

תרגיל 3: תכנות מונחה עצמים

תאריך פרסום: 23.06.2024

תאריך הגשה: 07.07.2024 בשעה 23:59

מתרגל אחראי: דור שמאי

דגשים:

1. מומלץ לקרוא את כל התרגיל לפני המימוש. אנו ממליצים להבין היטב את ארכיטקטורת המערכת ולתכנן אותה על דף נייר (או אייפד) לפני תחילת כתיבת הקוד.
2. בניגוד לתרגילים קודמים, **בתרגיל זה אין להניח שהקלט תקין!** עבור כל שיטה מוגדרות הדרישות לגבי הקלט. הנחות על קלטים מסוימים יצוינו באופן מפורש. במידה והקלט אינו תקין עליכם לזרוק חריגה מתאימה.
3. **בתרגיל לא תמיד יצוין באופן מפורש איך עליכם לממש את הפתרון. לדוגמה, לא תמיד יצוין מתי להשתמש בשיטות אב, להגדיר שדה פרטי, לבצע shallow copy או deep copy וכדומה. עליכם יהיה להחליט על פרטי המימוש תוך עמידה בדרישות התרגיל ובעקרונות שלמדתם בקורס (לדוגמה, להימנע משכפול קוד).**
4. השאלות יבדקו באופן אוטומטי. הפלט שעליכם להחזיר בכל תרגיל צריך להיות בדיוק כפי שנדרש. כמו כן, באופן אקראי יבדקו גם עבודות באופן ידני.
5. כאשר תבוצע בדיקה ידנית, תתבצע גם בדיקת Readability - שימו לב שאתם משתמשים בשמות משתנים אינפורמטיביים וכותבים הערות בכל סעיף.
6. בדיקה עצמית: כדי לוודא את נכונותן ואת עמידותן של הפונקציות לקלטים שונים, בכל שאלה הריצו אותן עם מגוון קלטים: אלה שמופיעים בדוגמאות וקלטים נוספים עליהם חשבתם. וודאו כי הפלט נכון. הבדיקה תתבצע על מגוון דוגמאות ולא בהכרח אלה שיינתנו פה.
7. נא לקרוא את כל העבודה לפני שאתם מתחילים לכתוב את הקוד (זו הפעם השניה של הסעיף הזה ולא במקרה..).

הקדמה:

ברוכים הבאים לעולם הפוקימונים! שמי פרופסור Python ואני אדריך אתכם בתחילת מסעכם בעולם זה. אני כרגע עובד על מערכת חדשה שתשנה את פני העולם, ומרחיבה את מערכת הפוקדקס המוכרת. הפוקדקס היא מערכת אחזור מידע של פוקימונים, סוגי פוקימונים וביצוע מניפולציות על הסוגים השונים. כעת נרחיבה גם לאפשר ניהול ומעקב אחר קרבות בין מאמני פוקימונים. על מנת לסיים לבנות את גרסת ה- POC (proof of concept) של הפוקדקס המורחב, אני זקוק לעזרתכם. השלימו את גרסה זו של הפוקדקס המורחב בשבילי - אמנם לא תקבלו אחוזים מהרווחים, אך מי מכם שיעשה עבודה מוצלחת יזכה בציון טוב וכנראה גם הכנה מעולה למבחן. בהצלחה!







למוטיבציה: <https://www.youtube.com/watch?v=rg6CiPl6h2g>
 כל הפרטים אודות הפוקימונים לקוחים מהאתר: <https://pokemondb.net/pokedex/national>

מערכת לניהול קרבות:

במטלה זו נממש קרב בין שני מאמני פוקימונים. הפוקימונים השייכים לכל מאמן מיוצגים על ידי רשימה של מופעים של פוקימון (**Pokemon**) -
 כל פוקימון שייך לאחד משלושת הסוגים הבאים:

- **Fire** - שלושה פוקימונים מסוג אש
- **Water** - שלושה פוקימונים מסוג מים
- **Electric** - פוקימון אחד מסוג חשמל

מבנה המחלקות מתואר באיור.

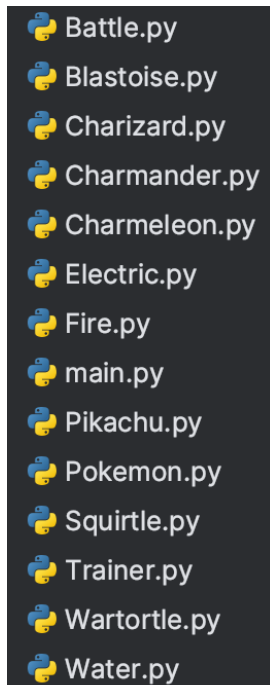
| Pokemon | | | |
|---|---|---|---|
| Fire | | | Electric |
| Charmander | Charmeleon | Charizard | Pikachu |
|  |  |  |  |
| Water | | | |
| Squirtle | Wartortle | Blastoise | |
|  |  |  | |

מבנה ההגשה:

שימו לב: עליכם להגיש בעבודה זו 13 קבצים. שמות הקבצים צריכים להיות תואמים בדיוק להנחיות. עליכם לשמור ב-Pycharm את כל הקבצים באותה תיקייה על מנת שתוכלו לייבא (import) אותם באמצעות שמם (ללא נתיב לקובץ).

לדוגמה: `from Pikachu import Pikachu`

לפניכם רשימת הקבצים שיש להגיש (שימו לב, אין צורך להגיש את קובץ ה-main):



- Battle.py
- Blastoise.py
- Charizard.py
- Charmander.py
- Charmeleon.py
- Electric.py
- Fire.py
- main.py
- Pikachu.py
- Pokemon.py
- Squirtle.py
- Trainer.py
- Wartortle.py
- Water.py

:Pokemon

מחלקה אבסטרקטית אשר מייצגת פוקימון. מחלקת פוקימון (בגרסה פשוטה זו) מכילה את השדות הבאים בלבד, אליהם אי אפשר לגשת ולא ניתן לשנות מחוץ למחלקת Pokemon (כאשר כתוב "בין הערכים" הכוונה כולל):

- **-name** - משתנה המייצג את שמו של הפוקימון מטיפוס מחרוזת.
- **-catch_rate** - משתנה מסוג int המייצג את היכולת לתפוס את הפוקימון, ערכו של המשתנה catch_rate חייב להיות בין הערכים 40-45.

ממשו את מחלקת Pokemon:

```
def __init__(self, name, catch_rate):
```

בנאי המאתחל את שדות המחלקה. במידה ואחד השדות לא תקין יש לזרוק אחת מהחריגות הבאות (לפי השגיאה): ValueError או TypeError.

```
def get_name(self):
```

השיטה מחזירה את שמו של הפוקימון.

```
def get_catch_rate(self):
```

השיטה מחזירה את יכולת תפיסת הפוקימון.

Abstract methods:

הוסיפו את השיטות האבסטרקטיות הבאות:

```
__repr__, absorb, attack, can_fight, get_damage, get_defense_power,
get_effective_against_me, get_effective_against_others, get_hit_points,
get_pokemon_type, level_up
```

דוגמאות הרצה:

```
from Pokemon import Pokemon
>>> pokemon = Pokemon("pikachu", 43)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't instantiate abstract class Pokemon with abstract
methods __repr__, absorb, attack, can_fight, get_damage,
get_defense_power, get_effective_against_me,
```

get_effective_against_others, get_hit_points, get_pokemon_type,
level_up

שימו לב כי לא ניתן ליצור מופע של מחלקה אבסטרקטית.

