

מבחן שירותים – שירות הצפנות

עליך למשם שירות FastAPI המספק שירות הצפנה ופענוח שונים. הצלנים פועלם על טקסט באנגלית, ולא רגישות לאותיות גדולות או קטנות (למשל - אפשר להציג תמידאותיות קטנות, כל עוד התשובה עומדת באлогריתם ההצפנה גם אם התקבלה בקשה באותיות גדולות) כמשמעות בונוס יש גם ניהול סטטיסטיות הקשורות לביצועים של השירות.

1) מטרות

- בניית שירות REST באמצעות FastAPI.
- שימוש של ENDPOINTS שמייצגים 2 סוגי צופן (ראו נספח א' לפירוט ודוגמאות לכל צופן):
 - קיסר (Caesar) – הצפנה ופענוח עם פרמטר היסט (offset).
 - גדר (Fence) – הצפנה ופענוח לפי מקומות זוגיים ואי-זוגיים.
- שימוש מידור זמן תגובה (בונוס).
- שימרת סטטיסטיות שימוש בקובצי JSON (בונוס)

2) דרישות מקדיימות

- יש ליצור REPO חדש ולעשוו לו CLONE.
- צרו קובץ README שבו השם המלא שלכם, הכתובת ות"ז - כולם בשורה נפרדת כל אחד.
- צרו סביבה וריטואלית בפייתון (uv / poetry / venv ועוד). הפעילו אותה וודאו שהיא בתוכה.
- התקינו בתוך הסביבה את הספריות: fastapi pydantic בעזרתdık. אתם יכולים גם להתקין את chalice אם תרצו להריץ את השירות בעזרתה. אפשר להתקין ספריות נוספות אם אתם רוצים.

- חובה ליצור קובץ requirements.txt בעזרת freeze –pip. ודאו כי אחרי הרצת הפקודה נוצר קובץ בו אתם רואים את כל dependencies שלuproject.
- הרצת השרת תיעשה בעזרת הפקודה: fastapi dev main.py או:

uvicorn main:app –reload

(3) דרישות כלויות

1. כל התשובות מכל הENDPOINTS יישו בפורמט JSON.
2. יש להוסיף PRINT ותייעוד לכל פעולה משמעותית שמתבצעת, ויש לוודא לפני תום המבחן שלא השארתם הדפסות שאינן כאה.
3. רצוי מאוד לוודא את תקינות הקוד של השרת בעזרת קריאות עם CURL.
בנוסף ג' יש דוגמאות לתקופתיים מוצפנים שאפשר להיעזר בהם כדי לוודא שתהיליך הפיענוח עובד.

(4) מבנהuproject (מומלץ בלבד)

חובה לממש את בסיס השרת בקובץ הראשי [main.py](#). אפשר אף מומלץ לחלק לראוטרים (אם אתם יודעים איך) ופונקציות עזר.

(5) קובץ README (חובה)

יש ליצור קובץ בשם README המכיל שלוש שורות בלבד:

שם מלא בעברית

כיתה

תעודת זהות

(6) הגדרת ה-Endpoints (ראו הגדרת צפנים בנספח א')

basic endpoints:

GET /test

Response: { "msg": "hi from test"}

תיאור: נקודת קצה בסיסית לצורך בדיקות.

GET /test/:name

Response: { "msg": "saved user"}

תיאור: חילוץ השם מתוך PATH PARAMS ושמירתו בתוך קובץ בשם txt.
.names.txt אין לדחוס טקסט קיימ - רק להוציא.

Caesar cipher endpoint:

POST /caesar

Body: { "text": string, "offset": int, "mode": "encrypt"/"decrypt" }

Response: { "encrypted_text": "..." } או { "decrypted_text": "..." }

תיאור:

מצפין או מפענח טקסט לפי צופן קיסר עם היסט (offset) מוגדר.

