BINARY FILES — Exam Cheat Sheet

Class XII Computer Science (083) | pickle, records, search, update and copy templates

1. What and why

- A binary file stores data in the same form the computer uses, not as readable characters. Opening it in Notepad shows junk.
- Extension is usually .dat or .bin. It is faster and takes less space than a text file, and it can store a whole list, tuple or dictionary in one go.
- Python handles binary files with the pickle module, so every program starts with: import pickle

Mode	Meaning	If file does not exist	If file exists
rb	read only	error (FileNotFoundError)	opens it
wb	write only	creates it	ERASES all records
ab	append only	creates it	keeps records, adds at end
rb+	read and write	error	opens it, does not erase
wb+	write and read	creates it	ERASES all records
ab+	append and read	creates it	keeps records

The letter b must be there. open('emp.dat','r') on a binary file is wrong and will not work with pickle.

2. The only two pickle functions you need

Function	Syntax	What it does
dump()	`pickle.dump(object, fileobject)`	writes one object (one record) into the binary file
load()	`object = pickle.load(fileobject)`	reads the next object from the binary file

- Each dump() writes one record. To store many records, call dump() many times, or dump one big list once.
- load() reads exactly one record and moves the pointer forward. At the end of the file it raises EOFError, which is how a program knows to stop.

3. A record can be a tuple, a list or a dictionary

```
(1, "ABC", 100, 5000)                # tuple  – cannot be changed
[101, "Aarav", "IT", 50000]          # list   – can be changed, use for update questions
{101: ["Sholay", "Action"]}          # dictionary – key : value
```

If a question asks you to UPDATE a field, the record must be a list, because tuples are immutable.

4. Template A — write / append records

```
import pickle

def append_data():
    with open("emp.dat", "ab") as f:      # ab keeps old records
        eid  = int(input("Enter Employee ID: "))
        name = input("Enter Employee Name: ")
        dept = input("Enter Department: ")
        sal  = float(input("Enter Salary: "))
        rec  = [eid, name, dept, sal]
        pickle.dump(rec, f)
    print("Employee data appended successfully.")

append_data()
```

5. Template B — read every record (learn this by heart)

```
import pickle

def display_all():
    f = open("emp.dat", "rb")
    try:
        while True:
            rec = pickle.load(f)          # one record at a time
            print(rec)
    except EOFError:
```

```
        pass                            # end of file reached
    f.close()

display_all()
```

Every other binary file program is this loop with an if inside it.

6. Template C — search and count with a condition

```
import pickle

def expensiveProducts():
    count = 0
    with open("INVENTORY.DAT", "rb") as f:
        while True:
            try:
                rec = pickle.load(f)      # (PID, PName, Qty, Price)
                if rec[3] > 1000:
                    print("Product ID:", rec[0])
                    count += 1
            except EOFError:
                break
    print("Total expensive products:", count)

expensiveProducts()
```

7. Template D — copy matching records into a new file

```
import pickle

def COPY_REC():
    fin  = open("FLIGHT.DAT", "rb")       # source : read mode
    fout = open("RECORD.DAT", "wb")       # target : write mode
    count = 0
    try:
        while True:
            rec = pickle.load(fin)        # [Fno,FName,Fare,Source,Dest]
            if rec[3] == "DELHI" and rec[4] == "MUMBAI":
                pickle.dump(rec, fout)
                count += 1
    except EOFError:
        pass
    fin.close()
    fout.close()
    return count                          # if the question asks for a count

COPY_REC()
```

8. Template E — update records (two accepted methods)

Method 1 — read all into a list, then rewrite the file (safest, use this)

```
import pickle

def update_data():
    employees = []
    with open("emp.dat", "rb") as f:      # STEP 1 : read everything
        try:
            while True:
                emp = pickle.load(f)
                if emp[2] == "IT":        # STEP 2 : change in the list
                    emp[3] = 200000
                employees.append(emp)
        except EOFError:
            pass
    with open("emp.dat", "wb") as f:      # STEP 3 : write the list back
        for emp in employees:
            pickle.dump(emp, f)
    print("Records updated")

update_data()
```

Method 2 — rb+ with seek() (only when the record size does not change)

```
f = open("emp.dat", "rb+")
while True:
    try:
        pos = f.tell()                    # remember where the record started
        emp = pickle.load(f)
        if emp[2] == "IT":
```

```
            emp[3] = 200000
            f.seek(pos)                  # go back to that record
            pickle.dump(emp, f)          # overwrite it
    except EOFError:
        break
f.close()
```

9. Template F — records stored as a dictionary

```
import pickle

def findType(mtype):
    f = open("CINEMA.DAT", "rb")         # {MNO: [MNAME, MTYPE]}
    try:
        while True:
            rec = pickle.load(f)
            for mno in rec:              # loop over the keys
                mname = rec[mno][0]
                movie_type = rec[mno][1]
                if movie_type == mtype:
                    print("Movie Number :", mno)
                    print("Movie Name   :", mname)
                    print("Movie Type   :", movie_type)
    except EOFError:
        pass
    f.close()

findType("Comedy")
```

10. Deleting a record

There is no delete function. Read every record, skip the one to be removed, write the rest back in wb mode — the same three steps as the update template.

```
import pickle

def delete_rec(eid):
    data = []
    with open("emp.dat", "rb") as f:
        try:
            while True:
                rec = pickle.load(f)
                if rec[0] != eid:        # keep everyone except eid
                    data.append(rec)
        except EOFError:
            pass
    with open("emp.dat", "wb") as f:
        for rec in data:
            pickle.dump(rec, f)
```

11. Quick reference — what the pointer does

Statement	Effect on the file pointer
`pickle.dump(rec, f)`	writes one record and moves the pointer past it
`pickle.load(f)`	reads one record and moves the pointer past it
`f.tell()`	reports the position in bytes, moves nothing
`f.seek(0)`	rewinds to the first record so the file can be read again
`f.seek(0, 2)`	jumps to the end of the file

In binary mode seek() accepts any offset with any value of from — the text-mode restriction does not apply here.

12. Marks are lost here — check every time

- Forgetting import pickle at the top.
- Opening with "wb" when the question says add / append — it erases every existing record.
- Missing the try / except EOFError block — the program crashes at the end of the file instead of stopping neatly.
- Using pickle.load() only once and assuming it read the whole file. One load() = one record.
- Trying to change a field of a tuple record — tuples cannot be modified, the record must be a list.
- Updating while still reading from the same handle in rb mode — read all records first, then rewrite.
- Forgetting to close both files in a copy question, or forgetting to call the function at the end.
- Writing print("Candidate ID: rec[0]") instead of print("Candidate ID:", rec[0]) — the first prints the text, not the value.