: Fire, Electric, Water

מחלקות אבסטרקטיות אשר מייצגות את סוגי הפוקימון השונים. כל אחת ממחלקות אלה מתוארת על ידי השדות הנוספים הבאים אליהם אי אפשר לגשת ולא ניתן לשנות מחוץ למחלקה (כאשר כתוב "בין הערכים" הכוונה כולל):

- **pokemon_type** - משתנה המייצג את סוגו של הפוקימון מטיפוס מחרוזת. תיזרק שגיאה מסוג ValueError במידה ו:
 - סוג הפוקימון Fire לא מאותחל להיות "fire".
 - סוג הפוקימון Electric לא מאותחל להיות "electric".
 - סוג הפוקימון Water לא מאותחל להיות "water".
- **effective_against_me** - רשימה אשר מייצגת את סוגי הפוקימון אשר אפקטיביים בקרבות נגד סוג הפוקימון הספציפי.
 - כאשר נוצר מופע חדש של Fire, ערכו של המשתנה effective_against_me מאותחל להיות ["water"].
 - כאשר נוצר מופע חדש של Electric, ערכו של המשתנה effective_against_me מאותחל להיות רשימה ריקה [], כיוון שפוקימונים מסוג אש ומים אינם אפקטיביים נגד סוג חשמל.
 - כאשר נוצר מופע חדש של Water, ערכו של המשתנה effective_against_me מאותחל להיות ["electric"].
- **effective_against_others** - רשימה אשר מייצגת את סוגי הפוקימון אשר הפוקימון מהסוג הספציפי אפקטיבי בקרבות נגדם.
 - כאשר נוצר מופע חדש של Fire, ערכו של המשתנה effective_against_others מאותחל להיות רשימה ריקה [], כיוון שאינו אפקטיבי נגד פוקימונים מסוג מים וחשמל.
 - כאשר נוצר מופע חדש של Electric, ערכו של המשתנה effective_against_others מאותחל להיות ["water"].
 - כאשר נוצר מופע חדש של Water, ערכו של המשתנה effective_against_others מאותחל להיות ["fire"].

ממשו את מחלקות Fire, Electric, Water:

```
def __init__(self, name, catch_rate, pokemon_type):
```

בנאי המאתחל את שדות המחלקה.

```
def get_pokemon_type(self):
```

השיטה מחזירה את הסוג של הפוקימון.

```
def get_effective_against_me(self):
```

השיטה מחזירה את רשימת הפוקימונים אשר אפקטיביים בקרבות נגד הפוקימון מהסוג הספציפי.

```
def get_effective_against_others(self):
```

השיטה מחזירה את רשימת הפוקימונים אשר הסוג הספציפי אפקטיבי בקרבות נגדם.

Abstract methods:

הוסיפו את השיטות האבסטרקטיות הבאות:

```
__repr__, absorb, attack, can_fight, get_damage, get_defense_power, get_hit_points, level_up
```

דוגמאות הרצה:

```
>>> from Fire import Fire
>>> fire = Fire("charmander", 43, "fire")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't instantiate abstract class Fire with abstract methods
__repr__, absorb, attack, can_fight, get_damage, get_defense_power,
get_hit_points, level_up
```

שימו לב כי לא ניתן ליצור מופע של מחלקה אבסטרקטית.

: Charmander

מחלקה אשר מייצגת את הפוקימון צ'ארמנדר. צ'ארמנדר מתואר על ידי השדות הנוספים הבאים אליהם אי אפשר לגשת ולא ניתן לשנות מחוץ למחלקת Charmander (כאשר כתוב "בין הערכים" הכוונה כולל):

- **level** - משתנה מסוג int המייצג את הרמה של הפוקימון, ערכו של המשתנה level חייב להיות בין הערכים 1-15.
- **hit_points** - משתנה מסוג int המייצג את כמות החיים שיש לפוקימון. כאשר הפוקימון מותקף מספר זה קטן (בהתאם לעוצמת המכה שקיבל). כאשר ערך משתנה זה נמוך מסף מסוים,

הפוקימון מתעלף ולא יוכל להמשיך בקרב (לא לדאוג הוא יחלים). כאשר נוצר מופע חדש של Charmander, ערכו של המשתנה hit_points חייב להיות בין הערכים 39-57.

- **attack_power** - משתנה מסוג int המייצג את כוח התקיפה של הפוקימון. כאשר נוצר מופע חדש של Charmander, כוח התקיפה שלו חייב להיות בין הערכים 52-63.

- **defense_power** - משתנה מסוג int המייצג את כוח ההגנה של הפוקימון. כאשר נוצר מופע חדש של Charmander, ערכו של המשתנה defense_power חייב להיות בין הערכים 43-57.

ממשו את מחלקת Charmander:

```
def __init__(self, name, catch_rate, pokemon_type, level, hit_points, attack_power,
defense_power):
```

בנאי המאתחל את שדות המחלקה.
במידה ואחד השדות לא תקין יש לזרוק אחת מהחריגות הבאות (לפי השגיאה): TypeError או ValueError.

```
def __repr__(self):
```

השיטה תחזיר מחרוזת המייצגת את הפוקימון Charmander בפורמט:
The Charmander **name** of level **level** with **hit_points** HP.

כאשר שחור מסמן את התווים הקבועים ואדום את ערכי השדות המתאימים.
לדוגמה:

The Charmander charmeey of level 13 with 46 HP.

```
def get_hit_points(self):
```

השיטה מחזירה את כמות החיים של הפוקימון Charmander.

```
def get_defense_power(self):
```

השיטה מחזירה את כוח ההגנה של הפוקימון Charmander.

```
def can_fight(self):
```

השיטה מחזירה True אם ל Charmander יש כמות חיים חיובית המספיקה כדי להילחם ואחרת False.
כמות זו צריכה להיות יותר מעשירית מכמות החיים שהייתה לו כשנוצר (מעוגלת כלפי מטה).

```
def get_damage(self, other):
```

השיטה מחזירה את עוצמת הנזק אשר הפוקימון Charmander יכול להסב לפוקימון אחר המיוצג על ידי המשתנה other. שימו לב כי other יכול להיות כל אחד מפוקימונים השונים. ניתן להניח קלט תקין למתודה.

עוצמת הנזק מחושבת על פי הנוסחה הבאה, תוך שימוש בכוח התקיפה והרמה של Charmander, בכוח ההגנה של הפוקימון האחר (שימו לב כי התוצאה מתעגלת כלפי מטה). הערך של eff יהיה שווה ל-2 במידה ו Charmander אפקטיבי נגד סוג הפוקימון האחר, ואחרת ל-0.5.

$$damage = \left\lfloor \left(\frac{2 \times Level}{5} + 2 \right) \times \frac{attack_power}{defense_power} \times eff \right\rfloor$$

`def attack(self, other):`

מטרת השיטה הינה הכאה של פוקימון אחר המיוצג על ידי המשתנה other. שימו לב כי other יכול להיות כל אחד מפוקימונים השונים. ניתן להניח קלט תקין למתודה. פעולת ההכאה מאופיינת על ידי השלבים הבאים:

1. היצור המכה בודק אם גם הוא וגם היצור המוכה יכולים להילחם. אם אחד היצורים לא יכול להילחם, פעולת ההכאה מפסיקה.
2. היצור המכה מפחית מכמות החיים שלו עשירית מכמות החיים ההתחלתית שהייתה לו כשנוצר (מעוגל כלפי מטה).
3. Charmander מפעיל את פעולת הספיגה של הפוקימון המוכה, תוך שימוש בעוצמה שמחושבת על ידי המתודה `get_damage` (ראו להלן פירוט על פעולת הספיגה).

