Input, output and file handling

6.4 · Problem solving with programming · revision notes

How a program communicates with its user and keeps data between runs: taking input, displaying output, reading from and writing to comma separated value (CSV) text files, and writing code that validates input and authenticates users. These skills are needed in many Paper 2 tasks.

Key facts

- input() always returns a string
- print(a, b) puts a space between items; + only joins strings
- line.strip().split(",") turns a CSV line into a list of strings
- file.close() when you have finished with a file
- Authentication: find the ID, then check the password stored with it
- int(input(...)) for whole numbers; float(input(...)) for decimals
- open(name, "r") read; "w" write a new file (overwrites); "a" append to the end
- file.write(text) needs a string; add "\n" to end the line
- Validation loop: input, then while invalid: message and input again

Notes

Input and output

- `input("Prompt: ")` shows the prompt and returns what the user types as a **string**.
- Convert before doing maths: `age = int(input("Age: "))` or `price = float(input("Price: "))`.
- `print("Total:", total)` outputs the items with a space between them, whatever their data types. `+` only joins strings, so write `"Total: " + str(total)`.
- `round(cost, 2)` rounds a real number to 2 decimal places, which is useful for money.
- Respond appropriately to input: give clear prompts, and a helpful message when input is rejected, e.g. `Enter a whole number from 1 to 10`.

Text files and CSV files

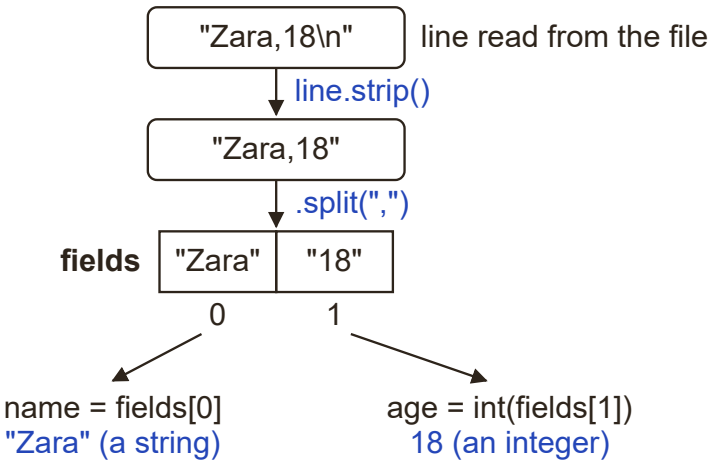
- A **CSV** (comma separated values) file is a text file with one record on each line and the fields separated by commas, e.g. `Ava,17,Blue`.
- `file = open("members.csv", "r")` opens a file for reading. Mode `"w"` creates a new, empty file for writing (any data already in it is lost); mode `"a"` adds to the end of the file.
- Trying to open a file that does not exist for reading causes a runtime error.
- Close every file with `file.close()` when you have finished, so the data is saved and the file is released.

Reading and writing records

- Reading every record in a CSV file:

```
file = open("members.csv", "r")
for line in file:
    fields = line.strip().split(",")
    name = fields[0]
    age = int(fields[1])
    print(name, "is", age)
file.close()
```

- `strip()` removes the newline character from the end of each line; `split(",")` breaks the line into a list of strings at each comma.



How one line of a CSV file becomes separate fields.

- Every field read from a file is a string, so convert numbers with `int()` or `float()`. `file.readline()` reads one line, e.g. to skip a header row.
- Writing a record: `file.write(name + "," + str(age) + "\n")`. `write()` takes one string and does not start a new line by itself, so add `"\n"`.

Validating input and authenticating users

- A validation loop asks again until the input passes every check:

```
mark = input("Mark (0-50): ")
while not mark.isdigit() or int(mark) > 50:
    print("Enter a whole number from 0 to 50")
    mark = input("Mark (0-50): ")
mark = int(mark)
```

- Each check as a condition that is True for invalid input: presence `name == ""`; length `len(pin) != 4`; range `age < 11 or age > 18`; pattern `not pin.isdigit()`.
- `isdigit()` is tested first, so `int()` is only used on text made of digits and cannot cause a runtime error. An empty input fails `isdigit()` too, so this also acts as a presence check. Grade 7+ only
- Authentication by lookup: search the stored IDs (in a list or a file) for the ID entered, then check that the password stored with that ID matches. Output one message after the search, and limit the number of attempts. Grade 8+ only

How to answer each type of question

State the output of a program that reads a file 2 to 4 marks **Grade 5**

1. Take the file one line at a time, as the loop does.
2. Work out what strip() and split(",") produce: a list of strings, with index 0 as the first field.
3. Apply the rest of the loop to that line, then move to the next line.

Example. The file `scores.csv` contains these lines.

```
Zara,18
Tom,11
Priya,20
```

```
01 file = open("scores.csv", "r")
02 for line in file:
03     fields = line.strip().split(",")
04     if int(fields[1]) >= 15:
05         print(fields[0])
06 file.close()
```

(a) State the output of the program. (2)
(b) State the value of `fields` after line 03 runs for the first time. (1)
(c) Explain why `int()` is needed in line 04. (1)

Model answer.

(a) `Zara` (1), then `Priya` on the next line (1); Tom is not output

(b) `["Zara", "18"]`, a list of two strings (1)

(c) Data read from a file is a string, and comparing a string with the integer 15 would cause a runtime error, so it is converted to an integer first (1)

Write a program that writes to a file

4 to 6 marks | **Grade 6**

1. Choose the mode: "a" to add to existing records, "w" to start a new file.
2. Build each record as one string: fields joined with "," and ending in "\n".
3. Close the file after the last write, outside any loop.

Example. Write a program that inputs the name and score of 3 players. Each player must be added to the end of the file `results.csv` on a new line, in the form `name,score`. (5)

Model answer.

```
file = open("results.csv", "a")
for count in range(3):
    name = input("Name: ")
    score = input("Score: ")
    file.write(name + "," + score + "\n")
file.close()
```

Opens `results.csv` in append mode (1); a loop that runs 3 times (1); inputs a name and a score inside the loop (1); writes the name, a comma and the score, followed by a newline (1); closes the file after the loop (1).

Write code to validate input

3 to 5 marks | **Grade 7**

1. Input the value once before the loop.
2. Write a while condition that is True when the input is invalid, joining the checks with or.
3. Inside the loop, output a helpful message and input the value again.
4. Check that the input is made of digits before using int() on it.

Example. A program asks how many tickets the user wants. The number must be a whole number from 1 to 8.
Write code that keeps asking until a valid number is entered, then stores it as an integer in the variable `tickets`. (4)

Model answer.

```
tickets = input("Tickets (1-8): ")
while not tickets.isdigit() or int(tickets) < 1 or int(tickets) > 8:
    print("Please enter a whole number from 1 to 8")
    tickets = input("Tickets (1-8): ")
tickets = int(tickets)
```

A loop that repeats while the input is invalid (1); a correct range check for 1 to 8 (1); checks the input is made of digits before `int()` is used, so the program cannot crash (1); outputs a message and inputs again inside the loop (1).

Write authentication code 5 to 6 marks **Grade 8**

1. Input the ID and the password.
2. Open the stored data and loop through every record.
3. Check that the ID and the password match the same record, and set a Boolean flag if they do.
4. After the loop, output one message based on the flag.

Example. Each line of the file `users.csv` holds a user ID and a password, e.g. `jokafor,Blue7fish`. Write a program that inputs a user ID and a password and outputs `Access granted` if they match a line in the file, or `Access denied` if they do not. (6)

Model answer.

```
userID = input("User ID: ")
password = input("Password: ")
found = False
file = open("users.csv", "r")
for line in file:
    fields = line.strip().split(",")
    if fields[0] == userID and fields[1] == password:
        found = True
file.close()
if found:
    print("Access granted")
else:
    print("Access denied")
```

Inputs the ID and the password (1); opens the file for reading and closes it (1); loops through every line of the file (1); strips and splits each line (1); checks that the ID and the password match the same line (1); outputs one correct message after the search (1).

Don’t lose marks

- Opening a file with "w" when you meant "a", so every existing record is wiped.
- Leaving out "\n" in write(), so all the records end up on one line.
- Comparing file data with numbers without int(): "9" > "10" is True for strings, and "18" >= 15 crashes the program.
- Leaving out strip(), so the last field keeps its newline and "Blue7fish\n" never equals "Blue7fish".
- Printing 'Access denied' inside the search loop, so it is output once for every line that does not match.
- Asking for the input before a validation loop but not again inside it, so an invalid entry loops forever.

More tips

Memory tricks

- File modes: r = read, w = wipe and write, a = add to the end.
- CSV reading recipe: open, for line in file, strip, split, convert, close.
- Everything coming in (from the keyboard or a file) is a string; everything going out to a file must be a string.
- Validation loop shape: ask; while it is bad, complain and ask again.

Exam technique

- Read the file format in the question carefully and note which field is at index 0, 1 and 2.
- Open, loop, strip and split, convert, write or process, close: make every one of these steps visible in your code.
- Use clear prompts and error messages: they show that your program responds appropriately to the user.
- In the onscreen exam, open the file after running your program to check that the data was written as expected.

What each grade needs

What you need to be able to do, from the first marks up to the top grade.

Grade 3 — **Take input and display output**
Use input() with a clear prompt and print() to show the results.

Grade 4 — **Convert input to the correct data type**
Wrap input() in int() or float() before doing any calculation with it.

Grade 5 — **Open a file in the right mode**
Use "r" to read, "w" to write a new file and "a" to add to the end of a file.

Grade 5 — **Read and split a CSV file**
Loop through the lines, strip the newline and split each line at the commas.

Grade 6 — **Write records to a text file**
Build each record as one string with commas and a newline, then close the file.

Grade 7 — **Code a validation loop for user input**
Keep asking until the input passes every check, showing a helpful message each time.

Grade 8 — **Code authentication using an ID lookup**
Search the stored IDs for a match and check the password stored with that ID.

Free notes, Full packs, practice papers and mock exams at chhetriacademy.com. Written for this site; not taken from past papers. Stuck? 1-to-1 tutoring with a free 30-minute trial lesson.