תרגיל 5: OOP

תאריך פרסום:03.12.20

תאריך הגשה: 20.12.20 בשעה 23:59

מתרגלת אחראית: שיר כהן

משקל תרגיל: 4 נקודות

הנחיות כלליות:

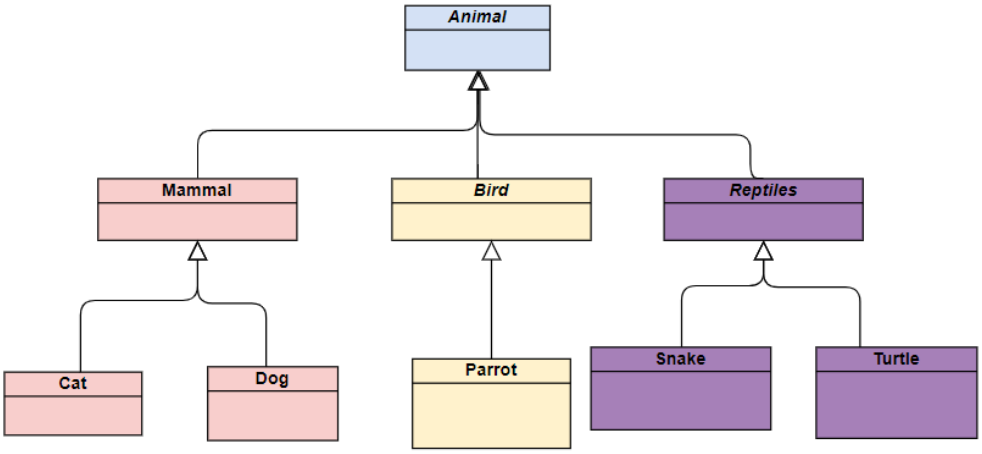
- העבודה תבוצע ביחידים.
- קראו את ההוראות לגבי הגשת תרגילי הבית באתר הקורס.
- מומלץ לקרוא את כל העבודה לפני תחילת הפתרון.
- כתבו תיעוד (הערות) שמסביר את הקוד שלכם. אין לכתוב הערות בעברית.
- עליכם להוריד את הקבצים מתיקיית "תרגיל בית 5 " מהמודל, ולהכניס את הקוד שלכם בשורות המתאימות בהתאם להוראות התרגיל.
- אין להשתמש בחבילות או במודולים, אלא אם הונחיתם לכך במפורש בתרגיל.
- ניתן להוסיף פונקציות עזר כל עוד לא נאמר אחרת בתרגיל\ בסעיף.
- ניתן להניח שהקלט תקין, אלא אם נכתב אחרת בשאלה.
- העבודה תיבדק באופן אוטומטי ולכן על הפלטים להיות בדיוק כפי שמוגדר בתרגיל (ללא רווחים מיותרים בהדפסות או בערכים המחוזרים). בנוסף, מערכת הבדיקות קוראת לפונקציה בצורה אוטומטית ולכן חתימה שגויה תגרור ציון 0 (הקפידו להגדיר את החתימות באופן מדויק כפי שהוגדר בתרגיל).
- העתקת קוד (משנים קודמות, מחברים או מהאינטרנט) אסורה בהחלט ועלולה להוביל לכישלון בקורס. בפרט אין להעביר קוד בין סטודנטים. **אל תעתיקו!**
- שאלות בנוגע לעבודה ישאלו ב-"פורום שאלות לתרגיל בית 5" במודל או בשעות הקבלה של המתרגל האחראי בלבד.
- את העבודה יש להגיש דרך מערכת ההגשה בכתובת: _ https://subsys.ise.bgu.ac.il/submission/login.aspx
  - יש להתחבר עם שם המשתמש והסיסמא של האוניברסיטה בצירוף תעודת זהות.
  - בכדי להעלות עבודה יש לבחור את הקורס Intro ואת משימה 5Assignment.
  - יש להעלאות את הקובץ הרצוי כך שהוא נמצא בתוך תיקייה עם שם התרגיל ועבור תיקייה זאת לבצע zip וללחוץ על . Upload
  - העבודה הועלתה בצורה תקינה רק לאחר שמוצגת הודעה " File uploaded successfully".
- טרם ההגשה אנא וודאו:
  - כל אחד מהקבצים רץ כנדרש.
  - המשתנים שכתבתם עם שמות משמעותיים (ללא שמות כמו a).

- בתחילת כל שאלה כתבתם הערות לקוד באנגלית.
- אין הדפסות מיותרות. וודאו שהסרתם הדפסות ובדיקות שביצעתם בקוד לפני ההגשה.
- וודאו כי כל חתימת פונקציה\שיטה, כמו גם שם מחלקה אותה כתבתם הן זהות לחלוטין לדרישות העבודה (כולל אותיות גדולות/קטנות), שם לא תקין יוביל לכישלון בהרצת הקוד על כל המשתמע מכך.