`def absorb(self, damage):`

בפעולה זו הפוקימון הסופג מאבד damage יחידות חיים. יש לעדכן את כמות החיים המדויקת גם אם לאחר המכה מתקבלת כמות שלילית.

`def level_up(self, level_gain):`

השיטה מעדכנת את הרמה של Charmander ומוסיפה לו level_gain רמות. ערכו של המשתנה level_gain חייב להיות בין הערכים 1-16, אחרת רמתו של הפוקימון לא תשתנה. במידה והעדכון מביא את הרמה של Charmander להיות גדולה מ-15, יש לקרוא למתודה `evolve` המתוארת להלן. השיטה מחזירה None במידה ולא היה צורך לקרוא למתודה `evolve`, ואחרת תחזיר את המופע של הפוקימון Charmeleon שנוצר.

`def evolve(self):`

השיטה מחזירה מופע חדש של הפוקימון Charmeleon המהווה את ההתפתחות של Charmander. את המופע החדש יש לאתחל עם הרמה המעודכנת, תוספת של 19 ל `hit_points` הנוכחי של Charmander, תוספת של 12 ל `attack_power` של Charmander, ותוספת של 15 ל `defense_power` של Charmander. במידה וכמות ה `hit_points` לאחר שנוסף לה 19 לא נמצאת בטווח שמאפשר יצירת מופע של Charmeleon, יש להגדירה להיות הערך המינימלי של `hit_points` הדרוש ליצירת מופע זה.

דוגמאות הרצה:

```
>>> from Charmander import Charmander
>>> charmy = Charmander("charmy", 41, "fire", 6, 50, 54, 51)
>>> lst = charmy.get_effective_against_me()
```



```
>>> lst
['water']
>>> lst[0] = "electric"
>>> lst
['electric']
>>> charmy.get_effective_against_me()
['water']
>>> charmy
The Charmander charmy of level 6 with 50 HP
```

: Charmeleon

מחלקה אשר מייצגת את הפוקימון צ'ארמיליון. צ'ארמיליון מתואר על ידי השדות הנוספים הבאים אליהם אי אפשר לגשת ולא ניתן לשנות מחוץ למחלקת Charmeleon (כאשר כתוב "בין הערכים" הכוונה כולל):

- **level** - משתנה מסוג int המייצג את הרמה של הפוקימון, ערכו של המשתנה level חייב להיות בין הערכים 16-31.
- **hit_points** - משתנה מסוג int המייצג את כמות החיים שיש לפוקימון. כאשר הפוקימון מותקף מספר זה קטן (בהתאם לעוצמת המכה שקיבל). כאשר ערך משתנה זה נמוך מסף מסוים, הפוקימון מתעלף ולא יוכל להמשיך בקרב (לא לדאוג הוא יחלים). כאשר נוצר מופע חדש של Charmeleon, ערכו של המשתנה hit_points חייב להיות בין הערכים 58-77.
- **attack_power** - משתנה מסוג int המייצג את כוח התקיפה של הפוקימון. כאשר נוצר מופע חדש של Charmeleon, כוח התקיפה שלו חייב להיות בין הערכים 64-83.
- **defense_power** - משתנה מסוג int המייצג את כוח ההגנה של הפוקימון. כאשר נוצר מופע חדש של Charmeleon, ערכו של המשתנה defense_power חייב להיות בין הערכים 58-77.

ממשו את מחלקת Charmeleon:

ממשו את השיטות הבאות עם אותן ההוראות כפי שניתנו עבור Charmander:

`__init__, get_hit_points, get_defense_power, can_fight, attack, absorb`

עבור שאר השיטות ההוראות אינן זהות ועל כן ניתן פירוט נוסף (עבור Charmeleon):

`def __repr__(self):`

השיטה תחזיר מחרוזת המייצגת את הפוקימון Charmeleon בפורמט:

The Charmeleon **name** of level **level** with **hit_points** HP.

כאשר שחור מסמן את התווים הקבועים ואדום את ערכי השדות המתאימים.

לדוגמה:

The Charmeleon charmel of level 19 with 63 HP.

`def get_damage (self, other):`

שיטה זאת מתבצעת באופן זהה לאופן בו מתבצעת עבור Charmander (ושאר הפוקימונים מסוג אש) מלבד עוצמת הנזק שהכאה של Charmeleon עושה, המחושבת באופן הבא:

$$damage = \left\lfloor \left(\frac{2 \times Level}{5} + 2 \right) \times \frac{attack_power}{defense_power} \times eff \right\rfloor + 2$$

`def level_up(self, level_gain):`

השיטה מעדכנת את הרמה של Charmeleon ומוסיפה לו level_gain רמות. ערכו של המשתנה level_gain חייב להיות בין הערכים 1-16, אחרת רמתו של הפוקימון לא תשתנה. במידה והעדכון מביא את הרמה של Charmeleon להיות גדולה מ-31, יש לקרוא למתודה evolve המתוארת להלן. השיטה מחזירה None במידה ולא היה צורך לקרוא למתודה evolve, ואחרת תחזיר את המופע של הפוקימון Charizard שנוצר.

`def evolve(self):`

השיטה מחזירה מופע חדש של הפוקימון Charizard המייצג את ההתפתחות של Charmeleon. את המופע החדש יש לאתחל עם הרמה המעודכנת, תוספת של 20 ל hit_points הנוכחי של Charmeleon, תוספת של 20 ל attack_power של Charmeleon, ותוספת של 20 ל defense_power של Charmeleon. במידה וכמות ה hit_points לאחר שנוסף לה 20 לא נמצאת בטווח שמאפשר יצירת מופע של Charizard, יש להגדירה להיות הערך המינימלי של hit_points הדרוש ליצירת מופע זה.

דוגמאות הרצה:

```
>>> from Charmander import Charmander
>>> charmy = Charmander("charmy", 41, "fire", 6, 50, 54, 51)
>>> charmilious = Charmander("charmilious", 42, "fire", 13, 45, 55, 47)
>>> charmy
The Charmander charmy of level 6 with 50 HP
>>> charmilious
The Charmander charmilious of level 13 with 45 HP
>>> charmilious.get_damage(charmy)
3
>>> charmilious.attack(charmy)
>>> charmy
The Charmander charmy of level 6 with 47 HP
>>> charmilious
The Charmander charmilious of level 13 with 41 HP
>>> evolved_charmy = charmy.level_up(5)
>>> if evolved_charmy != None: charmy = evolved_charmy
```

```

...
>>> del evolved_charmy
>>> print(charmy)
The Charmander charmy of level 11 with 47 HP
>>> evolved_charmy = charmy.level_up(5)
>>> if evolved_charmy != None: charmy = evolved_charmy
...
>>> del evolved_charmy
>>> charmy
The Charmeleon charmy of level 16 with 66 HP

```

: Charizard

מחלקה אשר מייצגת את הפוקימון צ'אריזארד. צ'אריזארד מתואר על ידי השדות הנוספים הבאים אליהם אי אפשר לגשת ולא ניתן לשנות מחוץ למחלקת Charizard (כאשר כתוב "בין הערכים" הכוונה כולל):

- **level** - משתנה מסוג int המייצג את הרמה של הפוקימון, ערכו של המשתנה level חייב להיות בין הערכים 32-50.
- **hit_points** - משתנה מסוג int המייצג את כמות החיים שיש לפוקימון. כאשר הפוקימון מותקף מספר זה קטן (בהתאם לעוצמת המכה שקיבל). כאשר ערך משתנה זה נמוך מסף מסוים, הפוקימון מתעלף ולא יוכל להמשיך בקרב (לא לדאוג הוא יחלים). כאשר נוצר מופע חדש של Charizard, ערכו של המשתנה hit_points חייב להיות בין הערכים 78-99.
- **attack_power** - משתנה מסוג int המייצג את כוח התקיפה של הפוקימון. כאשר נוצר מופע חדש של Charizard, כוח התקיפה שלו חייב להיות בין הערכים 84-99.
- **defense_power** - משתנה מסוג int המייצג את כוח ההגנה של הפוקימון. כאשר נוצר מופע חדש של Charizard, ערכו של המשתנה defense_power חייב להיות בין הערכים 78-99.

ממשו את מחלקת Charizard:

ממשו את השיטות הבאות עם אותן ההוראות כפי שניתנו עבור Charmander:

`__init__, get_hit_points, get_defense_power, can_fight, attack, absorb`

עבור שאר השיטות ההוראות אינן זהות ועל כן ניתן פירוט נוסף (עבור Charizard):

`def __repr__(self):`

השיטה תחזיר מחרוזת המייצגת את הפוקימון Charizard בפורמט:

The Charizard **name** of level **level** with **hit_points** HP.

כאשר שחור מסמן את התווים הקבועים ואדום את ערכי השדות המתאימים.
לדוגמה:

The Charizard charz of level 35 with 89 HP.

`def get_damage(self, other):`

שיטה זאת מתבצעת באופן זהה לאופן בו מתבצעת עבור Charmander (ושאר הפוקימונים מסוג אש) מלבד עוצמת הנזק שהכאה של Charmeleon עושה, המחושבת באופן הבא:

$$damage = \left\lfloor \left(\frac{2 \times Level}{5} + 2 \right) \times \frac{attack_power}{defense_power} \times eff \right\rfloor + 4$$

`def level_up(self, level_gain):`

השיטה מעדכנת את הרמה של Charizard ומוסיפה לו level_gain רמות. ערכו של המשתנה level_gain חייב להיות חיובי (אחרת רמתו של Charizard לא תשתנה), ורמתו של Charizard לא יכולה להיות גדולה מ-50. על כן כל תוספת מעבר לכך תשאיר את רמתו של Charizard 50.

שימו לב: לא קיימת מתודה evolve שכן אין התפתחות נוספת לפוקימון זה.

:Pikachu

מחלקה אשר מייצגת את הפוקימון פיקאצ'ו. פיקאצ'ו מתואר על ידי השדות הנוספים הבאים אליהם אי אפשר לגשת ולא ניתן לשנות מחוץ למחלקת Pikachu (כאשר כתוב "בין הערכים" הכוונה כולל):

- **level** - משתנה מסוג int המייצג את הרמה של הפוקימון, ערכו של המשתנה level חייב להיות בין הערכים 1-32.
- **hit_points** - משתנה מסוג int המייצג את כמות החיים שיש לפוקימון. כאשר הפוקימון מותקף מספר זה קטן (בהתאם לעוצמת המכה שקיבל). כאשר ערך משתנה זה נמוך מסף מסוים, הפוקימון מתעלף ולא יוכל להמשיך בקרב (לא לדאוג הוא יחלים). כאשר נוצר מופע חדש של Pikachu, ערכו של המשתנה hit_points חייב להיות בין הערכים 35-99.
- **attack_power** - משתנה מסוג int המייצג את כוח התקיפה של הפוקימון. כאשר נוצר מופע חדש של Pikachu, כוח התקיפה שלו חייב להיות בין הערכים 55-99.
- **defense_power** - משתנה מסוג int המייצג את כוח ההגנה של הפוקימון. כאשר נוצר מופע חדש של Pikachu, ערכו של המשתנה defense_power חייב להיות בין הערכים 40-99.
- **friendship** - משתנה מסוג int המייצג את כוח החברות של Pikachu. כאשר נוצר מופע חדש של Pikachu, ערכו של המשתנה friendship חייב להיות בין הערכים 1-5.

ממשו את מחלקת Pikachu:

ממשו את השיטות הבאות עם אותן ההוראות (לאו דווקא אותו המימוש) כפי שניתנו עבור Charmander:

`get_hit_points, get_defense_power, can_fight, attack, absorb`

עבור שאר השיטות ההוראות אינן זהות ועל כן ניתן פירוט נוסף (עבור Pikachu):

`def __init__(self, name, catch_rate, pokemon_type, level, hit_points, attack_power, defense_power, friendship):`

בנאי המאתחל את שדות המחלקה.
במידה ואחד השדות לא תקין יש לזרוק אחת מהחריגות הבאות (לפי השגיאה): `TypeError` או `ValueError`.
שימו לב כי ההוראות זהות אך יש שוני בחתימה של המתודה.

`def __repr__(self):`

השיטה תחזיר מחרוזת המייצגת את הפוקימון Pikachu בפורמט:

The Pikachu `name` of level `level` with `hit_points` HP.

כאשר שחור מסמן את התווים הקבועים ואדום את ערכי השדות המתאימים.
לדוגמה:

The Pikachu pika of level 40 with 46 HP.

`def get_damage(self, other):`

שיטה זאת מתבצעת באופן זהה לאופן בו מתבצעת עבור Charmander (ושאר הפוקימונים מסוג אש) מלבד עוצמת הנזק שהכאה של Pikachu עושה, המחושבת באופן הבא:

$$damage = \left\lceil \left(\frac{2 \times Level}{5} + 2 \right) \times \frac{attack_power}{defense_power} \times eff \right\rceil + friendship$$

`def level_up(self, level_gain):`

השיטה מעדכנת את הרמה של Pikachu ומוסיפה לו `level_gain` רמות. ערכו של המשתנה `level_gain` חייב להיות חיובי (אחרת רמתו של Pikachu לא תשתנה), ורמתו של Pikachu לא יכולה להיות גדולה מ-50. על כן כל תוספת מעבר לכך תשאיר את רמתו של Pikachu 50.

שימו לב: לא קיימת מתודה `evolve` שכן בתרגיל זה לא מוגדרת התפתחות נוספת לפיקאצ'ו.

: Squirrel

מחלקה אשר מייצגת את הפוקימון סקוירטל. סקוירטל מתואר על ידי השדות הנוספים הבאים אליהם אי (כאשר כתוב "בין הערכים" הכוונה כולל): Squirrel אפשר לגשת ולא ניתן לשנות מחוץ למחלקת

- **level** - משתנה מסוג int המייצג את הרמה של הפוקימון, ערכו של המשתנה level חייב להיות בין הערכים 1-15.
- **hit_points** - משתנה מסוג int המייצג את כמות החיים שיש לפוקימון. כאשר הפוקימון מותקף מספר זה קטן (בהתאם לעוצמת המכה שקיבל). כאשר ערך משתנה זה נמוך מסף מסוים, הפוקימון מתעלף ולא יוכל להמשיך בקרב (לא לדאוג הוא יחלים). כאשר נוצר מופע חדש של Squirrel, ערכו של המשתנה hit_points חייב להיות בין הערכים 44-58.
- **attack_power** - משתנה מסוג int המייצג את כוח התקיפה של הפוקימון. כאשר נוצר מופע חדש של Squirrel, כוח התקיפה שלו חייב להיות בין הערכים 48-62.
- **defense_power** - משתנה מסוג int המייצג את כוח ההגנה של הפוקימון. כאשר נוצר מופע חדש של Squirrel, ערכו של המשתנה defense_power חייב להיות בין הערכים 65-79.