Fence cipher endpoints:

GET /fence/encrypt?text=<טקסט>

Response:

{ "encrypted_text": "..." }

תיאור - מחלץ את הטקסט מהURL ומצפין אותו לפי צופן הגדר

POST /fence/decrypt

Body: { "text": "string" }

Response: { "decrypted": "..." }

תיאור - מקבל את הטקסט מהגוף של הבקשה ומפענח אותו לפי צופן גדר.

בונוס: שמירת נתוני ביצועים

לצד ה-`Endpoints`, השירות נדרש לתעד נתונים על כל בקשה שמתאפשרת ומטופלת, לצורך מעקב אחר ביצועי המערכת. המידע נשמר בשני קבצים בתוך תיקיית `:/data`:

endpoints_data.json .1

קובץ זה שומר מידע מצטבר לכל אנדפונט. הפורמט הוא רשימה של מיליון, כאשר כל מיליון מייצג אנדפונט יחיד/ב מיליון נשמר השם של ENDPOINT והמתודה, וסטטיסטיקה: `total_requests_received` - כמות הבקשות שהתקבלו לאותו ENDPOINT. `avg_handling_time` - זמן ממוצע מרגע קבלת הבקשה ועד שליחת התשובה.

מבנה קובץ לדוגמה:

```
[  
 {  
   "url": "/atbash",  
   "method": "POST",  
   "stats": {  
     "total_requests_received": 12,  
     "avg_handling_time": 10.4  
   }  
 }  
 ]
```

- בעת קבלת בקשה: יש לעדכן את `.total_requests_received`

- בעת שליחת תגובה: יש ליחס ולעדכן את `avg_handling_time`.
- בעת שליחת תגובה, אחרי עדכון שום, יש לעדכן את `summary.json`.

summary.json .2

קובץ זה מרכז את הנתונים מכל האנדפויינטים, ומציג ערכי שיא ומינימום לפי סוג מדידה:

```
{
  "highest_requests": {
    "name": "atbash POST",
    "number": 17
  },
  "lowest_requests": {
    "name": "fence GET",
    "number": 3
  },
  "highest_handeling_time": {
    "name": "caesar POST",
    "number": 21.7
  },
  "lowest": {
    "name": "atbash POST",
    "number": 7.9
  }
}
```

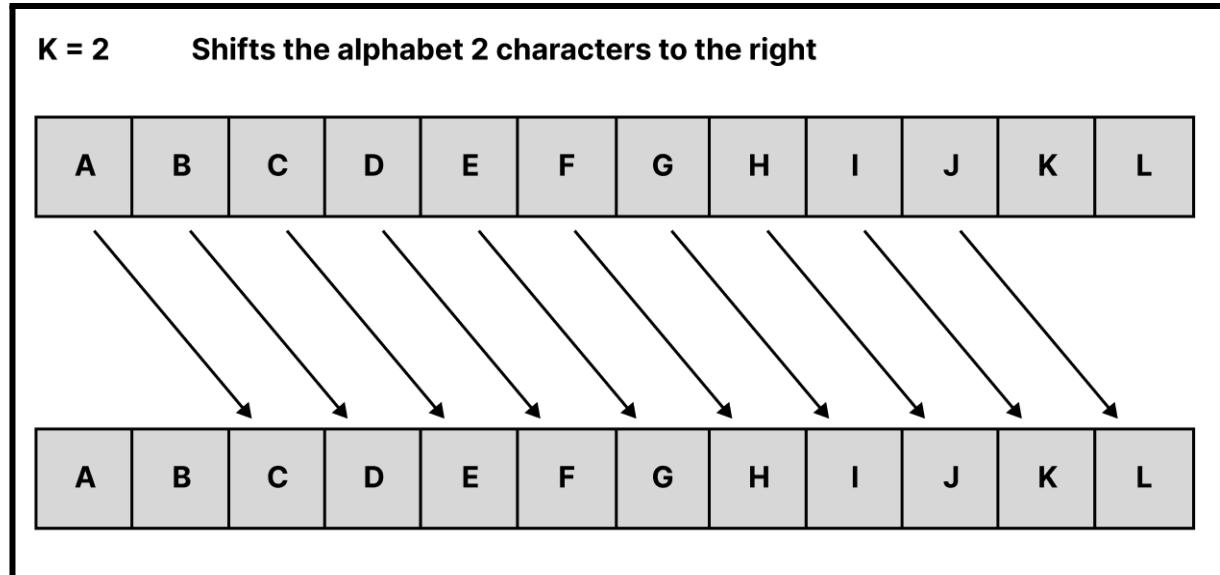
יש להוסיף ENDPOINT שמקבל את הנתונים הכלליים, ונתונים לפי URL.
METHOD

נספח א – הסבר על הצענים.

