Text Data Extractor

A small Python utility that scans a text file and extracts URLs, email addresses, hashtags, credit-card-like numbers, and USD currency amounts. Extracted values are masked where appropriate (emails and credit-card numbers) and saved as pretty-printed JSON.

Features

- Extracts:
  - Emails (with simple validation and double-dot filter)
  - URLs (HTTP/HTTPS with basic rejection of common malicious patterns)
  - Hashtags (starts with `#`, requires a letter)
  - Credit-card-like 16-digit groups (supports spaces or hyphens) and masks all but the last four digits
  - USD currency amounts (e.g., `$1,234.56` or `$123`)
- Masks email usernames and credit-card numbers for safer storage/display
- No third-party dependencies; uses Python standard library only

Installation

1. Place the script file in a project folder (example: `extractor.py`).
2. Provide an input text file named `sample_input.txt` in the same folder (or change `INPUT_FILE` constant).

Usage

Save the provided code as `extractor.py` (or any filename) and run:

python extractor.py

The script reads `sample_input.txt`, extracts items, writes `sample_output.json`, and prints the JSON result to stdout.

To change input/output filenames, modify the `INPUT_FILE` and `OUTPUT_FILE` constants at the top of the script.

Input example (`sample_input.txt`)

User contact: alice.smith@example.com recovery email bob@example.co.uk. Visit https://example.com/path?query=1 for details. Processed card 4111 1111 1111 1111 Subscription Price $1,299.99 Promotion: #Launch2025

Output example (`sample_output.json`)


```
{
  "emails": [
    "a*****h@example.com",
    "b***@example.co.uk"
  ],
  "urls": [
    "https://example.com/path?query=1"
  ],
  "credit_cards": [
    "**** **** **** 1111"
  ],
  "currency_amounts": [
    "$1,299.99"
  ],
  "hashtags": [
    "#Launch2025"
  ]
}
```


How it works — brief technical notes

- **Emails:** Regex enforces a common local-part pattern and domain labels; results containing `..` in the email are filtered out. Masking keeps only the first and last characters of the username (short usernames handled specially).
- **URLs:** Regex targets `http`/`https` and performs a negative lookahead to reject suspicious or commonly encoded malicious patterns (simple heuristic, not a full security filter).
- **Hashtags:** Pattern requires a `#` followed by optional digits/underscores but must contain at least one ASCII letter.
- **Credit cards:** Detects groups of 16 digits optionally separated by spaces or hyphens. It does not perform Luhn validation; output is masked.
- **Currency:** Matches USD-style `$` followed by digits, comma groups, and optional two-decimal cents.

Security & privacy

- Emails and credit-card-like values are masked by default in output to reduce exposure.
- The URL regex attempts to reject obvious malicious patterns, but it is not a substitute for a dedicated URL validation/sanitization library. Do not rely solely on this script for security-critical URL handling.

- The script does not perform Luhn checks or issuer detection for credit-card numbers and should not be used to store or process real card numbers in production.

Customization

- To change behavior, edit the regex patterns inside `extractor.py` files.
- To support additional currencies or international formats, extend `extract_currency` with new patterns.

Testing

- Create a `sample_input.txt` with representative edge cases (short emails, emails with subdomains, URLs with ports/query strings, hashtags with punctuation, numeric strings that look like cards but are not).
- Run the script and review `sample_output.json`.
- For unit tests, wrap extraction functions and assert on returned lists for specific inputs.