| 1 | write a full name in python. |
| --- | --- |
| CODE |  |
| OUTPUT |  |
| 2 | write a output for this  output-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  BCA6  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| CODE |  |
| OUTPUT |  |
| 3 | print below pattern using print function  \*  \*\*  \*\*\*  \*\*\*\* |
| CODE |  |
| OUTPUT |  |
| 4 | unrate a biodata which include  Enrollment no  Roll no  Name  Father name  Email  Mobile no  DOB  address  city  state  country  hobby |
| CODE |  |
| OUTPUT |  |
| 5 | print bellow pattern using print function  $$$$$$$$$$$$$  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  name  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  $$$$$$$$$$$$$$ |
| CODE |  |
| OUTPUT |  |
| 6 | update your biodata mo. no. without ". |
| CODE |  |
| OUTPUT |  |
| 7 | add 2 float values. |
| CODE |  |
| OUTPUT |  |
| 8 | generate a mini calculator. |
| CODE |  |
| OUTPUT |  |
| 9 | use the concept of comments. |
| CODE |  |
| OUTPUT |  |
| 10 | combine integer and float with a mini calculator. |
| CODE | def add(num1, num2):  return num1 + num2  def subtract(num1, num2):  return num1 - num2  def multiply(num1, num2):  return num1 \* num2  def divide(num1, num2):  return num1 / num2  num1 = float(input("Enter first number: "))  num2 = float(input("Enter second number: "))  print("\nSelect operation:")  print("1. Add")  print("2. Subtract")  print("3. Multiply")  print("4. Divide")  choice = input("Enter choice (1/2/3/4): ")  if choice == '1':  print(num1, "+", num2, "=", add(num1, num2))  elif choice == '2':  print(num1, "-", num2, "=", subtract(num1, num2))  elif choice == '3':  print(num1, "\*", num2, "=", multiply(num1, num2))  elif choice == '4':  print(num1, "/", num2, "=", divide(num1, num2))  else:  print("Invalid input") |
| OUTPUT |  |
| 11 | 5. Emojis  5. flowers  5. humanbin |
| CODE |  |
| OUTPUT |  |
| 12 | Data type |
| CODE | a=123  print(a)  b=12.34  print(b)  c="abcd"  print(c)  d="bca6"  print(d)  r=True  print(r)  e="null"  print(e)  print(type(a))  print(type(b))  print(type(c))  print(type(d))  print(type(r))  print(type(e)) |
| OUTPUT |  |
| 13 | create a script by taking 2 variables x1 and x2 perform addition, subtraction multiplication and division by checking the data type. |
| CODE | x1=14  x2=10  a=x1+x2  b=x1-x2  c=x1\*x2  d=x1/x2  print(a)  print(b)  print(c)  print(d) |
| OUTPUT |  |
| 14 | create a script by taking a variable dozen. you need to convert into pieces. |
| CODE | dozen = 12  pieces = dozen \* 2  print(pieces) |
| OUTPUT |  |
| 15 | write a script to convert cm to inches and inches to cm in one single line. |
| CODE | cm = 10  inches = cm / 2.54  cm\_back = inches \* 2.54  print(cm, "=", inches, "inches and", inches, "=", cm\_back, "cm") |
| OUTPUT |  |
| 16 | use qus. 14 convert pieces to a dozen. |
| CODE | pieces = 24  dozen = pieces / 12  print(pieces, "=", dozen, "dozen") |
| OUTPUT |  |
| 17 | Print all classes in python. |
| CODE |  |
| OUTPUT |  |
| 18(N) | Data type and print ans |
| CODE | n1=n2=n3=10  print(type(n1))  print(n1)  n1,n2,n3=100,200,300  print(n1,n2,n3) |
| OUTPUT |  |
| 19(N) | Arithmetic operations |
| CODE | x=14  y=10  ans=x+y  print("addition is :",ans)  ans=x-y  print("subtraction is :",ans)  ans=x/y  print("division is:% is :",ans)  ans=x\*y  print("multiplication is :",ans)  ans=x%y  print("reminder is :",ans)  ans=x\*\*y  print("x raised to y is :",ans)  ans=x//y  print("floor division :",ans) |
| OUTPUT |  |
| 20(N) | Enter the value of x and y. |
| CODE | print("enter value of x :")  x=int(input())  print("enter value of y :")  y=int(input())  ans=x+y  print("addition of x and y is :",ans) |
| OUTPUT |  |
| 21 | int ,float,str |
| CODE | w=int(input("enter no1:"))  print(w)  y=int(input("enter no2:"))  print(y)  z=(w+y)  print("total",z)  print(type (w))  print(type(y))  print(type(z))  …………………………………………….  w=(input("enter no1:"))  print(w)  y=int(input("enter no2:"))  print(y)  z=(w+y)  print("total",z)  print(type (w))  print(type(y))  print(type(z))  ………………………………………….  w=float(input("enter no1:"))  print(w)  y=int(input("enter no2:"))  print(y)  z=(w+y)  print("total",z)  print(type (w))  print(type(y))  print(type(z)) |
| OUTPUT |  |
| 22 | Write a python script to demonstrate simple interest . |
| CODE | def simple\_interest(p,r,t):  si = (p \* r \* t)/100  return si    p=int(input("Enter principal amount is :"))  r=int(input("Enter rate of interest is :"))  t=int(input("Enter time period :"))  print('The Simple Interest is', simple\_interest(p, r, t)) |
| OUTPUT |  |
| 23 | Write a python script to demonstrate compound interest. |
| CODE | def compound\_rate(PV, CRate, tp):  TotalAmount = PV \* (pow ((1 + CRate / 100), tp))  CInterest = TotalAmount - PV  print("Total return value after completion of given time period: ", TotalAmount)  print("Compound interest gained on given amount is", CInterest)  PV = float(input("Enter the principal amount: "))  CRate = float(input("Enter the rate for compound interest: ")) # taking interest rate value  tp = float(input("Enter the time period for which principal is invested: ")) # taking time period value      compound\_rate(PV, CRate, tp) |
| OUTPUT |  |
| 24 | Print name |
| CODE | name=input("enter 5 name :")  print(name) |
| OUTPUT |  |
| 25 | write a python code to enter address  city  in your address a number was printed(in index)  in address a number r |
| CODE | name=(input("enter a name of city :"))  print(name)  print(type(name))  print(name[0])  ……………………………………………………………  name=(input("enter a name of city :"))  print(name)  print(type(name))  print(name[1]) |
| OUTPUT |  |
| 26 | tuple |
