|  |  |  |
| --- | --- | --- |
| **with** open(**"binary"**,**'bw'**) **as** bin\_file:  **for** i **in** range(1,18):  bin\_file.write(bytes([i]))  **with** open(**"binary"**, **'br'**) **as** binu:  **for** b **in** binu:  print(b)  print() **with** open(**"binsex"**, **'bw'**) **as** binx:  binx.write(bytes(range(1,18)))  **with** open(**"binary"**, **'br'**) **as** binxr:  **for** b **in** binxr:  print(b) | b'\x01\x02\x03\x04\x05\x06\x07\x08\t\n'  b'\x0b\x0c\r\x0e\x0f\x10\x11'  b'\x01\x02\x03\x04\x05\x06\x07\x08\t\n'  b'\x0b\x0c\r\x0e\x0f\x10\x11' | |
| **import** pickle  imelda=(**"More mayham"**, **"Imelda may"**, **"2011"**, ((1,**"boobs"**),(2,**"nipple"**),(3,**"penus"**),(4,**"sex"**)))  **with** open(**"binary.pickle"**,**'wb'**) **as** pic:  pickle.dump(imelda, pic)  **with** open(**"binary.pickle"**,**'rb'**) **as** pick:  im\_read=pickle.load(pick)  print(im\_read)  print(**"\*"**\*50) album, name, year, songs=imelda print(album) print(name) print(year) **for** i **in** songs:  nos, song\_nm=i  print(nos, song\_nm) | ('More mayham', 'Imelda may', '2011', ((1, 'boobs'), (2, 'nipple'), (3, 'penus'), (4, 'sex')))  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  More mayham  Imelda may  2011  1 boobs  2 nipple  3 penus  4 sex | |
| PICKLE : uses protocol version 3 : if not mentioned | | |
| **import** pickle  imelda=(**"More mayham"**, **"Imelda may"**, **"2011"**, ((1,**"boobs"**),(2,**"nipple"**),(3,**"penus"**),(4,**"sex"**))) even=list(range(0,10,2)) odd=list(range(1,10,2))  *#pushing multiple data* **with** open(**"binary.pickle"**,**'wb'**) **as** pic:  pickle.dump(imelda, pic)  pickle.dump(even, pic)  pickle.dump(odd,pic)  pickle.dump(145896, pic)  *#fetching all items* **with** open(**"binary.pickle"**,**'rb'**) **as** pick:  im\_read=pickle.load(pick) *#tuple* evn=pickle.load(pick) *#even series* od=pickle.load(pick) *#odd s* sing=pickle.load(pick) *#single value  #printing main tuple* print(im\_read) *#tuple distribution* print(**"\*"**\*50) album, name, year, songs=imelda print(album) print(name) print(year) **for** i **in** songs:  nos, song\_nm=i  print(nos, song\_nm)  *#retriving even vals* print(**"\*"**\*50) **for** i **in** evn:  print(i)  *##retriving odd vals* print(**"\*"**\*50) **for** j **in** od:  print(j)  *#retriving single* print(**"\*"**\*50) print(sing) | ('More mayham', 'Imelda may', '2011', ((1, 'boobs'), (2, 'nipple'), (3, 'penus'), (4, 'sex')))  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  More mayham  Imelda may  2011  1 boobs  2 nipple  3 penus  4 sex  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  0  2  4  6  8  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  1  3  5  7  9  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  145896 | |
| Removing a pickle:  pickle.loads(**b"cos\nsystem\n(S'del binary.pickle'\ntR"**) | | |
| Shelves : makes a database, where the values come from | | |
| **import** shelve  **with** shelve.open(**"Shelves"**) **as** dick: *#shelve is read and write as well* dick[**"a"**]=**"sex"** dick[**"b"**]=**"boobs"** dick[**"c"**]=**"penus"** dick[**"d"**]=**"vegana"** print(dick[**"a"**])  print(dick[**"c"**])  print(dick) *#shelve object* print(**"\*"**\*50) **with** shelve.open(**"shoe"**) **as** tup:  tup={**"s"**:**"fuck"**,  **"b"**:**"duck"**,  **"c"**:**"poty"**,  **"d"**:**"shitty"**}   print(tup[**"s"**])  print(tup) *#results like a dictionary* | sex  penus  <shelve.DbfilenameShelf object at 0x0030FA30>  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  fuck  {'s': 'fuck', 'b': 'duck', 'c': 'poty', 'd': 'shitty'} | |
| print(**"\*"**\*50) **with** shelve.open(**"Shelves"**) **as** tupl:  **for** key **in** tupl:  print(key) | a  b  c  d | |
| print(**"\*"**\*50) **with** shelve.open(**"Shelves"**) **as** tupl:  **del** tupl[**"a"**]  **for** key **in** tupl:  print(key) | b  c  d | |
