(3,2))
```

In [14]:

```
b
```

Out[14]:

```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

In [15]:

```
a1=np.array([20,40,60,80])
```

In [16]:

```
a1
```

Out[16]:

```
array([20, 40, 60, 80])
```

In [19]:

```
a2=np.array([30,70,90,20])
```

In [20]:

```
a2
```

Out[20]:

```
array([30, 70, 90, 20])
```

In [21]:

```
np.concatenate([a1,a2])
```

Out[21]:

```
array([20, 40, 60, 80, 30, 70, 90, 20])
```

In [23]:

```
f3=np.array([[1,2,3],[4,5,6]])
k3=np.array([[1,2,3],[4,5,6]])
```

In [28]:

```
np.concatenate([f3,k3])
```

Out[28]:

```
array([[1, 2, 3],
       [4, 5, 6],
       [1, 2, 3],
       [4, 5, 6]])
```

In [29]:

```
k3.ndim
```

Out[29]:

```
2
```

In [30]:

```
np.concatenate([f3,k3], axis=0)
```

Out[30]:

```
array([[1, 2, 3],
       [4, 5, 6],
       [1, 2, 3],
       [4, 5, 6]])
```

In [34]:

```
w=np.array([100,670,6990,870,66,7,8,56])
w
```

Out[34]:

```
array([ 100,  670, 6990,  870,   66,    7,    8,   56])
```

In [35]:

```
w[6] #6 is a index position
```

Out[35]:

```
8
```

In [36]:

```
#defining the range of index
w[4:8]
```

Out[36]:

```
array([66,  7,  8, 56])
```

In [38]:

```
w[-6:-2]
```

Out[38]:

```
array([6990,  870,   66,    7])
```

In [46]:

```
q=w[3:7]
```

In [47]:

```
q[:]= # called the slice operator
```

In [48]:

```
q
```

Out[48]:

```
array([10, 10, 10, 10])
```

In [49]:

```
w
```

Out[49]:

```
array([ 100,  670, 6990,   10,   10,   10,   10,   56])
```

In [50]:

```
ar1= w[3:7].copy()
```

In [51]:

```
ar1[:]=15
```

In [52]:

```
ar1
```

Out[52]:

```
array([15, 15, 15, 15])
```

In [53]:

```
w
```

Out[53]:

```
array([ 100,  670, 6990,   10,   10,   10,   10,   56])
```

In [54]:

```
z=np.random.randn(3,3)
```

In [56]:

```
np.exp(z)
```

Out[56]:

```
array([[1.03955128, 0.62397377, 6.91052711],
       [1.25976345, 1.26077313, 1.82650993],
       [1.46976685, 5.97689335, 3.46515418]])
```

In [57]:

```
np.log(z)
```

```
C:\Users\johnb\AppData\Local\Temp\ipykernel_6992\2047398432.py:1: RuntimeW
arning: invalid value encountered in log
  np.log(z)
```

Out[57]:

```
array([[-3.24961443,         nan,  0.65909695],
       [-1.46566678, -1.4622034 , -0.50682197],
       [-0.95424242,  0.58104226,  0.2173324 ]])
```

In [58]:

```
np.rint(z)
```

Out[58]:

```
array([[ 0., -0.,  2.],
       [ 0.,  0.,  1.],
       [ 0.,  2.,  1.]])
```

In [59]:

```
#statistical methods
np.mean(z)
```

Out[59]:

```
0.6645562296437402
```

In [60]:

```
np.median(z)
```

Out[60]:

```
0.3851037798682038
```

In [61]:

```
np.std(z)
```

Out[61]:

```
0.7706536656604007
```

In [62]:

```
np.var(z)
```

Out[62]:

```
0.5939070723958128
```

In [63]:

```
np.max(z)
```

Out[63]:

```
1.933045917710525
```

In [64]:

```
np.min(z)
```

Out[64]:

```
-0.4716469442502318
```

In [ ]:

```

```

In [26]:

```
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [29]:

```
table_iris = pd.read_csv("D:\DataScience\DataElement\Iris.csv")
```

In [30]:

```
table_iris
```

Out[30]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [31]:

```
table_iris.head()
```

Out[31]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [32]:

```
table_iris = table_iris.drop("Id", axis=1)
```

In [33]:

```
table_iris.head()
```

Out[33]:

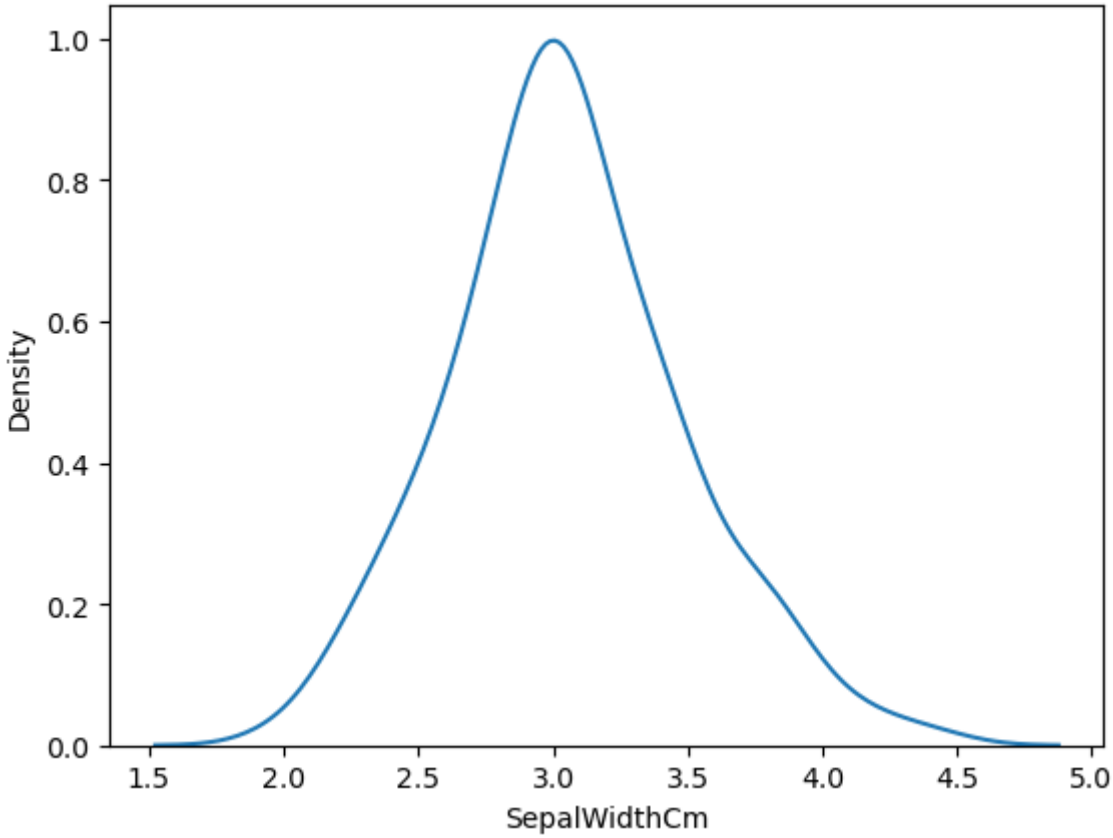
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [34]:

```
sns.kdeplot(table_iris["SepalWidthCm"])   #
```

Out[34]:

```
<Axes: xlabel='SepalWidthCm', ylabel='Density'>
```



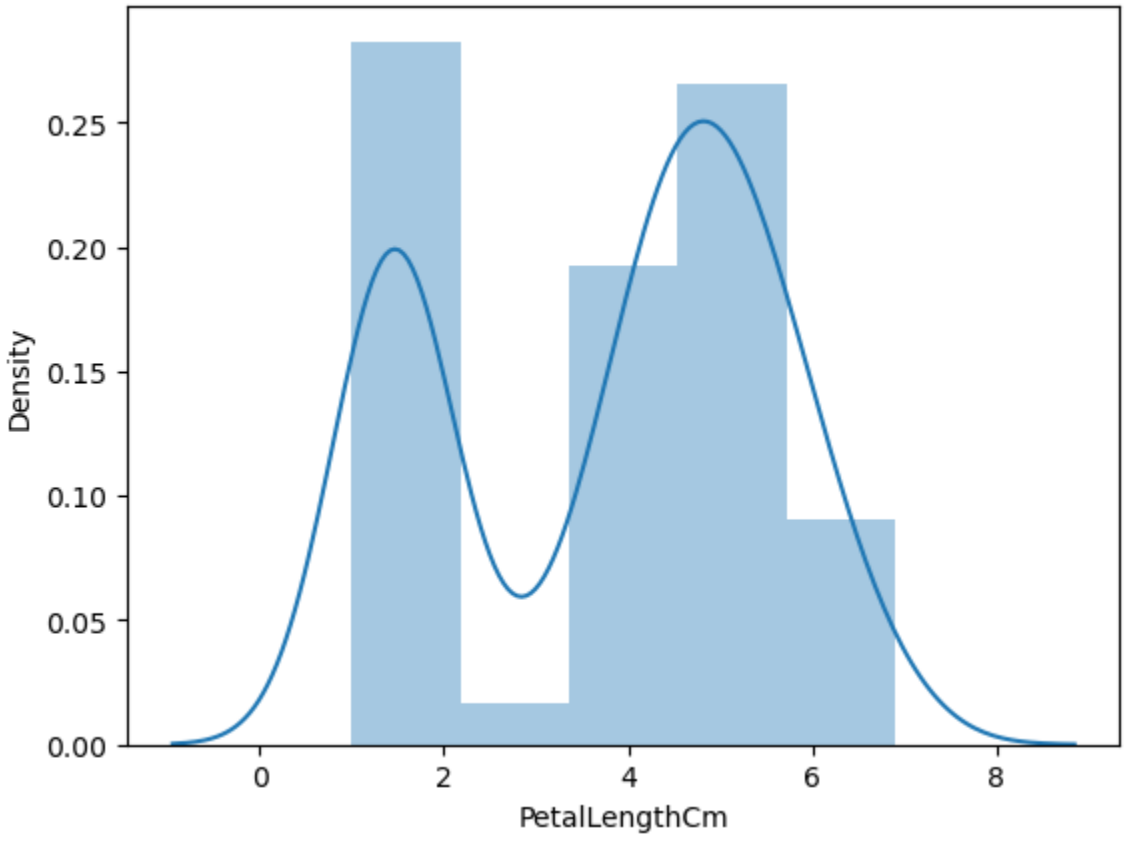
In [35]:

```
sns.distplot(table_iris["PetalLengthCm"],kde=True)
```

```
C:\Users\johnb\AppData\Local\Temp\ipykernel_3680\2845302684.py:1: UserWarn
ing:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://
gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(table_iris["PetalLengthCm"],kde=True)
```

Out[35]:

```
<Axes: xlabel='PetalLengthCm', ylabel='Density'>
```

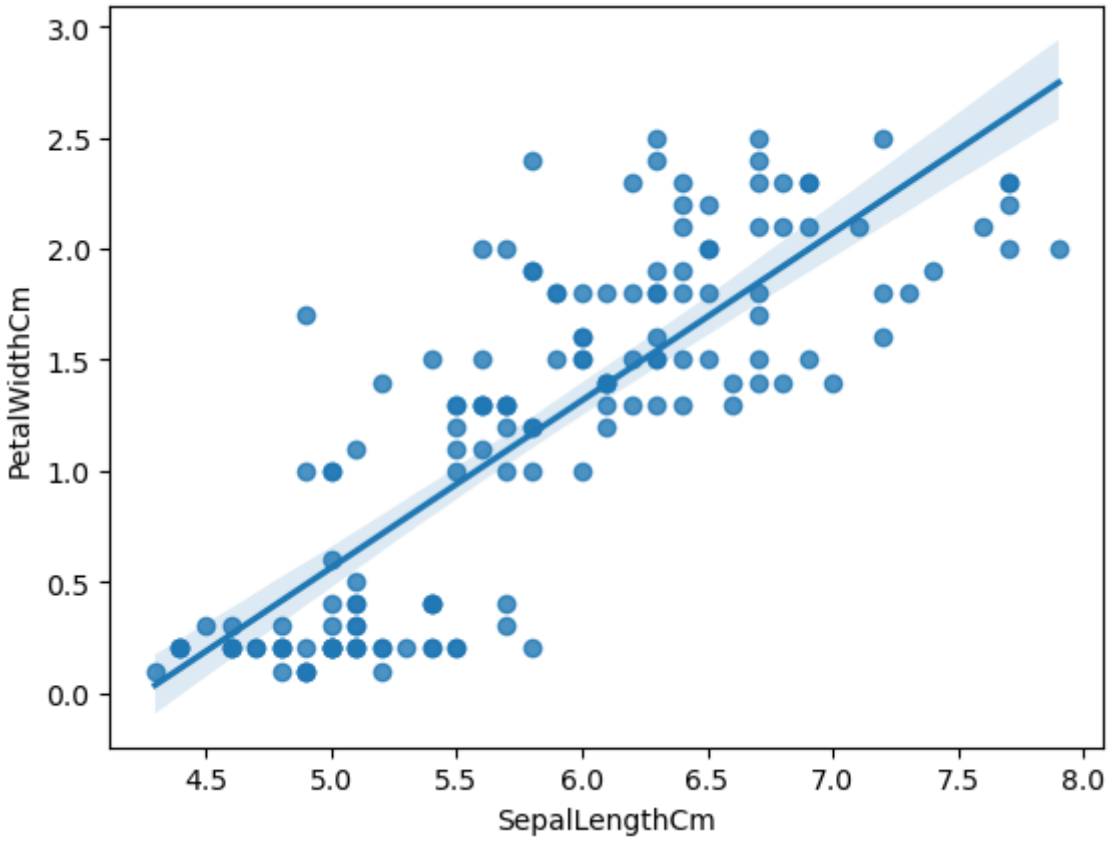


In [36]:

```
# regression plot is used to visualize the effect of one variable to another such that th
# y axis shows the dependent variable and x axis shows the independent variable
sns.regplot(x=table_iris["SepalLengthCm"], y = table_iris["PetalWidthCm"],fit_reg=True)
```

Out[36]:

```
<Axes: xlabel='SepalLengthCm', ylabel='PetalWidthCm'>
```

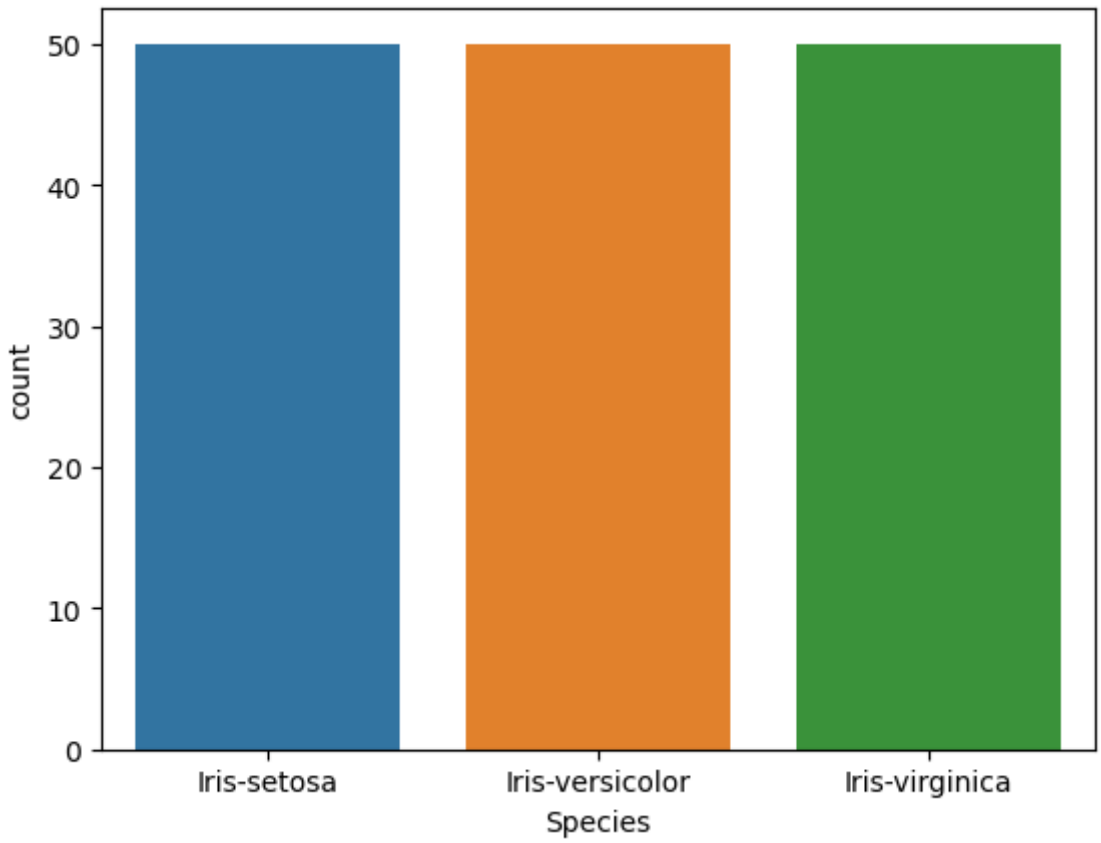


In [37]:

```
sns.countplot(x=table_iris["Species"])
```

Out[37]:

```
<Axes: xlabel='Species', ylabel='count'>
```



In [42]:

```
df1 = pd.read_csv("D:\DataScience\DataElement\Insurance.csv")
```

In [43]:

```
df1.head()
```

Out[43]:

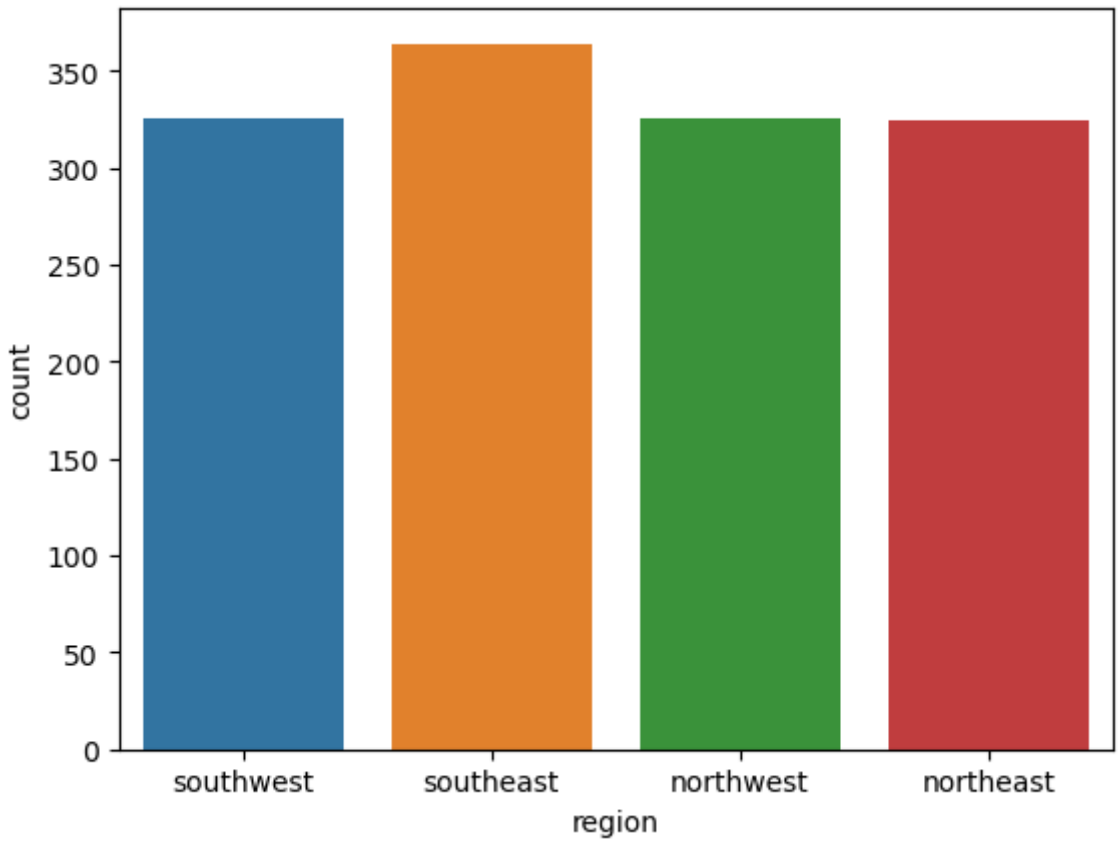
	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [46]:

```
sns.countplot(x=df1["region"])
```

Out[46]:

```
<Axes: xlabel='region', ylabel='count'>
```

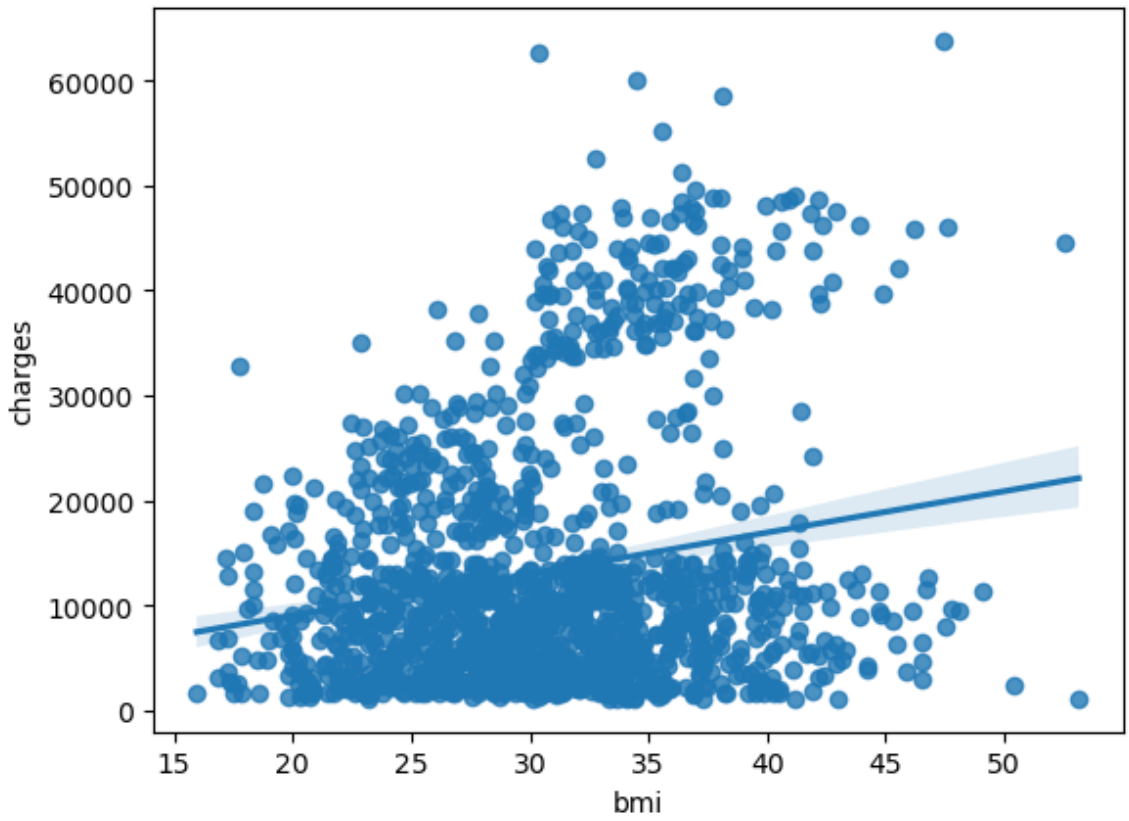


In [49]:

```
sns.regplot(x=df1["bmi"], y = df1["charges"],fit_reg=True)
```

Out[49]:

```
<Axes: xlabel='bmi', ylabel='charges'>
```

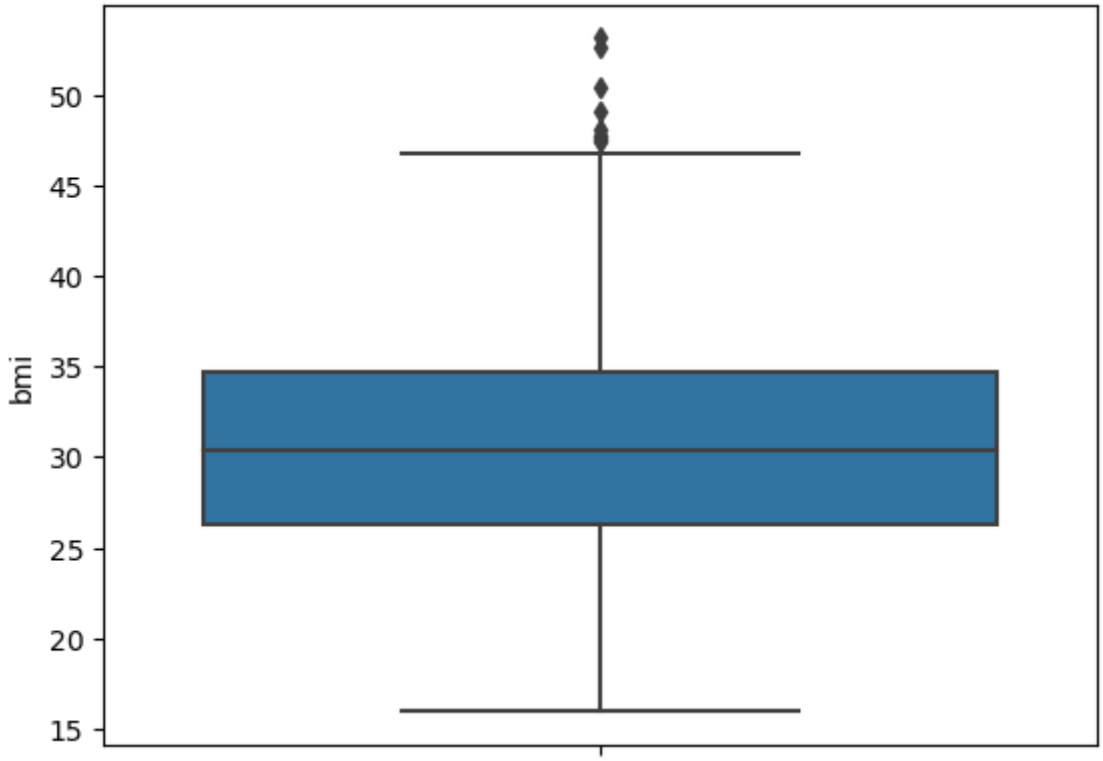


In [50]:

```
# to show the outlier present in the data we create the box
sns.boxplot(y = df1["bmi"])
```

Out[50]:

<Axes: ylabel='bmi'>

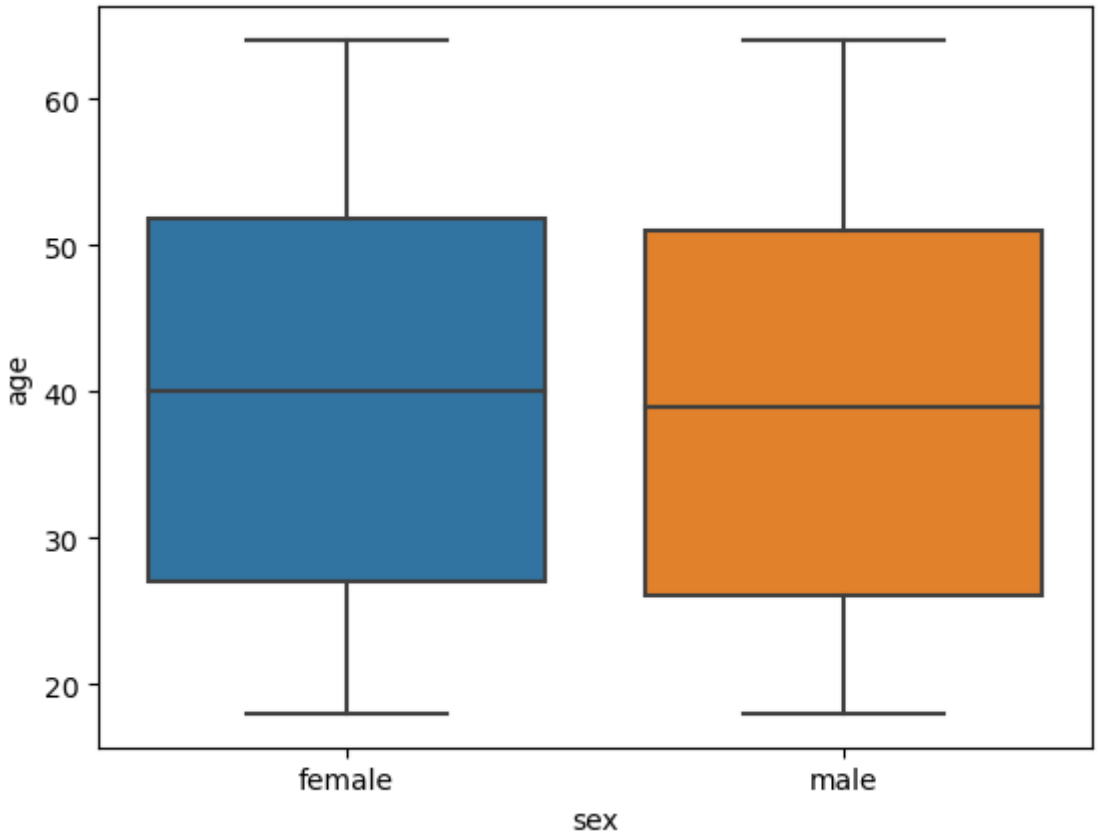


In [51]:

```
# coparing the categorial to numerical variable
sns.boxplot(x="sex", y="age", data=df1)
```

Out[51]:

```
<Axes: xlabel='sex', ylabel='age'>
```



In [ ]:

```
#heeatmap: core relation - is the selection between every variable exist in dataset .
```

In [ ]:

```
cor=df1.corr()
```

In [54]:

```
cor
```

Out[54]:

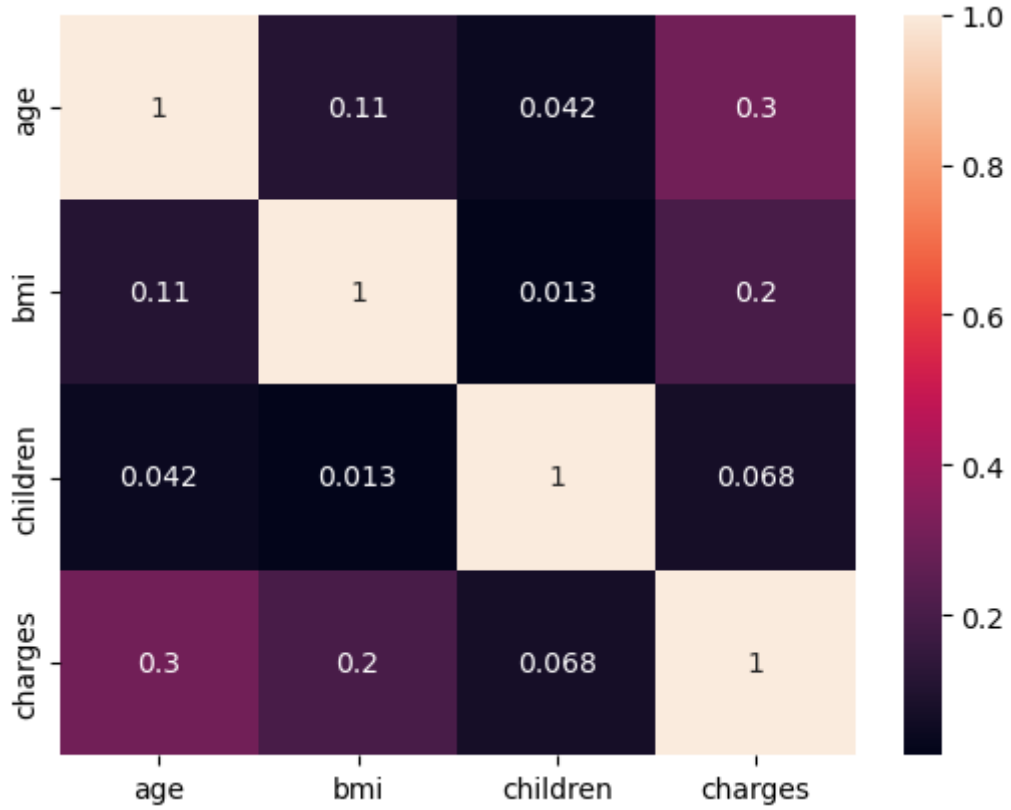
	age	bmi	children	charges
age	1.000000	0.109272	0.042469	0.299008
bmi	0.109272	1.000000	0.012759	0.198341
children	0.042469	0.012759	1.000000	0.067998
charges	0.299008	0.198341	0.067998	1.000000

In [55]:

```
sns.heatmap(cor,annot=True)
```

Out[55]:

<Axes: >



In [ ]: