

# Python

TomerBu

## **נושאים:**

מחלקות וירשה

ריבוי ארגומנטים לפונקציה עם kwargs

עבודה עם ספריות חיצונית:

requests, beautiful soup

# מחלקות:

tabniot ليיצרת אובייקטים:

תכונות: name, email וכו'

פועלות: מתודות כגון: בדוק-סימא, שם מלא, החלף אימייל

```
class Person:  
    # properties/attributes:  
    def __init__(self, email, first_name, last_name, username):  
        self.email = email  
        self.first_name = first_name  
        self.last_name = last_name  
        self.username = username  
  
    # methods/actions/behaviors:  
    def full_name(self):  
        return f"{self.first_name} {self.last_name}"  
  
    def say_hi(self):  
        return f"Hi, I'm {self.full_name()}"  
  
p = Person("Harry@gmail.com", "Harry", "Potter", "hpotter")  
p2 = Person("hermione@gmail.com", "Hermione", "Granger", "hgranger")  
  
print(p.full_name())  
print(p.say_hi())  
  
print(p2.full_name())  
print(p2.say_hi())
```

# מחלקות:

הדפסה של אובייקט בקונסולה:

```
print(my_object)
```

ינטראקטיבית: ידפיס את הכתובת של האובייקט בזיכרון ושם המחלקה

```
class Person:  
    # properties/attributes:  
    def __init__(self, email, first_name, last_name, username):  
        self.email = email  
        self.first_name = first_name  
        self.last_name = last_name  
        self.username = username  
  
    # methods/actions/behaviors:  
    def __str__(self):  
        return f"Person(first_name: {self.first_name}, last_name:{self.last_name}, email: {self.email}, username: {self.username})"  
  
    def full_name(self):  
        return f"{self.first_name} {self.last_name}"  
  
    def say_hi(self):  
        return f"Hi, I'm {self.full_name()}"  
  
p = Person("Harry@gmail.com", "Harry", "Potter", "hpotter")  
print(p)
```

מחזירה מחרוזת שתודפס בקונסולה

הMETHOD `__str__`

# מיני תרגיל:



צרו מחלוקת עבור קלף משחק:

תכונות: צורה (מעוין, לב, תלתן, עלה)

suit

תכונות: דרגת הקלף: (2, 3, 4, 5...J, Q, K, A)

rank

פעולות:

`_str_` return "rank:A, suit: ♣"

הציג קלף:  
`show_card` return "A of ♣"

צרו מחלוקת לחפירת קלפים:

תכונות: רשימה של קלפים או סט של קלפים

אם נגידר מתודה בשם

deal card

פעולות shuffle ערבות החפירה

`_say_hi_`

המשמעות היא להציג שזה מיועד לשימוש פרטי של המחלוקת.

# מיני תרגיל:

ציוו הטיפוס למטרות תיעוד  
(טיפוס ברירת מחדל any)

```
class Card:  
    # suit, rank:  
    def __init__(self, rank:str, suit:str):  
        self.rank = rank  
        self.suit = suit  
  
    def __str__(self):  
        return f"suit: {self.suit}, rank: {self.rank}"  
  
    def show(self):  
        return f"{self.rank} of {self.suit}"  
  
card = Card("A", "Hearts")  
print(card)  
print(card.show())
```

# מיני תרגיל:

צרו מחלקה לחפישת קלפים:

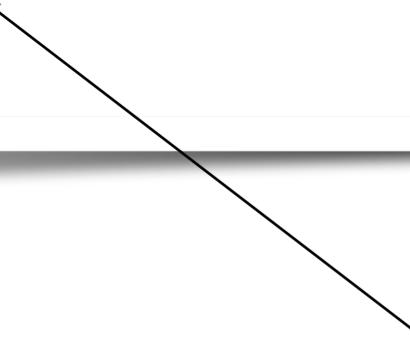
תכונות: רשימה של קלפים

הנחיה:

בבנייה של המחלקה יש למלא 52 קלפים בעזרת לולאות:

```
class Deck:  
    # props: list of cards:  
    def __init__(self):  
        self.cards = []  
        ranks = ["2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A"]  
        suits = ["Hearts", "Diamonds", "Clubs", "Spades"]  
  
        # loop through suits and ranks to populate the list of cards
```

your code here



# מיני תרגיל:

```
from random import shuffle

class Deck:
    # props: list of cards
    def __init__(self):
        self.cards = []
        ranks = ["2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A"]
        suits = ["Hearts", "Diamonds", "Clubs", "Spades"]

        # loop through suits and ranks to populate the list of cards
        for rank in ranks:
            for suit in suits:
                self.cards.append(Card(rank, suit))

    def __str__(self):
        return ", ".join([str(card) for card in self.cards])

    def shuffle(self):
        shuffle(self.cards)

deck = Deck()
deck.shuffle()
print(deck)
```

בנית החפיסה:

הציג של האובייקטים:

פונקציונליות לאובייקט:

# ירושה:

מחלקה עבורה בעל חיים:

Animal

תכונות

פעולות

מחלקה עבורה חתול:

Cat

בנוסף לתכונות והפעולות של Animal

אפשר להוסיף תכונות ופעולות נוספות

אפשר "לדרוס" פעולות מהמחלקה Animal

# ירושה:

```
class Animal:  
    # properties/attributes:  
    def __init__(self, name):  
        self.name = name  
  
    def make_sound(self):  
        return "Animal sound"
```

```
class Cat(Animal):  
    def __init__(self, name, breed):  
        super().__init__(name)  
        self.breed = breed  
  
    def make_sound(self):  
        return "Meow"
```

ירושה:

רשימת התכונות:  
כל התכונות של מחלקת האב  
ובנוסף - תכונות של חתול

העברה של תכונות לבניי של מחלקת האב:  
מבצע את `__init__` של מחלקת האב

דרישה של מתודות

# ירושה:

```
class Animal:  
    # properties/attributes:  
    def __init__(self, name):  
        self.name = name  
  
    def make_sound(self):  
        return "Animal sound"  
  
class Cat(Animal):  
    def __init__(self, name, breed):  
        # call the parent class constructor:  
        super().__init__(name)  
        self.breed = breed  
  
    def make_sound(self):  
        return "Meow"  
  
cat1 = Cat("Fluffy", "Siamese")  
print(cat1.make_sound())
```

מבנה למחקות שירשות:

يוצר מבנה אחד:

לכל בעל חיים יש:

name, make\_sound()

לכל מחלקת שירשת מ-Animal

יהי name

וגם make\_sound()

# ירושה מרובה:

```
class A:  
    pass  
  
class B:  
    pass  
  
class C(A, B):  
    pass
```

ניתן לרשת מספר מחלקות:

# ירושא מרובה:

```
# base class:  
class Animal:  
    def __init__(self, name):  
        self.name = name  
  
    def make_sound(self):  
        return "Animal sound"  
  
# behavior class:  
class Driver:  
    def drive(self):  
        return "Driving"  
  
# inheritance:  
class Cow(Animal, Driver):  
    def make_sound(self):  
        return "Moo"
```

ניתן לרשת מספר מחלקות:

נהוג לרשת מחלוקת אחת שהגינוי לרשת ממנה:

לדוגמה: חתול->בעל חיים

לדוגמה->div ירוש מHTMLElement

אפשר להוסיף מחלוקת נוספת לירושא:  
מחלקות שמוסיפות פונקציונליות:

רק מתודות

# ירושא מרובה:

צרו מחלקה בסיס `Duck`

לכל  
name  
`make_sound()`

צרו 3 מחלקות שיורשות מבסיס:

`RedHeadDuck`  
`MollardDuck`  
`RubberDuck`



דרישה חדשה מהבוסית:

ברווז מולרד  
וברווז רדヘד

להוסיף מתודה של `fly`  
המתודה תדפיס "flying"

# ירושה מרובה:

```
class Duck:  
    def __init__(self, name):  
        self.name = name  
    def make_sound(self):  
        return "Duck Sound"  
  
class FlyBehavior:  
    def fly(self):  
        return "Flying"  
  
class MollardDuck(Duck, FlyBehavior):  
    def make_sound(self):  
        return "Mollard Quack"  
  
class RedheadDuck(Duck, FlyBehavior):  
    def make_sound(self):  
        return "Quack"  
  
class RubberDuck(Duck):  
    def make_sound(self):  
        return "Squeak"
```

# ריבוי ארגומנטים עם kwargs = key word arguments

```
def draw_text(**kwargs):
    """
    kwargs: color, size, text, font, bold, italic, underline
    """
    print(kwargs["color"])
    print(kwargs["size"])
    print(kwargs["text"])

draw_text(text="Hello World", font="Arial",underline=False)
```

```
turtle.write(arg, move=False, align='left', font=('Arial', 8, 'normal'))¶
```

# ריבוי ארגומנטים עם kwargs = key word arguments

```
from typing import TypedDict, Unpack
```

```
class DrawArgs(TypedDict):  
    color: str  
    size: int  
    text: str  
    font: str  
    bold: bool  
    italic: bool  
    underline: bool
```

```
def draw_text(**kwargs: Unpack[DrawArgs]):  
    ....  
    kwargs: color, size, text, font, bold, italic, underline  
    ....  
    print(kwargs["color"])  
    print(kwargs["size"])  
    print(kwargs["text"])
```

```
draw_text(color="red", size=12, text="Hello World", font="Arial", bold=True, italic=False, underline=False)  
draw_text(text="Hello World", font="Arial",underline=False)
```

1

2

3

השלה  
אוטומטית  
**\*\*kwargs**

# התקנה של ספריה גלובלית למחשב:

הרעיוון: נתקן את הספריה פעם אחת במחשב  
ולא יצטברו לנו תיקיות גדולות עם המונע ספריות

```
pip install requests
```

בפרויקט גדול:  
যোগীর মন

ומתakinim בתוכו את הספריות  
וכך אפשר להעלות את הפרויקט יחד עם הספריות.

# הספריה: requests



## Requests

http for humans



52,475

Requests is an elegant and simple HTTP library for Python, built for human beings.

## Useful Links

[Quickstart](#)

[Advanced Usage](#)

[API Reference](#)

[Release History](#)

[Contributors Guide](#)

[Recommended Packages and Extensions](#)

## Requests: HTTP for Humans™

Release v2.32.3. ([Installation](#))

downloads/month 621M license Apache-2.0 wheel yes python 3.8 | 3.9 | 3.10 | 3.11 | 3.12

**Requests** is an elegant and simple HTTP library for Python, built for human beings.

### Behold, the power of Requests:

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
'{"type": "User", ...}'
>>> r.json()
{'private_gists': 419, 'total_private_repos': 77, ...}
```

See [similar code, sans Requests](#).

**Requests** allows you to send HTTP/1.1 requests extremely easily. There's no need to manually add query strings to your URLs, or to form-encode your POST data. Keep-alive and HTTP connection pooling are 100% automatic, thanks to [urllib3](#).

# הספרייה requests:

```
import requests

url = 'https://www.google.com'

response = requests.get(url)

print(response.text)
```

# הספריה: requests

<https://opentdb.com/api.php?amount=10>

{

"response\_code": 0,

"results": [

{↔},

אנחנו מקבלים מילון:

{↔},

ויש בו מפתחות: "results"

{↔},

הטיפוס של results הוא list  
ואפשר לזרע עלייו בלולאה

{↔},

{

"type": "multiple", ←

מפתח בשם type

"difficulty": "easy",

"category": "Entertainmer

"question": "What is the

← מפתח בשם question

"correct\_answer": "Kyoko

"incorrect\_answers": [

"Yoko Ono",

"Kyary Pamyu Pamyu",

# הספרייה requests:

<https://opentdb.com/api.php?amount=10>

