|  |  |
| --- | --- |
| Code | Output |
| #passenger.py  no\_of\_passenger =int(input("Enter No of Passenger:"))  while no\_of\_passenger!=0 :      # print (no\_of\_passenger)      nb=input("Enter No of Baggage:")      no\_of\_baggage=int(nb)      while no\_of\_baggage!=0:          status=input("Enter Status of the baggage(Checked-C/Not Checked -N):")          print("Passenger",no\_of\_passenger,"--Baggage",no\_of\_baggage,"is",status)          no\_of\_baggage-=1      no\_of\_passenger-=1 | Enter No of Passenger:2  Enter No of Baggage:2  Enter Status of the baggage(Checked-C/Not Checked -N):C  Passenger 2 --Baggage 2 is C  Enter Status of the baggage(Checked-C/Not Checked -N):N  Passenger 2 --Baggage 1 is N  Enter No of Baggage:3  Enter Status of the baggage(Checked-C/Not Checked -N):C  Passenger 1 --Baggage 3 is C  Enter Status of the baggage(Checked-C/Not Checked -N):C  Passenger 1 --Baggage 2 is C  Enter Status of the baggage(Checked-C/Not Checked -N):C  Passenger 1 --Baggage 1 is C |
| def convert\_temp(temp, unit):      if unit=='C':          res\_temp=temp\*9/5+32      elif unit=='F':          res\_temp=(temp-32)\*5/9      else:          res\_temp="Incorrect Unit"      return res\_temp    print("Temp - ",convert\_temp(104,'A')) | Temp - 219.2 |
| #Break  for i in range(1,6):      if i==3:          break      print(i)  # Continue  for i in range(1,6):      if i==3:          continue      print(i) | 1  2  1  2  4  5 |
| def calculate\_total\_ticket\_cost(no\_of\_adults, no\_of\_children):      rate\_of\_adult=37550      rate\_of\_child = rate\_of\_adult/3      total\_ticket\_cost=no\_of\_adults\*rate\_of\_adult+no\_of\_children\*rate\_of\_child      total\_ticket\_cost+=total\_ticket\_cost\*0.07      total\_ticket\_cost-=total\_ticket\_cost\*0.10        return total\_ticket\_cost  #Provide different values for no\_of\_adults, no\_of\_children and test your program  total\_ticket\_cost=calculate\_total\_ticket\_cost(5,2)  print("Total Ticket Cost:",total\_ticket\_cost) | Total Ticket Cost: 204910.35 |
| # Program to find the ASCII value of the given character  c = 'a'  print("The ASCII value of '" + c + "' is", ord(c)) | The ASCII value of 'a' is 97 |
| # WAP to convert decimal to binary  for i in range(1,10001):      b\_no=s=""      temp=i      while i>0:          rem=str(i%2)          b\_no=b\_no+rem          i//=2      for j in b\_no:          s = j + s      print("Decimal value of ",temp,":",s) |  |
| # For Loop  # for i in 1,2,3,4,5:  #     print(i)  # sum=0  # for i in 1,2,3,4,5:  #     sum+=i  #     print(sum)  # Use of Range  # for i in range(1,6):  #     print(i)  # Another use of Range Function  # Three parameter 1= start 10 = end , 2 = increment value  for i in range(1,10,2):      # print(i)      print(i,end="") # prints in same line |  |
| # Python Program to find the L.C.M. of two input number  def compute\_lcm(x, y):     # choose the greater number     if x > y:         greater = x     else:         greater = y     while(True):         if((greater % x == 0) and (greater % y == 0)):          #    print(greater)             lcm = greater             break         greater += 1     return lcm  num1 = int(input("Enter first No : "))  num2 = int(input("Enter second No : "))  print("The L.C.M. is", compute\_lcm(num1, num2)) | Enter first No : 15  Enter second No : 17  The L.C.M. is 255 |
| import time  def tryit1():      for i in range(1,1000000):          x=time.strftime("%d/%m/%Y")    t = time.time()  tryit1()  print('duration:', time.time()-t) | duration: 4.709304332733154 |
| # Write a python program to convert one data capacity to another.  # Example:  # 4 terrabytes = ? GB  no=int(input("Enter no to convert : "))  print(no," terrabytes = ", (no\*1024) , " GB") | Enter no to convert : 71  71 terrabytes = 72704 GB |
| # list of list  my\_list =[["abc","def"],["ghi"],["alpha","beta"],2]  # print(my\_list[0])  # print(my\_list[2][0])  for sample\_list in my\_list:      for element in sample\_list:          print(element) | abc  def  ghi  alpha  beta  Traceback (most recent call last):  File "d:/STUDY/INFY/GENERIC/PROGRAM/demo.py", line 7, in <module>  for element in sample\_list:  TypeError: 'int' object is not iterable |
| my\_list =["alpha","beta","gama","delta"]  print(my\_list[2:]) #from 2 to last  print(my\_list[:]) # complete list  print(my\_list[:2]) # first and second element element of index 1 | ['gama', 'delta']  ['alpha', 'beta', 'gama', 'delta']  ['alpha', 'beta'] |
| my\_list=[0]\*5  for index in range(1,5):      my\_list[index]=(index-1)\*10  print(my\_list)  list\_of\_airline=["AI","SJ","JA","EM","BA"]  #slicing can be done using : 1:4  1= starting index, 4 = ending index  # it creates new list as per the given index  '''  [ 1  2  3  4  ]  [ 0  1  2  3  ] positive indexing  [ -4  -3  -2  -1  ] negative indexing  indexing always move from left to right  [1:4:2]  1 - starting index  4 - ending index  2 - increment  '''  new\_list=list\_of\_airline[1:2,3:4]  print(new\_list) | [0, 0, 10, 20, 30]  Traceback (most recent call last):  File "d:/STUDY/INFY/GENERIC/PROGRAM/demo.py", line 21, in <module>  new\_list=list\_of\_airline[1:2,3:4]  TypeError: list indices must be integers or slices, not tuple |
| counter=0  while counter<=9:      if counter%2==0:          pass      else:          print(counter,end=" ")      counter+=1 | 1 3 5 7 9 |
| n1=input()  n2=input()  num1=int(n1)  num2=int(n2)  while num1>=2:      if num1>num2:          num1=num1/2      else:          print(num1)          break | 5  6  5 |
| for number in 10,15:      for counter in range(1,3):          print(number\*counter, end=" ") | 10 20 15 30 |
| for num in 23,45,50,65,76,90:      if num%5!=0:          continue      if num%10==0:          print(num, end=" ")          continue      if num%3==0:          print(num,end=" ") | 45 50 90 |
| number=28  for num in range(25,30):      if number>num:          print(num)      else:          print(num)          break | 25  26  27  28 |
| def count\_names(name\_list):      count1=0      count2=0        #start writing your code here      for i in name\_list:          if(((i.find("at"))==1) and len(i)==3):              count1+=1          if(i.find("at"))>=0:              count2+=1      print("\_at -> ",count1)      print("%at% -> ",count2)  name\_list=["Hat","Cat","rabbit","matter"]  # name\_list=['Rat', 'saturday']  count\_names(name\_list) | \_at -> 2  %at% -> 3 |
| row1 = (101,"Dallas",3.5)  row2 = (102,"Atlanta",5.6)  row3 = (103,"Tokyo",9.8)  table = [row1,row2,row3]  print(table[0])  print(table[1])  print(table[2]) | (101, 'Dallas', 3.5)  (102, 'Atlanta', 5.6)  (103, 'Tokyo', 9.8) |
| # String Funstions  boarding\_call="Good Evening, this is the final call to AI passengers for the flight AI 466 which is planned to take off at 8.40A.M."  if(boarding\_call.startswith("Good Evening")):      print(boarding\_call.replace("Good Evening","Good Morning"))  if(boarding\_call.find("AI"))>=0:      print("Welcome to Air India.")  if(boarding\_call.endswith("A.M.")):      print("Passengers are requested to have their breakfast.")  a=boarding\_call.split(" ")  for i in a:      if(i.isdigit()):          print("Flight Number is specified to the passengers.")  print("Total number of times flight service name is specified in the boarding call:",boarding\_call.count("AI"))  message="Thank you all..Have a nice journey!"  print(message.upper())  print(message.lower()) | Good Morning, this is the final call to AI passengers for the flight AI 466 which is planned to take off at 8.40A.M.  Welcome to Air India.  Passengers are requested to have their breakfast.  Flight Number is specified to the passengers.  Total number of times flight service name is specified in the boarding call: 2  THANK YOU ALL..HAVE A NICE JOURNEY!  thank you all..have a nice journey! |
| def generate\_ticket(airline,source,destination,no\_of\_passengers):      ticket\_no=100      ticket\_number\_list=[]      while no\_of\_passengers>0:          temp\_ticket=airline+':'+source[:3]+':'+destination[:3]+':'+str((ticket\_no+1))          ticket\_no+=1          ticket\_number\_list.append(temp\_ticket)          no\_of\_passengers-=1      return ticket\_number\_list[-5:]  #Provide different values for airline,source,destination,no\_of\_passengers and test your program  print(generate\_ticket("AI","Bangalore","London",10)) | ['AI:Ban:Lon:106', 'AI:Ban:Lon:107', 'AI:Ban:Lon:108', 'AI:Ban:Lon:109', 'AI:Ban:Lon:110'] |
| #Creating a string  pancard\_number="AABGT6715H"  #Length of the string  print("Length of the PAN card number:", len(pancard\_number))  #Concatenating two strings  name1 ="PAN "  name2="card"  name=name1+name2  print(name)  print("Iterating the string using range()")  for index in range(0,len(pancard\_number)):      print(pancard\_number[index])    print("Iterating the string using keyword in")  for value in pancard\_number:      print(value)  print("Searching for a character in string")  if "Z" in pancard\_number:      print("Character present")  else:      print("Character is not present")  #Slicing a string  print("The numbers in the PAN card number:", pancard\_number[5:9])  print("Last but one 3 characters in the PAN card:",pancard\_number[-4:-1])  # pancard\_number[2]="A" #This line will result in an error, i.e., string is immutable  print(pancard\_number) | Length of the PAN card number: 10  PAN card  Iterating the string using range()  A  A  B  G  T  6  7  1  5  H  Iterating the string using keyword in  A  A  B  G  T  6  7  1  5  H  Searching for a character in string  Character is not present  The numbers in the PAN card number: 6715  Last but one 3 characters in the PAN card: 715  AABGT6715H |
| # String is another type of collection  # String is also immutable  # String Follow same concept of indexing  # String can be added  # String Declaration  # string\_one=""  # string\_two=''  # String Addition  string\_one="alpha"  string\_two='beta'  string\_one=string\_one+'delta'  print(string\_one)  #  when we add anything to a string new value is allocated to the string. | alphadelta  Enter a worda  Enter a worda  Enter a wordd  Enter a wordalpha |
| #WAP to take input untill alpha is given  a=""  while (a!="alpha"):      a=input("Enter a word") |  |
| gem\_list=["Emerald","Ivary","Jasper","Ruby","Garnet"]  price\_list=[1760,2119,1599,3920,3999]  required\_gem="Ruby"  required\_index=None  # for i in range(0,len(gem\_list)):  #     if gem\_list[i]==required\_gem:  #         required\_index=i  # print("Price of required Item is ",price\_list[required\_index])  reqd\_gems=["Ivary","Emerald","Garnet"]  reqd\_quantity=[3,10,12]  total\_price=0  for i in range(0,len(reqd\_gems)):      if(reqd\_gems[i] in gem\_list):          print("Found")      qty = reqd\_quantity[i]      for j in range(0,len(gem\_list)):          if reqd\_gems[i]==gem\_list[j]:              required\_index=j      total\_price += price\_list[required\_index]\*qty | Found  Found  Found |
| '''  b=(1,2,3)  tuple value cant be change  b=1, - tuple  b=(1) - int  '''  b=(1,2,3)  print(b[2]) |  |
| #set.py  # Problem Statement  # Try out the below code for usage of set and observe the results.  #list of passengers  passengers\_list=["George","Annie", "Jack","Annie","Henry", "Helen","Maria","George","Jack","Remo"]  #set function - removes the duplicates from the list and returns a set  unique\_passengers=set(passengers\_list)  print(unique\_passengers)  #creating a set  flight\_set={500,520,600,345,520,634,600,500,200,200}  print(flight\_set)  flights\_at\_src = ["AI230","BA944","EM395","AI704","BA944","AI704"]  flights\_at\_dest = ["SI107","AI034","EM395","AI704","BA802","SI236"]  print(flights\_at\_src)  print(flights\_at\_dest)  #Creating list of unique flights at source and destination  uniq\_src\_flights = set(flights\_at\_src)  uniq\_dest\_flights = set(flights\_at\_dest)  print(uniq\_src\_flights)  print(uniq\_dest\_flights)  #setA-setB -> Gives the elements that are only in setA  #List of flights only at source airport  flights\_only\_at\_src = uniq\_src\_flights-uniq\_dest\_flights  print(flights\_only\_at\_src)  #setA&setB -> Gives the common elements between setA and setB  #List of flights common to source and destination airports  common\_flights=uniq\_src\_flights&uniq\_dest\_flights  print(common\_flights)  #setA|setB -> merges setA and setB after removing duplicates  #List of all flights at source and destination airports  all\_flights=uniq\_src\_flights|uniq\_dest\_flights  print(all\_flights) | {'Annie', 'Helen', 'Remo', 'George', 'Jack', 'Henry', 'Maria'}  {520, 200, 500, 600, 345, 634}  ['AI230', 'BA944', 'EM395', 'AI704', 'BA944', 'AI704']  ['SI107', 'AI034', 'EM395', 'AI704', 'BA802', 'SI236']  {'AI230', 'AI704', 'BA944', 'EM395'}  {'EM395', 'AI704', 'SI107', 'BA802', 'AI034', 'SI236'}  {'AI230', 'BA944'}  {'AI704', 'EM395'}  {'EM395', 'AI230', 'AI704', 'BA944', 'SI107', 'BA802', 'AI034', 'SI236'} |
| #Creating a dictionary  crew\_details={              "Pilot":"Kumar",              "Co-Pilot":"Raghav",              "Head-Strewardess":"Malini",              "Stewardess":"Mala"  }  print(crew\_details["Pilot"])  print("\nIterating the dictionary using items function")  for key,value in crew\_details.items():      print(key,":",value)  #Usually while working with dictionary, you will be interested in specific values.  #Let’s find the value of all pilots from crew\_details.  print("\nIterating the dictionary using keyword 'in'")  for key in crew\_details:      if(key=="Pilot" or key=="Co-Pilot"):          print(crew\_details[key])  #Note: Dictionary being unordered, the order of the values being displayed may vary during each execution of the above for loop.  #Dictionaries are mutable  crew\_details["Pilot"]="James" # Here the value for key "Pilot" is being updated to "James"  print("\nAfter modifying the value of Pilot:", crew\_details["Pilot"])  print("------------------------------------------------------------------")  print("Before update:")  # Usage of get method()  print("Co-pilot:",crew\_details.get("Co-Pilot"))  #Usage of update method()  crew\_details.update({"Flight Attendant":"Jane", "Co-pilot":"Henry"})  print("\nAfter update:")  print("Co-pilot:",crew\_details.get("Co-pilot"))  print("Flight Attendant:",crew\_details["Flight Attendant"]) | Iterating the dictionary using items function  Pilot : Kumar  Co-Pilot : Raghav  Head-Strewardess : Malini  Stewardess : Mala  Iterating the dictionary using keyword 'in'  Kumar  Raghav  After modifying the value of Pilot: James  ------------------------------------------------------------------  Before update:  Co-pilot: Raghav  After update:  Co-pilot: Henry  Flight Attendant: Jane |
| '''  Given a string containing uppercase characters (A-Z), compress the string using Run Length encoding. Repetition of character has to be replaced by storing the length of that run.  Write a python function which performs the run length encoding for a given String and returns the run length encoded String.  '''  def encode(message):      counter = 1      result = ""      previousLetter = message[0]      if len(message)==1:        return str(1) + message[0]      for i in range(1,len(message),1):          if not message[i] == previousLetter:              result += str(counter) + message[i-1]              previousLetter = message[i]              counter = 1          else:              counter += 1          if i == len(message)-1:                  result += str(counter) + message[i]      return result  #Provide different values for message and test your program  encoded\_message=encode("ABBBBCCCCCCCCAB")  print(encoded\_message) | 1A4B8C1A1B |