ממשו את מחלקת Squirrel:

ממשו את השיטות הבאות עם אותן ההוראות כפי שניתנו עבור Charmander:

`__init__, get_hit_points, get_defense_power, can_fight, attack, get_damage, absorb`

עבור שאר השיטות ההוראות אינן זהות ועל כן ניתן פירוט נוסף (עבור Squirrel):

`def __repr__(self):`

השיטה תחזיר מחרוזת המייצגת את הפוקימון Squirrel בפורמט:

The Squirrel **name** of level **level** with **hit_points** HP.

כאשר שחור מסמן את התווים הקבועים ואדום את ערכי השדות המתאימים.
לדוגמה:

The Squirrel squ of level 12 with 46 HP.

`def level_up(self, level_gain):`

השיטה מעדכנת את הרמה של Squirrel ומוסיפה לו level_gain רמות. ערכו של המשתנה level_gain חייב להיות בין הערכים 1-16, אחרת רמתו של הפוקימון לא תשתנה. במידה והעדכון מביא את הרמה של Squirrel להיות גדולה מ-15, יש לקרוא למתודה evolve המתוארת להלן.
השיטה מחזירה None במידה ולא היה צורך לקרוא למתודה evolve, ואחרת תחזיר את המופע של הפוקימון Wartortle שנוצר.

`def evolve(self):`

השיטה מחזירה מופע חדש של הפוקימון Wartortle המהווה את ההתפתחות של Squirtle. את המופע החדש יש לאתחל עם הרמה המעודכנת, תוספת של 15 ל hit_points הנוכחי של Squirtle, תוספת של 15 ל attack_power של Squirtle, ותוספת של 15 ל defense_power של Squirtle. במידה וכמות ה hit_points לאחר שנוסף לה 15 לא נמצאת בטווח שמאפשר יצירת מופע של Wartortle, יש להגדירה להיות הערך המינימלי של hit_points הדרוש ליצירת מופע זה.

: Wartortle

מחלקה אשר מייצגת את הפוקימון וורטורטל. וורטורטל מתואר על ידי השדות הבאים אליהם אי אפשר לגשת ולא ניתן לשנות מחוץ למחלקת Wartortle (כאשר כתוב "בין הערכים" הכוונה כולל):

- **level** - משתנה מסוג int המייצג את הרמה של הפוקימון, ערכו של המשתנה level חייב להיות בין הערכים 16-31.
- **hit_points** - משתנה מסוג int המייצג את כמות החיים שיש לפוקימון. כאשר הפוקימון מותקף מספר זה קטן (בהתאם לעוצמת המכה שקיבל). כאשר ערך משתנה זה נמוך מסף מסוים, הפוקימון מתעלף ולא יוכל להמשיך בקרב (לא לדאוג הוא יחלים). כאשר נוצר מופע חדש של Wartortle, ערכו של המשתנה hit_points חייב להיות בין הערכים 59-78.
- **attack_power** - משתנה מסוג int המייצג את כוח התקיפה של הפוקימון. כאשר נוצר מופע חדש של Wartortle, כוח התקיפה שלו חייב להיות בין הערכים 63-82.
- **defense_power** - משתנה מסוג int המייצג את כוח ההגנה של הפוקימון. כאשר נוצר מופע חדש של Wartortle, ערכו של המשתנה defense_power חייב להיות בין הערכים 80-99.

ממשו את מחלקת Wartortle:

ממשו את השיטות הבאות עם אותן ההוראות כפי שניתנו עבור Charmander:

`__init__, get_hit_points, get_defense_power, can_fight, attack, absorb`

עבור שאר השיטות ההוראות אינן זהות ועל כן ניתן פירוט נוסף (עבור Wartortle):

`def __repr__(self):`

השיטה תחזיר מחרוזת המייצגת את הפוקימון Wartortle בפורמט:

The Wartortle **name** of level **level** with **hit_points** HP.

כאשר שחור מסמן את התווים הקבועים ואדום את ערכי השדות המתאימים.
לדוגמה:

The Wartortle warty of level 23 with 60 HP.

`def get_damage(self, other):`

שיטה זאת מתבצעת באופן זהה לאופן בו מתבצעת עבור Charmander (ושאר הפוקימונים) מלבד עוצמת הנזק שהכאה של Wartortle עושה, המחושבת באופן הבא:

$$damage = \left\lfloor \left(\frac{2 \times Level}{5} + 2 \right) \times \frac{attack_power}{defense_power} \times eff \right\rfloor - 1$$

`def level_up(self, level_gain):`

השיטה מעדכנת את הרמה של Wartortle ומוסיפה לו level_gain רמות. ערכו של המשתנה level_gain חייב להיות בין הערכים 1-16, אחרת רמתו של הפוקימון לא תשתנה. במידה והעדכון מביא את הרמה של Wartortle להיות גדולה מ-31, יש לקרוא למתודה evolve המתוארת להלן. השיטה מחזירה None במידה ולא היה צורך לקרוא למתודה evolve, ואחרת תחזיר את המופע של הפוקימון Blastoise שנוצר.

`def evolve(self):`

השיטה מחזירה מופע חדש של הפוקימון Blastoise המייצג את ההתפתחות של Wartortle. את המופע החדש יש לאתחל עם הרמה המעודכנת, תוספת של 20 ל hit_points הנוכחי של Wartortle, תוספת של 20 ל attack_power של Wartortle, ותוספת של 20 ל defense_power של Wartortle. במידה וכמות ה hit_points לאחר שנוסף לה 15 לא נמצאת בטווח שמאפשר יצירת מופע של Blastoise, יש להגדירה להיות הערך המינימלי של hit_points הדרוש ליצירת מופע זה.

: Blastoise

מחלקה אשר מייצגת את הפוקימון בלסטויז. בלסטויז מתואר על ידי השדות הבאים אליהם אי אפשר לגשת ולא ניתן לשנות מחוץ למחלקת Blastoise (כאשר כתוב "בין הערכים" הכוונה כולל):

- **level** - משתנה מסוג int המייצג את הרמה של הפוקימון, ערכו של המשתנה level חייב להיות בין הערכים 32-50.
- **hit_points** - משתנה מסוג int המייצג את כמות החיים שיש לפוקימון. כאשר הפוקימון מותקף מספר זה קטן (בהתאם לעוצמת המכה שקיבל). כאשר ערך משתנה זה נמוך מסף מסוים, הפוקימון מתעלף ולא יוכל להמשיך בקרב (לא לדאוג הוא יחלים). כאשר נוצר מופע חדש של Blastoise, ערכו של המשתנה hit_points חייב להיות בין הערכים 80-99.
- **attack_power** - משתנה מסוג int המייצג את כוח התקיפה של הפוקימון. כאשר נוצר מופע חדש של Blastoise, כוח התקיפה שלו חייב להיות בין הערכים 83-99.

- **defense_power** - משתנה מסוג int המייצג את כוח ההגנה של הפוקימון. כאשר נוצר מופע חדש של Blastoise, ערכו של המשתנה defense_power חייב להיות בין הערכים 100-115.

ממשו את מחלקת Blastoise:

ממשו את השיטות הבאות עם אותן ההוראות כפי שניתנו עבור Charmander:

`__init__, get_hit_points, get_defense_power, can_fight, attack, absorb`

עבור שאר השיטות ההוראות אינן זהות ועל כן ניתן פירוט נוסף (עבור Blastoise):

`def __repr__(self):`

השיטה תחזיר מחרוזת המייצגת את הפוקימון Blastoise בפורמט:

The Blastoise **name** of level **level** with **hit_points** HP.

