Assigned: 26 August 2021

Homework #1 – Tools and Fundamentals

EE 547: Fall 2021

Due: Saturday, 04 September 2021 at 23:59. Late penalty: 10% per 24-hours before 06 September at 23:59. Submission instructions will follow separately on canvas.

1. Python command line

Write a python script that computes the number of **distinct** anagrams given a string argument. An anagram is a string formed by rearranging the characters of an input string using each of the original characters exactly once. Your script must write the calculated number of distinct case-insensitive anagrams as a single integer to `STDOUT` (standard output). If the input string is empty (*i.e.* zero-length string) your script must write "`empty`" to `STDOUT`.

Your script should accept a string as the first (and only) command line argument. A *valid* string contains only letters (`/[a-zA-Z]/`). If the string contains any other characters your script should write "`invalid`" to `STDERR` (standard error) instead of counting anagrams.

Your script should not produce any output except as described above. Do not use any modules except `sys` in your implementation (*i.e.* do not use any other `import`).

2. Simple python webserver

Use the python `http.server` module to create a simple webserver. Bind the server to all available interfaces and listen on port 8088. Your server must respond to GET requests for the following four paths. Your server should respond to all other GET requests with http status code 404.

- /ping

  Always respond with status code 204 and empty body.

- /shuffle?p=[string]&limit=[integer, default=4, min=0, max=25]

  Read string `p` from the query. Compute the number of distinct anagrams for `p` as in Problem (1). Respond with status code 400 and empty body if the string is invalid or empty. Else generate the first `limit` # anagrams sorted alphabetical ascending ("a" to "z"). Respond with http status code 200 and write the following dictionary to the response body as a JSON string:

```
{
      "p": "[p string]",
      "total": "[number of distinct anagrams]"
```

```
    "page": ["…", "…", …, "…"]    # the alphabetical anagrams, up to
    limit
}
```

- /status

Respond with http status code 200. Write the following dictionary to the response body as a JSON string:

```
{
    "time": "[UTC TIME in ISO-8601]",
    "req": "[number of requests received since server start]"
    "err": "[number of 404 errors, c.f. /secret and invalid path]"
}
```

- /secret

Respond with http status code 200 if the file /tmp/secret.key exists and write the content of the file /tmp/secret.key to the response body. If the file does not exist respond with http status code 404.

3. Create AWS account

Create an AWS account (https://aws.amazon.com/). You may use an AWS account that you already have admin access. Sign-up requires a credit or debit card. Don't start any servers or other services as this may incur costs. You will work primarily in the AWS "Free Tier" for EE 547 and will not incur charges. You will also receive AWS credit codes for required services that are not included in the Free Tier.

Login to your AWS account. Under "Security, Identity, & Compliance" go to "IAM". IAM stands for "Identity and Access Management". From here you can precisely control user permissions as well as restrict or grant access to your AWS resources.

First: Create a new IAM Policy with name *ee547-grader-policy*. The following permissions create a minimal-access user for the grader script. Future assignments may extend this policy.

Service: `STS`
Actions level: `Read:GetCallerIdentity`

Service: `IAM`
Actions: `Read:GetUser, List:ListAttachedUserPolicies`
Resources: Specific, user: (Account: [default], User: `ee547-grader-user`)

Service: `IAM`
Actions: `Read:GetPolicy, Read:GetPolicyVersion`
Resources: Specific, policy: (Account: [default], User: `ee547-grader-policy`)

Second: Create a new IAM User with name *ee547-grader-user*. Select *programmatic access* since the grader uses a script (*i.e.* command line interface, *CLI*) to connect (instead of a web browser). Then attach the existing `ee547-grader-policy` directly. Download (and save) the csv for submission to the autograder -- this is your only opportunity to download this file. This csv file contains the AccessKeyId and SecretAccessKey required to act as the `ee547-grader-user`. If you make an error delete the user and try again.

Submit the downloaded `new_user_credential.csv` to AutoLab.