```
url = 'https://opentdb.com/api.php?amount=10'

response = requests.get(url)

json_data = response.json()

results_list = json_data['results']

for result in results_list:
    # result has the following keys:
    # 'category', 'type', 'difficulty', 'question', 'correct_answer', 'incorrect_answers'
    print(result['question'])
```



# הספריה: requests

הטקסט מכיל HTML Entities

## Some Useful HTML Character Entities

Result	Description	Name	Number	
	non-breaking space	&nbsp;	&#160;	<a href="#">Try it »</a>
<	less than	&lt;	&#60;	<a href="#">Try it »</a>
>	greater than	&gt;	&#62;	<a href="#">Try it »</a>
&	ampersand	&amp;	&#38;	<a href="#">Try it »</a>

```
import html
```

# הספריה requests:

## הטקסט מכיל HTML Entities

```
import requests
import html
url = 'https://opentdb.com/api.php?amount=10'

response = requests.get(url)

json_data = response.json()

results_list = json_data['results']

for result in results_list:
    # result has the following keys:
    # 'category', 'type', 'difficulty', 'question', 'correct_answer',
    'incorrect_answers'
    question = result['question']
    question = html.unescape(question) #translate html entities to characters
    print(question)
```

# משחק טריויה מהAPI

```
import requests
import html
from random import shuffle

url = 'https://opentdb.com/api.php?amount=10'
response = requests.get(url)

json_data = response.json()

results_list = json_data['results']

for result in results_list:
    question = result['question']
    question = html.unescape(question)
    correct_answer = result['correct_answer']
    all_answers = result['incorrect_answers']

    # add correct answer to all answers
    # so we have all answers in one list
    all_answers.append(correct_answer)

    shuffle(all_answers)
    print('Question:', question)
    print(all_answers)

    # input of a number -> from 1 to 4
    # -1 converts the int to 0 to 3
    user_choice = int(input('enter your answer: 1 to 4:')) - 1

    if correct_answer == all_answers[user_choice]:
        print('Correct!')
    else:
        print('Wrong! The correct answer is:', correct_answer)
```

<https://opentdb.com/api.php?amount=10&category=9>

# מיני תרגיל:

<https://raw.githubusercontent.com/toedter/movies-demo/refs/heads/master/backend/src/main/resources/static/movie-data/movies-250.json>

```
{  
  "date": "2022-04-03",  
  "movies": [  
    {"Title": "The Shawshank Redemption", "Year": "1994", "Rated": "R", "Released": "14 Oct 1994", "Runtime": "142 min", "Genre": "Drama", "Director": "Frank Darabont", "Writer": "Stephen King, Frank Darabont", "Actors": "Tim Robbins, M
```

במקום results יש לróż על movies

עבור כל סרט (בלולאה)

יש להדפיס

**Title**

**Year**

**Actors**

מומלץ להוסיף ("Hit enter to continue")  
בגוף הלולאה.

# פונCTIONALITY :HTML

```
pip install beautifulsoup4
```

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

```
from bs4 import BeautifulSoup
import requests

url = 'https://www.w3schools.com/html/html_entities.asp'

response = requests.get(url)

if response.status_code != 200:
    print('Failed to fetch page')
    print(response)
    exit()

html = response.text

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

print(soup.title.text) # title tag from the document
print(soup.p.text.strip()) # text inside the first p tag
```

# פונCTIONALITY :HTML

```
from bs4 import BeautifulSoup
import requests

url = 'https://www.w3schools.com/html/html_entities.asp'

response = requests.get(url)

if response.status_code != 200:
    print('Failed to fetch page')
    print(response)
    exit()

html = response.text

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

paragraphs = soup.find_all('p')
links = soup.find_all('a')

for link in links:
    # text as usual
    print(link.text.strip())
    # href attribute link['href']
    href = link.get('href')
    href = f'https://www.w3schools.com{href}'
    print(href)

input("hit enter to continue")

for paragraph in paragraphs:
    print(paragraph.text.strip())
    input("hit enter to continue")
```

כל התגיות מאותה סוג

attribute

מיני תרגיל:

נסו למצוא את כל התגיות מסוג img  
הדפיסו את הsrc של כל הimages

(אפשר גם להוריד את התמונות ולשמור אותן)

# פונCTION :HTML

```
from bs4 import BeautifulSoup
import requests

url = 'https://www.w3schools.com/html/html_entities.asp'

response = requests.get(url)

if response.status_code != 200:
    print('Failed to fetch page')
    print(response)
    exit()

html = response.text

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

# find all images/links/paragraphs

all_images = soup.find_all('img')

for image in all_images:
    print(image.get('alt'))
    src = image.get('src') #/signup/lynxlogo.svg
    src = f'https://www.w3schools.com{src}'
    print(src)
```

# כתיבה של טקסט לקובץ:

```
# open a file for writing
f = open("123.txt", 'w')

# write to the file
f.write("Hello files")

# close the file
f.close()
```

פתיחה של קובץ לכתיבה טקסט

ביצוע כתיבה של טקסט

סגירת הקובץ בסיום

# הMETHOD split מפרזת מערך

```
str = '3,4,5,6,7,8'  
lst = str.split(',')  
print(lst) # ['3', '4', '5', '6', '7', '8']
```

```
str = "welcome to the hotel california"  
lst = str.split(' ')  
print(lst) # ['welcome', 'to', 'the', 'hotel', 'california']
```

```
str = "https://www.w3schools.com/html/html_entities.asp"  
lst = str.split('/')  
print(lst) # ['https:', '', 'www.w3schools.com', 'html', 'html_entities.asp']  
  
print(lst[-1]) # html_entities.asp
```

# הורדה של התמונות:

```
from bs4 import BeautifulSoup
import requests

url = 'https://www.w3schools.com/html/html_entities.asp'
response = requests.get(url)

if response.status_code != 200:
    print('Failed to fetch page')
    print(response)
    exit()

html = response.text
soup = BeautifulSoup(html, 'html.parser')

# find all images/links/paragraphs
all_images = soup.find_all('img')

for image in all_images:
    src = image.get('src') #/signup/lynxlogo.svg

    file_name = src.split('/')[-1]
    src = f'https://www.w3schools.com{src}'

    res = requests.get(src)

    # res.text (text), res.json (json), res.content (binary)

    binary_data = res.content
    file = open(file_name, 'wb')
    file.write(binary_data)
    file.close()
```

# הגדלה לשרת לגביה הדפסנו שבו אנו משתמשים

```
import requests
from bs4 import BeautifulSoup

url = 'https://www.imdb.com/chart/top/'

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'
}
response = requests.get(url, headers=headers)

html = response.text
soup = BeautifulSoup(html, 'html.parser')

all_titles = soup.select('.ipc-title__text')

for title in all_titles:
    print(title.text)
```

הMETHOD select מאפשר לנו לחפש בעמוד  
תוך שימוש בסלקטוריים של CSS

# מיני תרגיל

הציגו עבור כל סרט:

כותרת  
שנת יציאה  
דירוג  
כתובות של התמונה  
 קישור של הסרט

הקלאס של התמונה:

ipc-image

# פתרונות לתרגיל:

```
import requests
from bs4 import BeautifulSoup

url = 'https://www.imdb.com/chart/top/'

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'
}
response = requests.get(url, headers=headers)

html = response.text
soup = BeautifulSoup(html, 'html.parser')

all_list_items = soup.select('.ipc-metadata-list li')

for movie_li in all_list_items:
    title = movie_li.select_one('.ipc-title__text').text

    all_metadata = movie_li.select('.cli-title-metadata-item')

    year = all_metadata[0].text
    duration = all_metadata[1].text
    rating = all_metadata[2].text

    image = movie_li.select_one('.ipc-image').get('src')

print(title, year, duration, rating, image)
```