צופן קיסר

צופן קיסר הוא צופן החלפה מבוסס **היסט קבוע באלפבית**. כל אות מוזמת במספר צעדים קבוע (offset) קדימה. כאשר מגיעים לסוף האלפבית, ממשיכים מההתחלתה.

לדוגמה, בהיסט 2:



אם מגיעים לסוף האלפבית, חוזרים להתחלה. לדוגמה, אם צריך להצפין את Z עם offset של 2: חוזרים לא וממשיכים LB.

הפענוח זהה להתהליך ההצפנה - רק בכיוון ההפוך.

צופן גדר

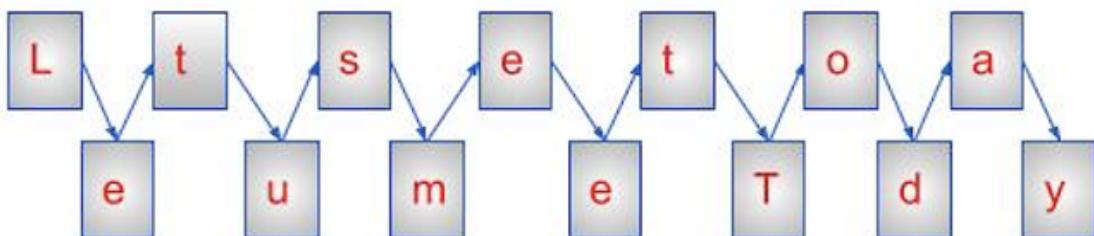
צופן גדר (או Rail Fence Cipher) הוא צופן טרנספוזיציה (החלפת סדר האותיות). הטקסט נכתב לשירוגין בשתי "مسילות" – אחת לאותיות במקומות אי-זוגיים והשנייה לזוגיים, ואז מוצמדות שתי המסילות זו לצד זו.

בגרסה שלנו משתמשים ב-2 מסילות בלבד. אפשר להשתמש ביותר.

תהליך הצפנה

1. מסירים רווחים.
2. מפרידים את האותיות למיקומים אי-זוגיים וזוגיים. האות הראשונה באינדקס 0 נחשבת זוגית.
3. מצמידים: תחילת הזוגיות ולאחר מכן האי זוגיות.

לדוגמה: הטקסט: let us meet today:



הtekst המוצפן: Itsetoaeumetdy

תהליך פענוח (אין צורך לשחזר את הרווחים):
תחשבו.

נספח ב – ארכיטטקטורה שעון (Clock Arithmetic / Modulo)

הסבר כללי

ארכיטקטורת שעון היא שיטה מתמטית שבה המספרים "נעימים במעגל", וכש/cgiעים לסופי — חוזרים להתחלה. השיטה מבוססת על פעולה **מודולו (mod)**, %, המגדירה את שארית החלוקה של מספר אחד במספר אחר.

בדומה לשעון עם הספרות 11-0, שבו $9 + 4 \neq 13$ אלא:



כך גם באלפבית — אחרי האות האחרונה חוזרים לראשונה. העיקרון זהה הוא הבסיס לפעלת צופן קיסר.

דוגמה — על שעון עם 12 ספרות.

נניח שהשעה עכשו 9:00,
ואנו מוסיפים 5 שעות קדימה.
נחשב:

$$(9 + 5) \bmod 12 = 2$$

$$(9 + 5) \% 12 = 2 \# \text{ in python}$$

כלומר, התוצאה היא השעה 2:00.