| **import** shelve  **with** shelve.open(**"cat"**) **as** catu:  catu[**"a"**]=**"big cat"** catu[**"b"**]=**"small cat"** catu[**"c"**]=**"bojjat cat"** catu[**"d"**]=**"lomba cat"** print(catu[**"b"**]) print(**"\*"**\*50) *#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* #updating the value* **with** shelve.open(**"cat"**) **as** fat:  fat[**"n"**]=**"male cat"  for** cats **in** fat:  print(fat[cats])  *#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** print(**"\*"**\*50)  fat[**"a"**]=**"shitty cat"** *#assining the same value in a particular key,  #actually it updates the previous value* **for** cats **in** fat:  print(fat[cats])  *#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\** | small cat  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  big cat  small cat  bojjat cat  lomba cat  male cat  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  shitty cat  small cat  bojjat cat  lomba cat  male cat | |
| **import** shelve  **with** shelve.open(**"cat"**) **as** catu:  catu[**"a"**]=**"big cat"** catu[**"b"**]=**"small cat"** catu[**"c"**]=**"bojjat cat"** catu[**"d"**]=**"lomba cat"** *#an undefined key; that causes an error* **with** shelve.open(**"cat"**) **as** fat:  print(fat[**"j"**]) *#error* | Error | |
| **import** shelve  **with** shelve.open(**"cat"**) **as** catu:  catu[**"a"**]=**"big cat"** catu[**"b"**]=**"small cat"** catu[**"c"**]=**"bojjat cat"** catu[**"d"**]=**"lomba cat"** *#HANDLING THE undefined key; that causes an error issue #making it use defined #if NO KEY IS FOUND : it returns NONE* **with** shelve.open(**"cat"**) **as** fat:  **while True**:  get\_key = input(**"enter the fruit name : "**) *# taking the value from user* **if** get\_key==**"quit"**: *#if quit then break* **break** description=fat.get(get\_key) *#fetching description by the get\_key* print(description) *#printing the description* | enter the fruit name : a  big cat  enter the fruit name : c  bojjat cat  enter the fruit name : p  None  enter the fruit name : QUIT  None  enter the fruit name : quit | |
| **import** shelve  **with** shelve.open(**"cat"**) **as** catu:  catu[**"a"**]=**"big cat"** catu[**"b"**]=**"small cat"** catu[**"c"**]=**"bojjat cat"** catu[**"d"**]=**"lomba cat"** *#HANDLING THE undefined key; that causes an error issue #making it use defined #if NO KEY IS FOUND : it returns NONE #if no key is found : SHOWING MESSAGE instade NONE* **with** shelve.open(**"cat"**) **as** fat:  **while True**:  get\_key = input(**"enter the fruit name : "**) *# taking the value from user* **if** get\_key==**"quit"**: *#if quit then break* **break** description=fat.get(get\_key, **"we dont have a "** + get\_key) *#fetching description by the get\_key* print(description) *#printing the description* | enter the fruit name : a  big cat  enter the fruit name : f  we dont have a f  enter the fruit name : quit | |
| **DICTIONARY COMES IN UNORDERED SHELVES DOES THE SAME**  **import** shelve  *#creating a new shelve for order testing* **with** shelve.open(**"catx"**) **as** catu:  catu[**"b"**]=**"big cat"** catu[**"c"**]=**"small cat"** catu[**"a"**]=**"bojjat cat"** catu[**"d"**]=**"lomba cat"  with** shelve.open(**"catx"**) **as** goat:  **for** j **in** goat:  print(j + **" : "** + goat[j])  print(**"\*"**\*50)  short\_key=list(goat.keys())  short\_key.sort()   **for** i **in** short\_key:  print(i + **" : "** + goat[i]) | b : big cat  c : small cat  a : bojjat cat  d : lomba cat  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  a : bojjat cat  b : big cat  c : small cat  d : lomba cat | |
| SHELVE must have a key with STRING VALUE  DICTIONARY key, may have any value | | |
| **import** shelve  *#creating a new shelve for order testing* **with** shelve.open(**"catx"**) **as** catu:  catu[**"b"**]=**"big cat"** catu[**"c"**]=**"small cat"** catu[**"a"**]=**"bojjat cat"** catu[**"d"**]=**"lomba cat"  with** shelve.open(**"catx"**) **as** goat:  print(**"\*"**\*50)  print(goat.values()) *#returns description* print(**"\*"** \* 50)  **for** i **in** goat.values():  print(i)  print(**"\*"** \* 50)  print(goat.items()) *#returns tuple (key value pair), not dict* print(**"\*"** \* 50)  **for** j **in** goat.items():  print(j)  print(**"\*"** \* 50)  print(goat.keys()) *#returns keys* print(**"\*"** \* 50)  **for** k **in** goat.keys():  print(k) | | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ValuesView(<shelve.DbfilenameShelf object at 0x0028A1F0>)  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  big cat  small cat  bojjat cat  lomba cat  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ItemsView(<shelve.DbfilenameShelf object at 0x0028A1F0>)  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ('b', 'big cat')  ('c', 'small cat')  ('a', 'bojjat cat')  ('d', 'lomba cat')  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  KeysView(<shelve.DbfilenameShelf object at 0x0028A1F0>)  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  b  c  a  d |