| CODE | x=("abc","def","ghi")  print(x)  print(type(x))  print(x[2]) |
| OUTPUT |  |
| 27 | list |
| CODE | x=["abc","def","ghi"]  print(x)  print(type(x))  print(x[2]) |
| OUTPUT |  |
| 28 | write a script using user input fetch 5 names of city and print it(use concept of list)  i-display 2 and 3 city  ii-display index 0  iii-display prevus last and last index  iv-display index 2,3,4,5  v-display index 3 |
| CODE | city=("rajkot","surat","junagadh","gandhinagar","morbi")  print(city)  print(type(city))  print(city[2],city[3])  print(city[0])  print(city[3:5])  print(city[1:5])  print(city[3])  print(city[-4:-2]) |
| OUTPUT |  |
| 29 | take 7 fruit names  print all the data forward and backward  print middle value  Print first and last index  Print backward last 3 data  Print all the data starting from the 3 position |
| CODE | fruits = ["apple", "banana", "orange", "kiwi", "mango", "grape", "pineapple"]  print(fruits)  print(fruits[::-1])  print(fruits[len(fruits)//2])  print("First index:", fruits[0])  print("Last index:", fruits[-1])  print(fruits[-3:][::-1])  print(fruits[2:]) |
| OUTPUT |  |
| 30 | 4-qus 3 convert in to tuple |
| CODE | fruits = ("apple", "banana", "orange", "kiwi", "mango", "grape", "pineapple")  print(fruits)  print(fruits[::-1])  print(fruits[len(fruits)//2])  print("First index:", fruits[0])  print("Last index:", fruits[-1])  print(fruits[-3:][::-1])  print(fruits[2:]) |
| OUTPUT |  |
| 31(N) | write a program to reverse number |
| CODE | for i in range(10, 0, -1):  print(i) |
| OUTPUT |  |
| 32(N) | Even number |
| CODE | for i in range(2, 21, 2):  print(i) |
| OUTPUT |  |
| 33(N) | FACTORIAL |
| CODE | print("Enter Number:")  no = int(input())  i = 1  fact = 1  while i <= no:  fact = fact \* i  i = i + 1  print(f"Factorial of {no} is: {fact}") |
| OUTPUT |  |
| 34 | **GUI code in python** |
|  |  |
|  |  |
|  |  |
|  |  |
| 35 | **Label , textbox, button using GUI** |
| CODE |  |
| OUTPUT |  |
| 36 | create a frame with your name  background color of the fream navy blue  design frame size 550,550  insert label left middle bank information  4 text box in the center bank id , bank name , ifsc code , city  Bottom submit button |
| CODE | import tkinter as tk  root = tk.Tk()  root.title("My Frame")  root.geometry("550x550")  root.configure(bg="navy")  label = tk.Label(root, text="Bank Information", font=("Arial", 20), fg="white", bg="navy")  label.place(relx=0.5, rely=0.1, anchor="center")  bank\_id\_label = tk.Label(root, text="Bank ID:", font=("Arial", 12), fg="white", bg="navy")  bank\_id\_label.place(relx=0.3, rely=0.4, anchor="center")  bank\_id\_textbox = tk.Entry(root, font=("Arial", 12))  bank\_id\_textbox.place(relx=0.6, rely=0.4, anchor="center")  bank\_name\_label = tk.Label(root, text="Bank Name:", font=("Arial", 12), fg="white", bg="navy")  bank\_name\_label.place(relx=0.3, rely=0.5, anchor="center")  bank\_name\_textbox = tk.Entry(root, font=("Arial", 12))  bank\_name\_textbox.place(relx=0.6, rely=0.5, anchor="center")  ifsc\_code\_label = tk.Label(root, text="IFSC Code:", font=("Arial", 12), fg="white", bg="navy")  ifsc\_code\_label.place(relx=0.3, rely=0.6, anchor="center")  ifsc\_code\_textbox = tk.Entry(root, font=("Arial", 12))  ifsc\_code\_textbox.place(relx=0.6, rely=0.6, anchor="center")  city\_label = tk.Label(root, text="City:", font=("Arial", 12), fg="white", bg="navy")  city\_label.place(relx=0.3, rely=0.7, anchor="center")  city\_textbox = tk.Entry(root, font=("Arial", 12))  city\_textbox.place(relx=0.6, rely=0.7, anchor="center")  submit\_button = tk.Button(root, text="Submit", font=("Arial", 12), bg="white", fg="navy")  submit\_button.place(relx=0.5, rely=0.9, anchor="center")  root.mainloop() |
| OUTPUT |  |
| 37 | design 350,350 fream  2 textbox for number  4 button for +,-,\*,/ |
| CODE | import tkinter as tk  root = tk.Tk()  root.title("Calculator")  root.geometry("350x350")  num1\_label = tk.Label(root, text="Enter first number:", font=("Arial", 12))  num1\_label.place(relx=0.2, rely=0.2, anchor="center")  num1\_textbox = tk.Entry(root, font=("Arial", 12))  num1\_textbox.place(relx=0.7, rely=0.2, anchor="center")  num2\_label = tk.Label(root, text="Enter second number:", font=("Arial", 12))  num2\_label.place(relx=0.2, rely=0.4, anchor="center")  num2\_textbox = tk.Entry(root, font=("Arial", 12))  num2\_textbox.place(relx=0.7, rely=0.4, anchor="center")  def add\_numbers():  num1 = float(num1\_textbox.get())  num2 = float(num2\_textbox.get())  result\_label.config(text="Result: " + str(num1 + num2))  def subtract\_numbers():  num1 = float(num1\_textbox.get())  num2 = float(num2\_textbox.get())  result\_label.config(text="Result: " + str(num1 - num2))  def multiply\_numbers():  num1 = float(num1\_textbox.get())  num2 = float(num2\_textbox.get())  result\_label.config(text="Result: " + str(num1 \* num2))  def divide\_numbers():  num1 = float(num1\_textbox.get())  num2 = float(num2\_textbox.get())  if num2 == 0:  result\_label.config(text="Result: Cannot divide by zero")  else:  result\_label.config(text="Result: " + str(num1 / num2))  add\_button = tk.Button(root, text="+", font=("Arial", 12), command=add\_numbers)  add\_button.place(relx=0.3, rely=0.6, anchor="center")  subtract\_button = tk.Button(root, text="-", font=("Arial", 12), command=subtract\_numbers)  subtract\_button.place(relx=0.5, rely=0.6, anchor="center")  multiply\_button = tk.Button(root, text="\*", font=("Arial", 12), command=multiply\_numbers)  multiply\_button.place(relx=0.7, rely=0.6, anchor="center")  divide\_button = tk.Button(root, text="/", font=("Arial", 12), command=divide\_numbers)  divide\_button.place(relx=0.9, rely=0.6, anchor="center")  result\_label = tk.Label(root, text="Result: ", font=("Arial", 12))  result\_label.place(relx=0.5, rely=0.8, anchor="center")  root.mainloop() |