דוגמה — באלפבית האנגלי

נניח שה-A היא 0, B היא 1, ... Z היא 25.
אם נרצה להציג את האות W (אינדקס 22) ב-5 צעדים קדימה:

$$(22 + 5) \bmod 26 = 1$$

$$(22 + 5) \% 26 = 1 \# \text{ in python}$$

1 מייצג את האות B.
כלומר, W עם offset של 5 הופכת ל-B.

נספח ג' - דוגמאות לקוד מקודד ומפוענה, בכל צופן

הטקסטים הבאים הם לשימושכם, כחלק מקריאות CURL לשרת - על מנת לוודא שהHTTP ENDPOINT באמת עובד.

צופן קיוסר:

תוצאה	הסטה	טקסט
olssv dvysk	7	hello world
welcome kodkod	16	mubsecu aetaet

צופן גדרה:

תוצאה	טקסט
hteeihr	hi there
welcomekodkod	wloekdecmkdo

נספח ד' - פקודות CURL מוכנות לבדיקה:

 **בדיקה תקינות שרת 1.**

```
curl -i -X GET "http://127.0.0.1:8000/test"
```

Expected output:

```
{  
  "msg": "hi from test"  
}
```

 **2. נשמרות שם ל-** names.txt

```
curl -i -X GET "http://127.0.0.1:8000/test/Avi"
```

Expected output:

```
{  
  "msg": "saved user"  
}
```

(תתוסף השורה "Avi" בקובץ names.txt)

 **3. Caesar encrypt — hello world, offset 7**

```
curl -i -X POST "http://127.0.0.1:8000/caesar" -H "Content-Type: application/json" -d "{\"text\":\"hello world\",\"offset\":7,\"mode\":\"encrypt\"}"
```

Expected output:

```
{  
  "encrypted_text": "olssv dvysk"  
}
```

4. Caesar decrypt — olssv dvysk, offset 7

```
curl -i -X POST "http://127.0.0.1:8000/caesar" -H "Content-Type: application/json" -d "{\"text\": \"olssv dvysk\", \"offset\": 7, \"mode\": \"decrypt\"}"
```

Expected output:

```
{  
  "decrypted_text": "hello world"  
}
```

5. Caesar encrypt — welcome kodkod, offset 16

```
curl -i -X POST "http://127.0.0.1:8000/caesar" -H "Content-Type: application/json" -d "{\"text\": \"welcome kodkod\", \"offset\": 16, \"mode\": \"encrypt\"}"
```

Expected output:

```
{
```

```
"encrypted_text": "mubsecu aetaet"  
}
```

6. Caesar decrypt — mubsecu aetaet, offset 16

```
curl -i -X POST "http://127.0.0.1:8000/caesar" -H "Content-Type: application/json" -d "{\"text\":\"mubsecu aetaet\",\"offset\":16,\"mode\":\"decrypt\"}"
```

Expected output:

```
{  
  "decrypted_text": "welcome kodkod"  
}
```

7. Fence encrypt — hi there

```
curl -i -X GET "http://127.0.0.1:8000/fence/encrypt?text=hi%20there"
```

Expected output:

```
{  
  "encrypted_text": "hteeihr"  
}
```

8. Fence decrypt — hteeihr

```
curl -i -X POST "http://127.0.0.1:8000/fence/decrypt" -H "Content-Type: application/json" -d "{\"text\":\"hteeihr\"}"
```

Expected output:

```
{  
    "decrypted": "hithere"  
}
```

9. Fence encrypt — welcome kodkod

```
curl -i -X GET  
"http://127.0.0.1:8000/fence/encrypt?text=welcome%20kodkod"
```

Expected output:

```
{  
    "encrypted_text": "wloekdecmkdo"  
}
```

10. Fence decrypt — wloekdecmkdo

```
curl -i -X POST "http://127.0.0.1:8000/fence/decrypt" -H "Content-Type: application/json" -d "{\"text\":\"wloekdecmkdo\"}"
```

Expected output:

```
{  
    "decrypted": "welcomekodkod"  
}
```