| Updating SELVES | | |
| **import** shelve fruit=[**"apple"**,**"grapes"**,**"berry"**,**"banana"**] vegitables=[**"spinach"**,**"Potato"**,**"tomato"**] non\_veg=[**"meat"**,**"pork"**] drinks=[**"cola"**,**"wine"**,**"beer"**]  **with** shelve.open(**"eat"**) **as** food:  *# food["fruit"]=fruit  # food["vegitables"]=vegitables  # food["drinks"]=drinks   #printing all elements* **for** i **in** food:  print(i , food[i]) *#print(i , food[i]) is correct, print(i + food[i]) is not bcz it returns a list   #adding the rest element* food[**"non\_veg"**]=non\_veg   *#printing again* print(**"\*"**\*50)  **for** i **in** food:  print(i , food[i])   print(**"\*"**\*50)  *#updating value of a shelve  #try 1: alike list* food[**"drinks"**].append(**"Ram"**)  food[**"fruit"**].append(**"orange"**)  **for** i **in** food:  print(i , food[i])  *#it didn't work,* print(**"\*"**\*50)  *#try 2:* temp=food[**"drinks"**]  temp.append(**"Ram"**)  food[**"drinks"**]=temp   temp1=food[**"fruit"**]  temp1.append(**"orange"**)  food[**"fruit"**]=temp1   **for** i **in** food:  print(i , food[i])  print(**"\*"**\*2) print(**"\*"**\*2)  *#trying writeback parameter while updating an element* **with** shelve.open(**"eat"**, writeback=**True**) **as** fat:  **for** i **in** fat:  print(i, fat[i])   *#when writeback is true then .append will work*  print(“\*”\*2)fat[**"drinks"**].append(**"whisky"**)   **for** i **in** fat:  print(i , fat[i]) | | fruit ['apple', 'grapes', 'berry', 'banana', 'orange', 'orange']  vegitables ['spinach', 'Potato', 'tomato']  drinks ['cola', 'wine', 'beer', 'Ram', 'Ram']  non\_veg ['meat', 'pork']  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  fruit ['apple', 'grapes', 'berry', 'banana', 'orange', 'orange']  vegitables ['spinach', 'Potato', 'tomato']  drinks ['cola', 'wine', 'beer', 'Ram', 'Ram']  non\_veg ['meat', 'pork']  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  fruit ['apple', 'grapes', 'berry', 'banana', 'orange', 'orange']  vegitables ['spinach', 'Potato', 'tomato']  drinks ['cola', 'wine', 'beer', 'Ram', 'Ram']  non\_veg ['meat', 'pork']  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  fruit ['apple', 'grapes', 'berry', 'banana', 'orange', 'orange', 'orange']  vegitables ['spinach', 'Potato', 'tomato']  drinks ['cola', 'wine', 'beer', 'Ram', 'Ram', 'Ram']  non\_veg ['meat', 'pork']  fruit ['apple', 'grapes', 'berry', 'banana', 'orange', 'orange', 'orange']  vegitables ['spinach', 'Potato', 'tomato']  drinks ['cola', 'wine', 'beer', 'Ram', 'Ram', 'Ram']  non\_veg ['meat', 'pork']  \*\*  \*\*  fruit ['apple', 'grapes', 'berry', 'banana', 'orange', 'orange', 'orange']  vegitables ['spinach', 'Potato', 'tomato']  drinks ['cola', 'wine', 'beer', 'Ram', 'Ram', 'Ram', 'whisky']  non\_veg ['meat', 'pork'] |
| UPDATING WITH sync() will clear the cache, and will re initialize the list | | |
| *#trying writeback parameter while updating an element* **with** shelve.open(**"eat"**, writeback=**True**) **as** fat:  **for** i **in** fat:  print(i , fat[i])  print(**"\*"**\*10)  *#sync() method clears the cache that is why  # recently added values are cleared* fat[**"drinks"**]=drinks  fat.sync()  drinks.append(**"votka"**)   **for** i **in** fat:  print(i , fat[i]) | | fruit ['apple', 'grapes', 'berry', 'banana', 'orange', 'orange', 'orange']  vegitables ['spinach', 'Potato', 'tomato']  drinks ['cola', 'wine', 'beer', 'Ram', 'Ram', 'Ram', 'whisky']  non\_veg ['meat', 'pork']  \*\*\*\*\*\*\*\*\*\*  fruit ['apple', 'grapes', 'berry', 'banana', 'orange', 'orange', 'orange']  vegitables ['spinach', 'Potato', 'tomato']  drinks ['cola', 'wine', 'beer']  non\_veg ['meat', 'pork'] |