| OUTPUT |  |
| 38 | Data display in list (Ctrl+space ) |
| CODE | w=[67,78,56,54,71]  print(type(w))  w.pop(2)  w.remove(78)  w.reverse()  w.insert(3,28)  w.sort()  print(w) |
| OUTPUT |  |
| 39 | height = 23.45,72.14,11.12,7.56,23.14  1-insert 3rd position value 14  2-reverse the list  3-insert negative value starting of the list  4-sort your data  5-insert duplicate 3 values with the series of 30  6-remove negative value of the list  7-reverse your data  8-count the values in the list  9-delete the data which is starting from 3  10-display ascending data |
| CODE | # Given list  heights = [23.5, 72.14, 11.12, 7.56, 23.14]  print("")  print("")  # i. Insert 3rd position value 14.  heights.insert(2, 14)  print("i. Insert 3rd position value 14:", heights)  print("")  # ii. Reverse the list  heights.reverse()  print("ii. Reverse the list:", heights)  print("")  # iii. Insert negative value starting of the list.  heights.insert(0, -1)  heights.insert(0, -2)  heights.insert(0, -3)  print("iii. Insert negative value starting of the list:", heights)  print("")  # iv. Sort your data.  heights.sort()  print("iv. Sort your data:", heights)  print("")  # v. Insert duplicate 3 values with the series of 30  for i in range(3):  heights.extend([30] \* 3)  print("v. Insert duplicate 3 values with the series of 30:", heights)  print("")  # vi. Remove negative values from the list.  heights = [x for x in heights if x >= 0]  print("vi. Remove negative values from the list:", heights)  print("")  # vii. Reverse the data  heights.reverse()  print("vii. Reverse the data:", heights)  print("")  # viii. Count the value  count\_value = heights.count(30)  print("viii. Count the value 30:", count\_value)  print("")  # ix. Delete the data which is starting from 3.  del heights[:3]  print("ix. Delete the data which is starting from 3:", heights)  print("")  # x. Display ascending data.  heights.sort()  print("x. Display ascending data:", heights) |
| OUTPUT |  |
| 40 | list,tuple,set |
| CODE | x=[67,{78,89},(90)]  print(type(x))  y=(56,{78,56},[34,78],56)  print(type(y))  z={(56,56),(1,2),78,34,56,9}  print(type(z))  z.add(1)  print(z) |
| OUTPUT |  |
| 41 | fruits ={apple,grapes,mango,dragon,orchi,kiwi,chiku,lemon}  1-arrange the data in descending order  2-delete middle aliment  3-replace the fruit lemon with another name  4-count the data  5-use the join function with star  6-guntrate another set with category inside fruits it is if={apple,grapes,guava,cheeku}  7-match both the data  8-use different operators for matching |
| CODE | # Given set of fruits  fruits = {'Apple', 'Grapes', 'Mango', 'Dragon', 'Orchi', 'Kiwi', 'Cheeku', 'Lemon'}  # 1. Arrange data in descending order  sorted\_fruits = sorted(fruits, reverse=True)  print("1. Descending Order:", sorted\_fruits)  # 2. Delete middle element  fruits\_list = list(fruits)  middle\_index = len(fruits\_list) // 2  del fruits\_list[middle\_index]  updated\_fruits = set(fruits\_list)  print("2. After Deleting Middle Element:", updated\_fruits)  # 3. Replace the fruit 'Lemon' with another name  updated\_fruits.remove('Lemon')  updated\_fruits.add('Orange') # Replace 'Lemon' with 'Orange'  print("3. After Replacing Lemon:", updated\_fruits)  # 4. Count the data  count\_of\_fruits = len(updated\_fruits)  print("4. Count of Fruits:", count\_of\_fruits)  # 5. Use the join function with star  joined\_fruits = '\*'.join(updated\_fruits)  print("5. Joined Fruits:", joined\_fruits)  # 6. Generate another set with categories 'Indian fruits'  indian\_fruits = {'Apple', 'Grapes', 'Guava', 'Cheeku'}  print("6. Indian Fruits Set:", indian\_fruits)  # 7. Match both sets  matching\_fruits = updated\_fruits.intersection(indian\_fruits)  print("7. Matching Fruits:", matching\_fruits)  # 8. Use different operators for matching  is\_subset = indian\_fruits <= updated\_fruits  is\_superset = updated\_fruits >= indian\_fruits  is\_disjoint = updated\_fruits.isdisjoint(indian\_fruits)  print("8. Using Different Operators:")  print(" - Is Subset:", is\_subset)  print(" - Is Superset:", is\_superset)  print(" - Is Disjoint:", is\_disjoint) |
| OUTPUT |  |
| 42 | w=[45,35,25,1,2,3,45,35,25]  1-sort the data  2-remove the duplicate aliment  3-replace aliment 1  4-insert negative value  4-arrange to ascending  6-delete the last element |
| CODE | w=[45,35,25,1,2,3,45,35,25]  w\_sorted = sorted(w)  print(w\_sorted)  w\_unique = list(set(w))  print(w\_unique)  w\_replaced = [10 if x==1 else x for x in w\_unique]  print(w\_replaced)  w\_negative = [-x for x in w\_replaced]  print(w\_negative)  w\_ascending = sorted(w\_negative)  print(w\_ascending)  del w\_ascending[-1]  print(w\_ascending) |
| OUTPUT |  |
| 43(N) | #loop |
| CODE | i=1  while i<=10:  print(i)  i=i+1 |
| OUTPUT |  |
| 44(N) | #factorial |
| CODE | print("enter any number to find factorial :")  no=int(input())  fact=1  i=1  while i <= no:  fact=fact\*i  i=i+1  print("factorial of {} is:{}".format(no,fact)) |
| OUTPUT |  |
| 45(N) | #write a sum of digits of given number |
| CODE | print("Enter a number: ")  no= int(input())  sum = 0  while no > 0:  rem = no % 10  sum =sum + rem  no =no // 10  print("Sum of digits is:", sum) |
| OUTPUT |  |
| 46(N) | #print reverse number |
| CODE | num = int(input("Enter a number: "))  reverse\_num = 0  while num > 0:  digit = num % 10  reverse\_num = (reverse\_num \* 10) + digit  num //= 10  print("Reverse of the number:", reverse\_num) |
| OUTPUT |  |
| 47(N) | #to find Armstrong number |
| CODE | num = int(input("Enter a number: "))  sum = 0  temp = num  while temp > 0:  digit = temp % 10  sum += digit \*\* 3  temp //= 10  if num == sum:  print(num, "is an Armstrong number")  else:  print(num, "is not an Armstrong number") |
| OUTPUT |  |
| 48(N) | fibonacci series |
| CODE | print("enter term to generate fibonacci series")  no=int(input())  i=1  n1=0  n2=1  while i<=no:  print(n1) #0  n=n1+n2 #n = 1  n1=n2 #n1 = 1  n2=n #n2 = 1  i=i+1 |
| OUTPUT |  |
| 49(N) | prime number |
| CODE | print("Enter a number:")  no = int(input())  i = 2  flag = 0  while i <= no/2:  if no % i == 0:  flag = 1  pass  i += 1  if flag == 0:  print(no, "is a prime number")  else:  print(no, "is not a prime number") |
| OUTPUT |  |
| 50 | dictionary |
| CODE | #dictionary  x={1:{5,6,7},2:{8,6,7},'a':{"abc","def","xyz"}}  print(x)  print(type(x))  print(max(x[1]))  print(len(x[1]))  print(x[1])  print(x[2])  print(x['a'])  print("length:",len(x[1]),"min:",min(x[2])) |
| OUTPUT |  |
| 51 | create a dictionary which is hending  integer 5 values  float 5 values  string 5 values  character values |
| CODE | # Original dictionary  x = {1: {5, 6, 7}, 2: {8, 6, 7}, 'a': {"abc", "def", "xyz"}}  print("Original Dictionary:")  print(x)  print("\nType of x:", type(x))  print("\nMaximum value in set associated with key 1:", max(x[1]))  print("Length of set associated with key 1:", len(x[1]))  print("\nSet associated with key 1:", x[1])  print("Set associated with key 2:", x[2])  print("Set associated with key 'a':", x['a'])  x[3] = {10, 11, 12}  print("\nAfter Insertion (Key 3):")  print(x)  x[2].update({9, 10})  print("\nAfter Update (Key 2):")  print(x)  del x['a']  print("\nAfter Deletion (Key 'a'):")  print(x) |
| OUTPUT |  |
| 52 | create a dictionary  5 fruit,5 roll numbers,5 percentage.  1-after roll no insert name.  2-short percentage wise data.  3-rearrange data a to z.  4-find minimum and maximum data in percentage.  5-insert the last element in the fruit and delete the first element.  6-find out the minimum character in the fruit and display it.  7-update any 1 fruit .  8-remove the middle element.  9-add the 1 name and that mark is 0.  10-Display the whole dictionary with data type. |
| CODE | dictionary = {  "fruit": ["apple", "banana", "cherry", "date", "elderberry"],  "roll no": [1, 2, 3, 4, 5],  "percentage": [80, 90, 75, 85, 95]  }  # 1.  dictionary["name"] = ["vrundali", "vishva", "ishita", "happy", "manisha"]  # 2.  sorted\_percentage = sorted(dictionary["percentage"])  print("Sorted percentage:", sorted\_percentage)  # 3.  sorted\_fruit = sorted(dictionary["fruit"])  sorted\_roll\_no = sorted(dictionary["roll no"])  sorted\_name = sorted(dictionary["name"])  sorted\_percentage = sorted(dictionary["percentage"])  dictionary = {  "fruit": sorted\_fruit,  "roll no": sorted\_roll\_no,  "name": sorted\_name,  "percentage": sorted\_percentage  }  print("Sorted data:", dictionary)  # 4.  minimum\_percentage = min(dictionary["percentage"])  maximum\_percentage = max(dictionary["percentage"])  print("Minimum percentage:", minimum\_percentage)  print("Maximum percentage:", maximum\_percentage)  # 5.  dictionary["fruit"].append("fig")  del dictionary["fruit"][0]  print("Updated fruit:", dictionary["fruit"])  # 6.  minimum\_character = min(dictionary["fruit"], key=len)  print("Minimum character in fruit:", minimum\_character)  # 7.  dictionary["fruit"][2] = "grape"  print("Updated fruit:", dictionary["fruit"])  # 8.  middle\_index = len(dictionary["roll no"]) // 2  del dictionary["roll no"][middle\_index]  print("Updated roll no:", dictionary["roll no"])  # 9.  dictionary["name"].append("khushi")  dictionary["percentage"].append(0)  print("Updated data:", dictionary)  # 10.  for key, value in dictionary.items():  print(key, ":", type(value), ":", value) |
| OUTPUT |  |
| 53(N) | Print using for loop |
| CODE | test=[1,2,3,4,5]  for i in test:  print (i)  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  test=[1.1,2.2,3.3,4.4,5.5]  for i in test:  print (i,end = " ") |
| OUTPUT |  |
| 54(N) | loop |
| CODE | str="rku-Rajkot"  for i in str:  print (i,end='') |
| OUTPUT |  |
| 55 | Default function |
| CODE | def abc():  print("hello")  abc() |
| OUTPUT |  |
| 56 | Default function |
| CODE | def abc(a):  print("a")  abc(2) |
| OUTPUT |  |
| 57 | Default function |
| CODE | def abc(a):  print("Square:",a\*a)  print("enter no :")  x=int(input())  abc(x) |
| OUTPUT |  |
| 58 | Default function check for int. |
| CODE | def abc(\*x):  #int  print(x[1])  abc(1,2,3) |
| OUTPUT |  |
| 59 | Default function |
| CODE | def abc(\*x):  print(x)    abc("aa")  abc()  abc("a") |
| OUTPUT |  |
| 60 | Task Based Assignment(use function) |
| ANS | **Task Based Assignment(use function)**  A-cube of numbers.  def abc(a):  print("Cube:",a\*a\*a)  print("enter no :")  x=int(input())  abc(x)    B-check whether the number is integer or float.  def number(num):  if isinstance(num, int):  print("The number is an integer")  elif isinstance(num, float):  print("The number is a float")    number(10)  number(10.14)    C-check whether leap year or not.  def is\_leap\_year(year):  if year % 4 == 0:  if year % 100 == 0:  if year % 400 == 0:  return True  else:  return False  else:  return True  else:  return False  year = 2024  if is\_leap\_year(year):  print(f"{year} is a leap year")  else:  print(f"{year} is not a leap year")    D-parameter parsing performs simple interest.  def calculate\_simple\_interest(principal, rate, time):  """  Calculate simple interest.  Parameters:  - principal (float): The principal amount.  - rate (float): The interest rate per period.  - time (float): The time period in years.  Returns:  - float: The calculated simple interest.  """  interest = (principal \* rate \* time) / 100  return interest  principal\_amount = 1000  interest\_rate = 5  time\_period = 2  simple\_interest = calculate\_simple\_interest(principal\_amount, interest\_rate, time\_period)  print(f"Principal Amount: ${principal\_amount}")  print(f"Interest Rate: {interest\_rate}%")  print(f"Time Period: {time\_period} years")  print(f"Simple Interest: ${simple\_interest}")    E-set a limit and perform a pattern .  def pattern(limit):  for i in range(1, limit + 1):  print('\*' \* i)  pattern(5)    F-set a limit and perform fibonacci series.  def fibonacci(limit):  fib\_series = [0, 1]  while fib\_series[-1] + fib\_series[-2] <= limit:  fib\_series.append(fib\_series[-1] + fib\_series[-2])  return fib\_series  limit\_f = 20  result\_f = fibonacci(limit\_f)  print(result\_f)    G-factorial number.  def factorial(n):  if n == 0 or n == 1:  return 1  else:  return n \* factorial(n - 1)  number\_g = 10  result\_g = factorial(number\_g)  print(result\_g)    H-string counts the length.  def string\_length(s):  return len(s)  string\_h = "Hello, World!"  result\_h = string\_length(string\_h)  print(result\_h)    I-reverse the string.  def reverse\_string(s):  return s[::-1]  string\_i = "vrundali"  result\_i = reverse\_string(string\_i)  print(result\_i)    J-prime number or not.  def prime(n):  if n < 2:  return False  for i in range(2, int(n\*\*0.5) + 1):  if n % i == 0:  return False  return True  number = 13  result = prime(number)  print(result)    K-Check whether the entered character is constant or vowel.  def check\_vowel\_or\_consonant(char):  vowels = "aeiouAEIOU"    if char.isalpha():  if char in vowels:  return f"{char} is a vowel."  else:  return f"{char} is a consonant."  else:  return "Invalid input. Please enter a valid alphabet."  char\_input = input("Enter a character: ")  result = check\_vowel\_or\_consonant(char\_input)  print(result)    L-Check whether the string is Palindrome.  def is\_palindrome(string):  cleaned\_string = ''.join(char.lower() for char in string if char.isalnum())  return cleaned\_string == cleaned\_string[::-1]  string\_input = input("Enter a string: ")  if is\_palindrome(string\_input):  print("The string is a palindrome.")  else:  print("The string is not a palindrome.")    M-Pass 3 arguments and find maximum among them.  def find\_maximum(a, b, c):  return max(a, b, c)  arg1 = int(input("Enter the first number: "))  arg2 = int(input("Enter the second number: "))  arg3 = int(input("Enter the third number: "))  max\_value = find\_maximum(arg1, arg2, arg3)  print(f"The maximum value among {arg1}, {arg2}, and {arg3} is: {max\_value}")    N=Count the space in the string.  def count\_spaces(input\_string):  space\_count = 0  for char in input\_string:  if char.isspace():  space\_count += 1  return space\_count  user\_input = input("Enter a string: ")  result = count\_spaces(user\_input)  print(f"The number of spaces in the string is: {result}") |
| 61 | create a function of compound interest. |
| CODE | def compound\_interest(principal, rate, time, compounding\_frequency):  compound\_interest = principal \* (1 + rate / compounding\_frequency) \*\* (compounding\_frequency \* time)  return compound\_interest  principal\_amount = 1000  annual\_interest\_rate = 0.05  investment\_time = 3  compounding\_frequency = 12  result = compound\_interest(principal\_amount, annual\_interest\_rate, investment\_time, compounding\_frequency)  print("Compound Interest:", round(result, 2)) |
| OUTPUT |  |
| 62 | Create a function of the pyramid of palindromes. |
| CODE | def generate\_palindrome\_pyramid(levels):    for i in range(levels):  row = " ".join(str(j) for j in range(i + 1))  palindrome = row + row[-2::-1]  spaces = " " \* (levels - i - 1)  print(spaces + palindrome)  num\_levels = 5  generate\_palindrome\_pyramid(num\_levels) |
| OUTPUT |  |
| 63 | Create a function of the pyramid of even numbers. |
| CODE | def generate\_even\_odd\_pyramid(levels, use\_even=True):    for i in range(levels):  numbers = range(2, 2 \* (i + 1) + 1, 2) if use\_even else range(1, 2 \* i + 2, 2)  row = " ".join(str(num) for num in numbers)  spaces = " " \* (levels - i - 1)  print(spaces + row)  num\_levels = 5  generate\_even\_odd\_pyramid(num\_levels, use\_even=True) |
| OUTPUT |  |
| 64 | Create a function of the pyramid of odd numbers. |
| CODE | def generate\_odd\_number\_pyramid(levels):    for i in range(levels):    numbers = range(1, 2 \* i + 2, 2)  row = " ".join(str(num) for num in numbers)  spaces = " " \* (levels - i - 1)  print(spaces + row)    num\_levels = 5  generate\_odd\_number\_pyramid(num\_levels) |
| OUTPUT |  |
| 65 | function |
| CODE | def abc(p,r,n):  print((p\*r\*n)/100)  p=2000  r=10  n=1  abc(p,r,n)  #abc(1000,10,1) |
| OUTPUT |  |
| 66 | function |
| CODE | def abc(fruits="kiwi"):  print(fruits)  abc("apple")  abc()  abc("mango") |
| OUTPUT |  |
| 67 | function |
| CODE | def abc(fruits="kiwi",fruits1="dragon"):  print(fruits,fruits1)  abc("apple")  abc()  abc("mango") |
| OUTPUT |  |
| 68 | Return function |
| CODE | def abc(x):  return x\*2  print(abc(3)) |
| OUTPUT |  |
| 69 | Function |
| CODE | def abc(a,b,\*,d):  print(a+b\*d)  abc(a=10,b=1,d=3) |
| OUTPUT |  |
| 70 | write a python function with the name frequency which takes string from the user and return the word removing space , and punctuation. |
| CODE | import string  def frequency(input\_str):  cleaned\_words = []  words = input\_str.translate(str.maketrans('', '', string.punctuation)).split()  for word in words:  cleaned\_word = ''.join(char for char in word if char.isalnum())  if cleaned\_word:  cleaned\_words.append(cleaned\_word)  return cleaned\_words  user\_input = input("Enter a string: ")  cleaned\_words = frequency(user\_input)  print("Cleaned words:", cleaned\_words) |
| OUTPUT |  |
| 71 | create a function in python where you need to perform a palindrome . |
| CODE | def check\_palindrome(input\_str):  input\_str = input\_str.lower().replace(" ", "")  if input\_str == input\_str[::-1]:  return True  else:  return False  user\_input = input("Enter a number: ")  if check\_palindrome(user\_input):  print("The input number is a palindrome.")  else:  print("The input number is not a palindrome.") |
| OUTPUT |  |
| 72 | write a python function to reverse the string . |
| CODE | def reverse\_string(input\_str):  return input\_str[::-1]  user\_input = input("Enter a string: ")  reversed\_str = reverse\_string(user\_input)  print("Reversed string:", reversed\_str) |
| OUTPUT |  |
| 73 | you need to print a pattern using a function .  1  2 2  3 3 3  A  B B  C C C |
| CODE | def print\_pattern(n):  for i in range(1, n+1):  for j in range(i):  print(i, end=" ")  print()  n = 3  print\_pattern(n)  def print\_pattern(rows):  for i in range(rows):  print((chr(65 + i) + ' ') \* (i+1))  rows = 3  print\_pattern(rows) |
| OUTPUT |  |
| 74 | write a python function print table of ten. |
| CODE | def print\_table\_of\_ten(rows):  for i in range(1, rows+1):  result = 10 \* i  print(f"10 x {i} = {result}")  rows = 10  print\_table\_of\_ten(rows) |
| OUTPUT |  |
| 75 | Print first 50 even numbers using for loop in range |
| CODE | #print first 50 number in range:  for i in range(1,101):  if i%2==0:  print(i,end=" ")  else:  pass |
| OUTPUT |  |
| 76 | Nested for loop  Inner loop is consider column  Outer loop is consider row |
| CODE | #Nested for loop:  list1 = [1,2,3]  list2 = [7,8,9]  for i in list1: #outer loop  for j in list2:  print(i)  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  list1 = [1,2,3]  list2 = [7,8,9]  for i in list1: #outer loop  for j in list2:  print(j)  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  #outer loop in i print:  for i in range (5):  for j in range(5):  print(i+1,end=" ")  print()  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  #outer loop in j print:  for i in range (5):  for j in range(5):  print(j+1,end=" ")  print() |
| OUTPUT |  |
| 77 | Range program to print  1  1 2  1 2 3  1 2 3 4  1 2 3 4 5 |
| CODE | #print pattern in range  rows = 5  for i in range(1, rows + 1):  for j in range(1, i + 1):  print(j, end=" ")  print() |
| OUTPUT |  |
| 78 | capitalize(): return string with first letter capital => str.capitalize() |
| CODE | # first letter capital  original\_string = "rk university"  capitalized\_string = original\_string.capitalize()  print(capitalized\_string) |
| OUTPUT |  |
| 79 | count number of characters and number of digits from str2 |
| CODE | str1 = "rk university"  str2 = "Kasturbadham - 360020"  #count number of characters and number of digits from str2  alpha=0  digit=0  for i in str2:  if(i.isalpha()):  alpha=alpha+1  elif(i.isdigit()):  digit=digit+1  print("Number of alphabetics in given string is:",alpha)  print("Number of digit in given string is:",digit) |
| OUTPUT |  |
| 80 | PASSWORD |
| CODE | print("Enter password:")  passwd=input()  hasLength=False  hasUcase=False  hasLcase=False  hasDigit=False  hasSpChar=False  if(len(passwd)>=8):  hasLength=True  for i in passwd:  if (i.isupper()):  hasUcase=True  elif(i.islower()):  hasLcase=True  elif(i.isdigit()):  hasDigit=True  elif(i=='!' or i=='@' or i=='#' or i=='$' or i=='%' or i=='^' or i=='&' or i=='\*'):  hasSpChar=True  if hasLength and hasUcase and hasLcase and hasDigit and hasSpChar:  print("Password is strong. Good job!")  else:  print("Password is not strong. Please make sure it meets the following criteria:")  if not hasLength:  print("- Minimum length of 8 characters.")  if not hasUcase:  print("- At least one uppercase letter.")  if not hasLcase:  print("- At least one lowercase letter.")  if not hasDigit:  print("- At least one digit.")  if not hasSpChar:  print("- At least one special character (!, @, #, $, %, ^, &, \*).") |
| OUTPUT |  |
| 81 | BUTTON |
| CODE | from tkinter import \*  a=Tk()  a.geometry('350x350')  Label(a,text="Name").pack()  def b1():  Entry(a).pack()  Button(a,text="click here",command=b1).pack()  a.mainloop() |
| OUTPUT |  |
| 82 | calculator for 2 numbers tkinter import addition, subtraction,division amd multiplication in python code. |
| CODE | import tkinter as tk  def calculate():  num1 = float(entry1.get())  num2 = float(entry2.get())    result\_label.config(text="")    if operation.get() == "Addition":  result = num1 + num2  elif operation.get() == "Subtraction":  result = num1 - num2  elif operation.get() == "Multiplication":  result = num1 \* num2  elif operation.get() == "Division":  if num2 != 0:  result = num1 / num2  else:  result\_label.config(text="Error: Division by zero")  return    result\_label.config(text="Result: " + str(result))  root = tk.Tk()  root.title("Simple Calculator")  entry1 = tk.Entry(root)  entry1.pack()  entry2 = tk.Entry(root)  entry2.pack()  operation = tk.StringVar()  operation.set("Addition")  operations\_menu = tk.OptionMenu(root, operation, "Addition", "Subtraction", "Multiplication", "Division")  operations\_menu.pack()  calculate\_button = tk.Button(root, text="Calculate", command=calculate)  calculate\_button.pack()  result\_label = tk.Label(root)  result\_label.pack()  root.mainloop() |
| OUTPUT |  |
| 83 | write a python function to calculate the string( tkinter import) |
| CODE | import tkinter as tk  import math  def calculate\_expression(expression):  try:  result = eval(expression)  return result  except Exception as e:  return "Error: " + str(e)  def on\_calculate():  expression = entry.get()  result = calculate\_expression(expression)  result\_label.config(text="Result: " + str(result))  root = tk.Tk()  root.title("String Calculator")  entry = tk.Entry(root)  entry.pack()  calculate\_button = tk.Button(root, text="Calculate", command=on\_calculate)  calculate\_button.pack()  result\_label = tk.Label(root)  result\_label.pack()  root.mainloop() |
| OUTPUT |  |
| 84 | 1-create 1 calculator ajio  b1=+  b2=-  b3=\*  b4=/  text box 1=enter a 1 value  text box 2= enter a 2 value  text box 3=answer  in this program there should be label about text type is arial and calculator color is dark gray |
| CODE | import tkinter as tk  from tkinter import font  class Calculator:  def \_\_init\_\_(self, root):  self.root = root  self.root.title("Calculator Ajio")  self.root.configure(bg='dark gray')  custom\_font = font.Font(family="Arial", size=12)  tk.Label(root, text="Enter Value 1:", font=custom\_font, bg='dark gray').grid(row=0, column=0)  tk.Label(root, text="Enter Value 2:", font=custom\_font, bg='dark gray').grid(row=1, column=0)  tk.Label(root, text="Answer:", font=custom\_font, bg='dark gray').grid(row=2, column=0)  self.entry1 = tk.Entry(root, width=15, font=custom\_font)  self.entry1.grid(row=0, column=1)  self.entry2 = tk.Entry(root, width=15, font=custom\_font)  self.entry2.grid(row=1, column=1)  self.answer = tk.Entry(root, width=15, font=custom\_font)  self.answer.grid(row=2, column=1)  buttons = [  ('+', lambda: self.calculate('+')),  ('-', lambda: self.calculate('-')),  ('\*', lambda: self.calculate('\*')),  ('/', lambda: self.calculate('/'))  ]  row = 3  col = 0  for button\_text, command in buttons:  tk.Button(root, text=button\_text, width=2, height=2, font=custom\_font, command=command).grid(row=row, column=col)  col += 1  def calculate(self, operator):  try:  value1 = float(self.entry1.get())  value2 = float(self.entry2.get())  if operator == '+':  result = value1 + value2  elif operator == '-':  result = value1 - value2  elif operator == '\*':  result = value1 \* value2  elif operator == '/':  if value2 == 0:  raise ZeroDivisionError  result = value1 / value2  self.answer.delete(0, tk.END)  self.answer.insert(tk.END, str(result))  except ValueError:  self.answer.delete(0, tk.END)  self.answer.insert(tk.END, "Invalid input")  except ZeroDivisionError:  self.answer.delete(0, tk.END)  self.answer.insert(tk.END, "Cannot divide by zero")  if \_\_name\_\_ == "\_\_main\_\_":  root = tk.Tk()  calculator = Calculator(root)  root.mainloop() |
| OUTPUT |  |
| 85 | create a class calculator which is having 5 member function  1-square of the given number  2-area of the circle  3-perform power of given number  4-given string in reverse  5-any 10 numbers insert by users it perform shorting |
| CODE | class Calculator:  def square(self, num):  return num \*\* 2  def area\_of\_circle(self, radius):  return 3.14159 \* (radius \*\* 2)  def power(self, base, exponent):  return base \*\* exponent  def reverse\_string(self, input\_str):  return input\_str[::-1]  def sort\_numbers(self, numbers):  return sorted(numbers)  calc = Calculator()  print("Square of 5:", calc.square(5))  print("Area of circle with radius 3:", calc.area\_of\_circle(3))  print("2 raised to the power of 3:", calc.power(2, 3))  print("Reverse of 'hello':", calc.reverse\_string("hello"))  numbers = [5, 3, 8, 1, 9, 2, 7, 4, 6, 10]  print("Sorted numbers:", calc.sort\_numbers(numbers)) |
| OUTPUT |  |
| 86 | take 2 string user function (static)using set.list,tuple,dictionary compare the string do intersection,short the string ,upend the string,insert,delete,count,compared |
| CODE | class StringUtils:  @staticmethod  def string\_operations(str1, str2):    set1 = set(str1)  set2 = set(str2)  intersection = set1.intersection(set2)  print("Intersection of strings:", intersection)  sorted\_str1 = ''.join(sorted(str1))  sorted\_str2 = ''.join(sorted(str2))  print("Sorted strings:", sorted\_str1, sorted\_str2)  concatenated\_str = str1 + str2  print("Concatenated string:", concatenated\_str)  inserted\_str = str1[:3] + 'XYZ' + str1[3:]  print("String with 'XYZ' inserted:", inserted\_str)  deleted\_str = str2[1:]  print("String with first character deleted:", deleted\_str)  count\_e = str1.count('e')  print("Count of 'e' in first string:", count\_e)  if len(str1) == len(str2):  print("Lengths of the strings are equal")  else:  print("Lengths of the strings are not equal")  str1 = "vrundali"  str2 = "boghara"  StringUtils.string\_operations(str1, str2) |
| OUTPUT |  |
| 87 | use the concept of the function design the pattern first you need to take user choice the pattern can be  1-\*$..all symbol  2-integers  3-character. |
| CODE | def generate\_pattern(choice):  pattern = ""  if choice == 1:  pattern = "!@#$%^&\*()\*-+/?><[]"  elif choice == 2:  pattern = "12345"  elif choice == 3:  pattern = "abcde"  return pattern  def main():  print("Choose a pattern:")  print("1. symbols")  print("2. Integers")  print("3. Characters")  choice = int(input("Enter your choice: "))  if choice not in [1, 2, 3]:  print("Invalid choice!")  return  pattern = generate\_pattern(choice)  print("Generated Pattern:", pattern)  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| OUTPUT |  |
| 88 | EXCEPTION HANDLING  except |
| CODE | x=10  y=0  try:  print(x/y)  except:  print("Error") |
| OUTPUT |  |
| 89 | EXCEPTION HANDLING |
| CODE | x=10  y=0  try:  print(x/y)  except SyntaxError:  print("Error")  except:  print("Error1")  finally:  print("Successful") |
| OUTPUT |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

–what is python in inverted and author names?

Guido van Rossum began working on Python in the late 1980s as a successor to the **ABC programming language** and first released it in 1991 as Python 0.9.0.

Python was conceived in the late 1980s by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) at [Centrum Wiskunde & Informatica](https://en.wikipedia.org/wiki/Centrum_Wiskunde_%26_Informatica) (CWI) in the [Netherlands](https://en.wikipedia.org/wiki/Netherlands) as a successor to the [ABC programming language](https://en.wikipedia.org/wiki/ABC_(programming_language)), which was inspired by [SETL](https://en.wikipedia.org/wiki/SETL),capable of [exception handling](https://en.wikipedia.org/wiki/Exception_handling) and interfacing with the [Amoeba](https://en.wikipedia.org/wiki/Amoeba_(operating_system)) operating system.Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's "[benevolent dictator for life](https://en.wikipedia.org/wiki/Benevolent_dictator_for_life)", a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a five-member Steering Council to lead the project.

–features of python.

Easy to Learn. One of the most significant features of Python is that it is very easy to learn.

Easy to Code.

Interpreted Language.

Free and Open Source.

Object-Oriented Language.

Cross-Platform Language.

Extensive Feature.

High-Level Language.

–Advantages and disadvantages of python.

Advantages

Presence of Third Party Modules.

Extensive Support Libraries.

Open Source and Community Development.

Learning Ease and Support Available.

User-friendly Data Structures.

Productivity and Speed.

–Disadvantages

Slow Speed. Python is a dynamically typed and interpreted language, as previously mentioned. Not Memory Efficient. Python must make a small tradeoff to make development simpler.

Weak Mobile Computing. Most of the time, server-side programming calls for Python.

Database Access.

Runtime Errors.

–difference between list tuple

| Sno | LIST | TUPLE |
| --- | --- | --- |
| 1 | [Lists](https://www.geeksforgeeks.org/python-lists/) are [mutable](https://www.geeksforgeeks.org/mutable-vs-immutable-objects-in-python/) | [Tuples](https://www.geeksforgeeks.org/python-tuples/) are immutable |
| 2 | The implication of iterations is Time-consuming | The implication of iterations is comparatively Faster |
| 3 | The list is better for performing operations, such as insertion and deletion. | A Tuple data type is appropriate for accessing the elements |
| 4 | Lists consume more memory | Tuple consumes less memory as compared to the list |
| 5 | Lists have several built-in methods | Tuple does not have many built-in methods. |
| 6 | Unexpected changes and errors are more likely to occur | In a tuple, it is hard to take place. |

LIST

A list is a data structure in Python that is a mutable, or changeable, ordered sequence of elements. Each element or value that is inside of a list is called an item. Just as strings are defined as characters between quotes, lists are defined by having values between square brackets [ ] .

TUPLE

Python tuples are a type of data structure that is very similar to lists. The main difference between the two is that tuples are immutable, meaning they cannot be changed once they are created. This makes them ideal for storing data that should not be modified, such as database records.