כאשר שחור מסמן את התווים הקבועים ואדום את ערכי השדות המתאימים.
לדוגמה:

The Blastoise blast of level 36 with 99 HP.

`def get_damage(self, other):`

שיטה זאת מתבצעת באופן זהה לאופן בו מתבצעת עבור Charmander (ושאר הפוקימונים) מלבד עוצמת הנזק שהכאה של Wartortle עושה, המחושבת באופן הבא:

$$damage = \left\lceil \left(\frac{2 \times Level}{5} + 2 \right) \times \frac{attack_power}{defense_power} \times eff \right\rceil - 2$$

`def level_up(self, level_gain):`

השיטה מעדכנת את הרמה של Blastoise ומוסיפה לו level_gain רמות. ערכו של המשתנה level_gain חייב להיות חיובי (אחרת רמתו של Blastoise לא תשתנה), ורמתו של Blastoise לא יכולה להיות גדולה מ-50. על כן כל תוספת מעבר לכך תשאיר את רמתו של Blastoise 50.

שימו לב: לא קיימת מתודה evolve שכן אין התפתחות נוספת לפוקימון זה.

מאמן (Trainer):

מחלקה אשר מייצגת מאמן. מאמן מתואר על ידי השדות הבאים אליהם אי אפשר לגשת ולא ניתן לשנות למחלקת מאמן (כאשר כתוב "בין הערכים" הכוונה כולל):

- name - שם מטיפוס מחרוזת ולא ריקה.
- age - גיל מטיפוס int, כאשר נוצר מופע חדש של Trainer, גילו חייב להיות בין הערכים 120-16.
- exp_modifier - משתנה מסוג float המייצג את מקדם הניסיון של המאמן. כאשר נוצר מופע חדש של Trainer, מקדם הניסיון שלו חייב להיות בין הערכים 12.5-1.5.
- pokemons_lst - רשימה של הפוקימונים השייכים למאמן. כאשר נוצר מופע חדש של Trainer הרשימה תאותחל להיות ריקה.

ממשו את מחלקת Trainer:

```
def __init__(self, name, age, exp_modifier, pokemons_lst=None):
```

בנאי המאתחל את שדות המחלקה.
 במידה ואחד השדות קיבל ערך לא תקין יש לזרוק חריגה מתאימה.
 שימו לב לעקרון חשוב שלמדנו בכיתה – יש להימנע ממתן ערך דיפולטיבי שהוא mutable (ניתן לשינוי) בפונקציות/מתודות.

```
def __len__(self):
```

ממשו את אופרטור האורך של מאמן שיחזיר את כמות הפוקימונים שיש למאמן.

```
def __repr__(self):
```

השיטה תחזיר מחרוזת המייצגת את המאמן בפורמט (ארבעה רווחים לפני המחרוזת שמייצגת כל פוקימון):

The Trainer **name** is **age** years old and has the following pokemons (**x** in total):

The **pokemon_class name** of level **level** with **hit_points** HP.

The **pokemon_class name** of level **level** with **hit_points** HP.

...

כאשר שחור מסמן את התווים הקבועים ואדום את ערכי השדות המתאימים (שימו לב כי x אינו שדה אמיתי של המופע Trainer).
 לדוגמה:

The Trainer Ash is 17 years old and has the following pokemons (3 in total):

The Pikachu pika of level 40 with 46 HP.

The Wartortle warty of level 23 with 60 HP.

The Charmander charmey of level 13 with 46 HP.

```
def get_name(self):
```

השיטה מחזירה את השם של המאמן.

```
def get_age(self):
```

השיטה מחזירה את הגיל של המאמן.

```
def get_exp_modifier(self):
```

השיטה מחזירה את מקדם הניסיון של המאמן.

```
def get_pokemon_lst(self):
```

השיטה מחזירה את רשימת הפוקימונים של המאמן.

```
def change_pokemon_lst(self, pokemon, pokemon_id):
```

השיטה משנה את הפוקימון שנמצא באינדקס `pokemon_id` ברשימת הפוקימונים של המאמן, ומחליפה אותו ב `pokemon` שנכנס כקלט. שימו לב כי זוהי המתודה היחידה בעבודת הבית שדרכה ניתן לשנות משתנה גם מחוץ למחלקה.

```
def catch_pokemon(self, pokemon):
```

מטרת שיטה זו הנה תפיסה של פוקימון חופשי. בהינתן פוקימון נחשב את סיכויי התפיסה שלו על ידי המאמן. סיכויי תפיסת הפוקימון על ידי המאמן מחושבים על פי הנוסחה הבאה:

$$capture_chances = (catch_rate \times exp_modifier \times \frac{100 - hit_points}{100})\%$$

במידה וסיכויי התפיסה גדולים מ 50% הפוקימון יתווסף לסוף רשימת הפוקימונים של המאמן, והמתודה תדפיס את ההודעה הבאה:

```
trainer_name caught pokemon_name
```

לדוגמה:

```
Ash caught pika
```

אחרת, המתודה תדפיס את ההודעה הבאה:

```
trainer_name couldn't catch pokemon_name
```

לדוגמה:

```
Ash couldn't catch pika
```

Comparison operators (`==`, `>`, etc.):

ממשו את אופרטורי ההשוואה בין שני מאמנים על פי כמות החיים (hit_points) הכוללת של הפוקימונים שלהם. כך שמאמן שכמות החיים הכוללת של הפוקימונים שלו גדולה יותר הוא יותר גדול/חזק. שימו לב שאין חובה לממש את כל אופרטורי השוואה בשביל להשתמש בכולם.
רשימת האופרטורים:

```
__eq__(self, other) for equality
__ne__(self, other) for inequality
__gt__(self, other) for greater than
__lt__(self, other) for less than
__ge__(self, other) for greater than or equal to
__le__(self, other) for less than or equal to
```

`def __add__(self, other):`

עליכם לממש את אופרטור החיבור אשר ייצור מופע חדש של Trainer אשר מהווה איחוד של שני מאמנים. המופע החדש של Trainer יהיה בעל התכונות הבאות:

1. השם שלו הוא חיבור של שני שמות המאמנים הקודמים כך ששם המאמן היותר חזק מופיע ראשון, ולאחר מכן שם המאמן השני. המחרוזת "-" מופיעה בין שמותיהם של שני המאמנים.
2. הגיל שלו יהיה הגיל הממוצע של שני המאמנים (מעוגל כלפי מטה).
3. מקדם הניסיון שלו יהיה ממוצע של שני מקדמי הניסיון של שני המאמנים.
4. רשימה מאוחדת של הפוקימונים של שני המאמנים, כאשר הפוקימונים של המאמן החזק יותר יופיעו ראשונים (לפי סדרם המקורי), והפוקימונים של המאמן החלש יותר יופיעו לאחר מכן (גם לפי סדרם המקורי).

מספר הערות:

1. ברגע שנעשה חיבור בין שני מאמנים הם הופכים למאמן חדש וזוהי פעולה בלתי הפיכה. כלומר, לאחר חיבור בין שני מאמנים, ניתן להניח שמאותו הרגע לא תתבצע גישה למופעי המאמנים שהיו לפני ביצוע פעולת החיבור, שכן הם ממשיכים להתקיים.
2. לאחר שאוחדו המאמנים ניתן לחבר אליהם גם מאמן נוסף. כל הוראות פעולת החיבור עדיין תקפות במקרה זה, ואפשר להתייחס למאמן שנוצר אחרי איחוד המאמנים הקודם כאילו היה מאמן אחד. ראו דוגמאות מצורפות.

דוגמאות הרצה:

```
>>> from Pikachu import Pikachu
>>> from Wartortle import Wartortle
>>> warty = Wartortle("warty", 43, "fire", 27, 65, 73, 90)
Traceback (most recent call last):
...
    raise ValueError
ValueError
>>> warty = Wartortle("warty", 43, "water", 27, 65, 73, 90)
>>> pika = Pikachu("pika", 40, "electric", 19, 43, 60, 70, 3)
>>> pika.get_damage(warty)
15
>>> warty.get_damage(pika)
5
>>> from Trainer import Trainer
```

```
>>> ash = Trainer("Ash", 18, 6.0)
>>> ash.catch_pokemon(pika)
Ash caught pika
>>> ash
The Trainer Ash is 18 years old and has the following pokemons (1 in
total):
    The Pikachu pika of level 19 with 43 HP
>>> misty = Trainer("Misty", 18, 5.5)
>>> misty.catch_pokemon(warty)
Misty caught warty
>>> misty
The Trainer Misty is 18 years old and has the following pokemons (1 in
total):
    The Wartortle warty of level 27 with 65 HP
>>> ash >= misty
False
>>> combined_trainer = ash + misty
>>> combined_trainer
The Trainer Misty-Ash is 18 years old and has the following pokemons
(2 in total):
    The Wartortle warty of level 27 with 65 HP
    The Pikachu pika of level 19 with 43 HP
>>> from Charizard import Charizard
>>> charzy = Charizard("charzy", 45, "fire", 36, 80, 99, 85)
>>> brook = Trainer("Brook", 21, 4.1)
>>> brook.catch_pokemon(charzy)
Brook couldn't catch charzy
>>> brook
The Trainer Brook is 21 years old and has the following pokemons (0 in
total):
>>> may = Trainer("May", 22, 7.3)
>>> may.catch_pokemon(charzy)
May caught charzy
>>> may
The Trainer May is 22 years old and has the following pokemons (1 in
total):
    The Charizard charzy of level 36 with 80 HP
>>> may < combined_trainer
True
>>> combined_trainer += may
>>> combined_trainer
The Trainer Misty-Ash-May is 20 years old and has the following
pokemons (3 in total):
    The Wartortle warty of level 27 with 65 HP
    The Pikachu pika of level 19 with 43 HP
    The Charizard charzy of level 36 with 80 HP
>>> combined_trainer.get_exp_modifier()
6.525
```

:Battle

מחלקה אשר מייצגת קרב בין פוקימונים (ומאמניהם).
 קרב מתואר על ידי השדות הבאים אליהם אי אפשר לגשת ולא ניתן לשנות מחוץ למחלקה (כאשר כתוב "בין הערכים" הכוונה כולל):

- **trainer1** - משתנה המייצג את המאמן אשר יתקוף ראשון בתחילת הקרב. ניתן להניח קלט תקין.

- **trainer2** - משתנה המייצג את המאמן השני. ניתן להניח קלט תקין.

ממשו את מחלקת Battle:

```
def __init__(self, trainer1, trainer2):
```

בנאי המאתחל את שדות המחלקה.
 במידה ואחד מהשדות קיבל ערך לא תקין יש לזרוק חריגה מתאימה.

```
def dual_battle(self, trainer1_pokemon_id, trainer2_pokemon_id):
```

שיטה זו מקבלת אינדקס של פוקימון מתוך רשימת הפוקימונים של המאמן הראשון, ואינדקס של פוקימון מתוך רשימת הפוקימונים של המאמן השני, ומנהלת ביניהם דו-קרב. דו-קרב הינו סדרה של סיבובים, הנמשכת עד שאחד מהפוקימונים לא יכול להמשיך להילחם. בכל סיבוב מתחיל הפוקימון של המאמן הראשון לתת מכה, ולאחריו הפוקימון של המאמן השני נותן מכה (אם הוא עדיין יכול להילחם לאחר שקיבל את המכה).
 על המתודה להחזיר tuple עם שני איברים. האיבר הראשון ייצג את מספר הסיבובים שקרו בקרב והאיבר השני יהיה הספרה 1 במידה והפוקימון של המאמן הראשון ניצח, הספרה 2 במידה והפוקימון של המאמן השני ניצח, והספרה 0 במידה ושני הפוקימונים לא יכולים להילחם עם סוף הקרב (חשבו כיצד זה אפשרי).
 ניתן להניח שהקלט תקין, אך יש לבדוק ששני היצורים יכולים להילחם בתחילת הקרב, במידה ולא עליכם להחזיר באיבר הראשון של ה tuple 0 סיבובים ובאיבר השני את הספרה המתאימה לפי החוקיות שתוארה.

```
def total_battle(self):
```

שיטה זו מנהלת קרב כולל בין המאמנים.

הקרב הכולל מתנהל באופן הבא:

1. המאמן הראשון בוחר את הפוקימון הראשון המופיע ברשימה שלו, במידה והפוקימון יכול להילחם. אחרת, יבחר את הפוקימון הראשון שיכול להילחם ברשימה שלו. אם אף אחד מהפוקימונים שלו לא יכול להילחם, יש לבדוק האם למאמן השני יש פוקימונים שיכולים להילחם, במידה וכן, המאמן השני יוגדר כמנצח, ואחרת הקרב יסתיים בתיקו.

2. המאמן השני בוחר את הפוקימון מתוך רשימת הפוקימונים שלו, שיכול להסב הכי הרבה נזק לפוקימון שנבחר ע"י המאמן הראשון, ובתנאי שהוא יכול להילחם. אחרת, יבחר את הפוקימון הראשון שעדיין יכול להילחם ולהסב הכי הרבה נזק לפוקימון שנבחר ע"י המאמן הראשון.
3. מתנהל קרב בין שני הפוקימונים.
4. לאחר שהקרב בין שני הפוקימונים מסתיים, אם רק אחד מהפוקימונים לא יכול להמשיך להילחם, המאמן שפוקימון זה שייך לו, יבחר את הפוקימון הבא ברשימה שלו שיכול להילחם ויסב הכי הרבה נזק לפוקימון שניצח בקרב הקודם שנוהל. אם שני הפוקימונים לא יכולים להמשיך להילחם בסוף הקרב הקודם שנוהל, המאמן הראשון יבחר את הפוקימון עם האינדקס הבא ברשימה שעדיין יכול להילחם, ושוב המאמן השני יבחר את הפוקימון מרשימת הפוקימונים שלו, שיכול להילחם ויסב הכי הרבה נזק לפוקימון שנבחר על ידי המאמן הראשון.
5. שלבים 1-4 חוזרים על עצמם עד אשר לאחד או לשני המאמנים לא נותרו פוקימונים שיכולים להילחם.

על השיטה להדפיס את אחת משתי התוצאות האפשריות לקרב:

במידה ואחד מהמאמנים ניצח תודפס המחרוזת:

"Trainer **name** won the battle in **total_rounds** rounds"

במידה והקרב הסתיים בתיקו תודפס המחרוזת:

"The battle ended with a draw"

כאשר שחור מסמן את התווים הקבועים ואדום את ערכי השדות המתאימים.

דוגמאות הרצה:

דוגמה ראשונה:

```
>>> from Pikachu import Pikachu
>>> from Blastoise import Blastoise
>>> from Trainer import Trainer
>>> from Battle import Battle
>>> pika = Pikachu("pika", 40, "electric", 26, 90, 63, 85, 2)
>>> blasty = Blastoise("blasty", 43, "water", 37, 82, 93, 113)
>>> ash = Trainer("ash", 18, 6.0, [pika])
>>> misty = Trainer("misty", 18, 5.5, [blasty])
>>> battle = Battle(ash, misty)
>>> battle_score = battle.dual_battle(0, 0)
>>> battle_score
(4, 1)
>>> ash
The Trainer ash is 18 years old and has the following pokemons (1 in total):
    The Pikachu pika of level 26 with 33 HP
>>> misty
The Trainer misty is 18 years old and has the following pokemons (1 in total):
    The Blastoise blasty of level 37 with -2 HP
```

דוגמה שנייה:

```
>>> from Pikachu import Pikachu
>>> from Charmander import Charmander
>>> from Charizard import Charizard
>>> from Wartortle import Wartortle
>>> from Trainer import Trainer
>>> from Battle import Battle
>>> charzy = Charizard("charzy", 45, "fire", 37, 93, 93, 82)
>>> pika = Pikachu("pika", 40, "electric", 32, 35, 60, 40, 2)
>>> ash = Trainer("ash", 18, 6.0, [charzy, pika])
>>> ash
The Trainer ash is 18 years old and has the following pokemons (2 in total):
    The Charizard charzy of level 37 with 93 HP
    The Pikachu pika of level 32 with 35 HP
>>> warty = Wartortle("warty", 43, "water", 30, 78, 80, 93)
>>> charmy = Charmander("charmy", 41, "fire", 15, 57, 63, 44)
>>> misty = Trainer("misty", 18, 5.5, [warty, charmy])
>>> misty
The Trainer misty is 18 years old and has the following pokemons (2 in total):
```



```
The Wartortle warty of level 30 with 78 HP
The Charmander charmy of level 15 with 57 HP
>>> battle = Battle(ash, misty)
>>> battle.total_battle()
The battle ended with a draw
>>> misty
The Trainer misty is 18 years old and has the following pokemons (2 in
total):
    The Wartortle warty of level 30 with 0 HP
    The Charmander charmy of level 15 with -6 HP
>>> ash
The Trainer ash is 18 years old and has the following pokemons (2 in
total):
    The Charizard charzy of level 37 with -12 HP
    The Pikachu pika of level 32 with 2 HP
>>> charzy
The Charizard charzy of level 37 with 93 HP
>>> pika
The Pikachu pika of level 32 with 35 HP
>>> ash = Trainer("ash", 18, 6.0, [pika, charzy])
>>> misty = Trainer("misty", 18, 5.5, [warty, charmy])
>>> misty
The Trainer misty is 18 years old and has the following pokemons (2 in
total):
    The Wartortle warty of level 30 with 78 HP
    The Charmander charmy of level 15 with 57 HP
>>> battle = Battle(ash, misty)
>>> battle.total_battle()
Trainer ash won the battle in 5 rounds
>>> misty
The Trainer misty is 18 years old and has the following pokemons (2 in
total):
    The Wartortle warty of level 30 with 3 HP
    The Charmander charmy of level 15 with 5 HP
>>> ash
The Trainer ash is 18 years old and has the following pokemons (2 in
total):
    The Pikachu pika of level 32 with 3 HP
    The Charizard charzy of level 37 with 34 HP
>>> charzy
The Charizard charzy of level 37 with 93 HP
>>> ash = Trainer("ash", 18, 6.0, [pika, charzy])
>>> misty = Trainer("misty", 18, 5.5, [charmy, warty])
>>> battle = Battle(misty, ash)
>>> battle.total_battle()
The battle ended with a draw
>>> misty
The Trainer misty is 18 years old and has the following pokemons (2 in
total):
    The Charmander charmy of level 15 with 5 HP
```

```

The Wartortle warty of level 30 with -4 HP
>>> ash
The Trainer ash is 18 years old and has the following pokemons (2 in
total):
    The Pikachu pika of level 32 with 3 HP
    The Charizard charzy of level 37 with 8 HP

```

שימו לב:

- המופעים של הפוקימונים שהפכו שייכים לכל אחד מהמאמנים הם אינם אותם המופעים שנמצאים ב global scope. על כן כשנדפיס את charzy למשל, הוא אותו ה charzy שנוצר בהתחלה (ולא זה ששייך למאמן והשתנה במהלך הקרב שנוהל).
- קיימת תלות בסדר הפוקימונים של כל אחד מהמאמנים. בנוסף, הקרב תלוי גם במי המאמן שהתחיל את הקרב (סדר המאמנים שקיבלה המחלקה Battle).

דוגמה שלישית:

```

>>> from Pikachu import Pikachu
>>> from Charmander import Charmander
>>> from Charmeleon import Charmeleon
>>> from Charizard import Charizard
>>> from Squirtle import Squirtle
>>> from Wartortle import Wartortle
>>> from Blastoise import Blastoise
>>> from Trainer import Trainer
>>> from Battle import Battle
>>> charmilious = Charmeleon("charmilious", 42, "fire", 19, 60, 71, 68)
>>> warty = Wartortle("warty", 43, "water", 30, 78, 80, 93)
>>> pika = Pikachu("pika", 40, "electric", 32, 35, 60, 40, 2)
>>> charzy = Charizard("charzy", 45, "fire", 35, 95, 94, 83)
>>> ash = Trainer("ash", 18, 6.0, [charmilious, warty, pika, charzy])
>>> charizardious = Charizard("charizardious", 45, "fire", 37, 93, 93,
82)
>>> squirtly = Squirtle("squirtly", 41, "water", 9, 55, 49, 72)
>>> blasty = Blastoise("blasty", 43, "water", 37, 82, 93, 113)
>>> charmy = Charmander("charmy", 41, "fire", 15, 57, 63, 44)
>>> misty = Trainer("misty", 18, 5.5, [charizardious, squirtly,
blasty, charmy])
>>> battle = Battle(misty, ash)
>>> battle.total_battle()
Trainer misty won the battle in 15 rounds
>>> misty
The Trainer misty is 18 years old and has the following pokemons (4 in
total):
    The Charizard charizardious of level 37 with -12 HP
    The Squirtle squirtly of level 9 with 3 HP
    The Blastoise blasty of level 37 with 4 HP
    The Charmander charmy of level 15 with 38 HP
>>> ash

```

The Trainer ash is 18 years old and has the following pokemons (4 in total):

The Charmeleon charmilious of level 19 with 6 HP

The Wartortle warty of level 30 with 2 HP

The Pikachu pika of level 32 with -2 HP

The Charizard charzy of level 35 with 1 HP

```
>>> ash = Trainer("ash", 18, 6.0, [charmilious, warty, pika, charzy])
```

```
>>> misty = Trainer("misty", 18, 5.5, [charizardious, squirtly,
blasty, charmy])
```

```
>>> battle = Battle(ash, misty)
```

```
>>> battle.total_battle()
```

Trainer ash won the battle in 15 rounds

```
>>> misty
```

The Trainer misty is 18 years old and has the following pokemons (4 in total):

The Charizard charizardious of level 37 with 5 HP

The Squirtle squirtly of level 9 with 5 HP

The Blastoise blasty of level 37 with 4 HP

The Charmander charmy of level 15 with 3 HP

```
>>> ash
```

The Trainer ash is 18 years old and has the following pokemons (4 in total):

The Charmeleon charmilious of level 19 with 5 HP

The Wartortle warty of level 30 with 30 HP

The Pikachu pika of level 32 with -5 HP

The Charizard charzy of level 35 with -2 HP

בהצלחה ועבודה מהנה !