מטרות העבודה:OOP

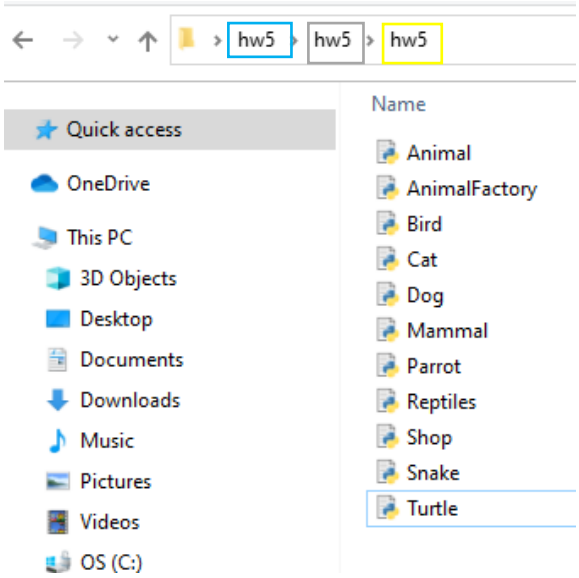
בהצלחה!

מבנה ההורשה



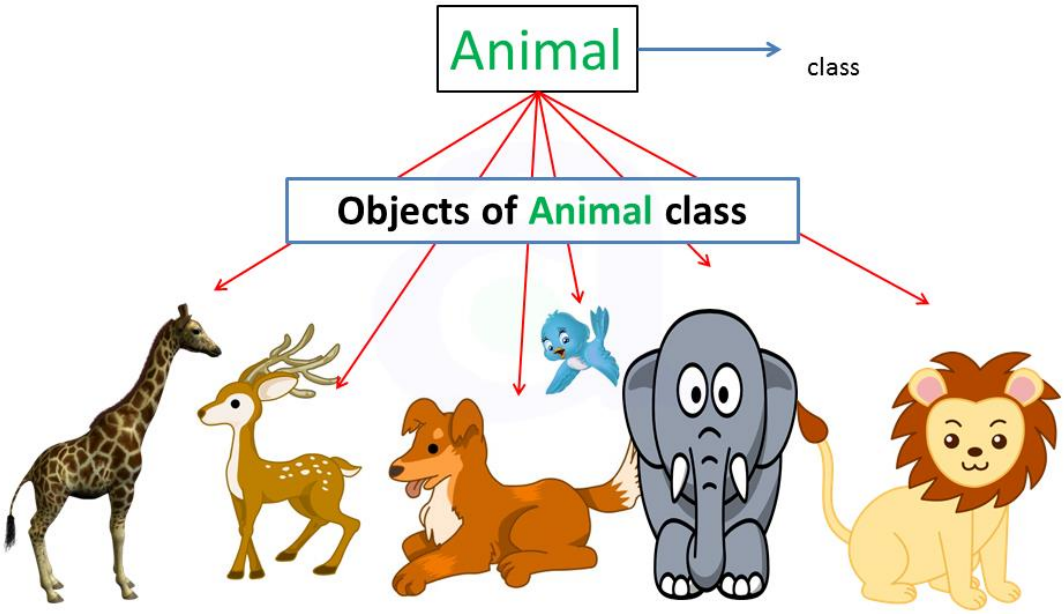
מבנה ההגשה –

את תיקייה שקיבלתם hw5 (המכילה את הקבצים) הכניסו לתיקייה נוספת בשם hw5. ולתיקייה החיצונית בצעו zip כאשר שם התיקייה המכווצת (של ה-zip) יהיה hw5.



בעקבות מגפת הקורונה העולמית והסגר שהוטל על המדינה הביקוש לבעלי חיים עלה. יוסי, בעל חנות החיות "חיות הם חלק מהמשפחה" ביקש ממך, מפתח תוכנה צעיר, לפתח עבורו תוכנית שתסייע לו להתמודד עם סוגי החיות בחנות.

חנות החיות, מורכבת ממספר מחלקות של בעלי חיים (יונקים, זוחלים ועופות) וכל מחלקה מכילה חיות שונות. כל החיות מאופיינות על ידי פרמטרים זהים (שם, מחיר וכוח) ובין החיות יש הבדלים כגון: יכולת תעופה, מהירות, קולות וכדומה. בחנות החיות יש נוהל שבו שתי חיות יכולות לשחק אחת עם השנייה. בתום המשחק מוכרזת החיה המנצחת והחיה המפסידה.



דגש 1 – דוגמת הרצה למתודות השונות קיימת בסוף העבודה.

דגש 2 – העבודה ארוכה כדי לתאר את המשימה באופן הברור ביותר. שימו לב, התרגיל מכיל משימות רבות, אבל רובן המכריע מאוד פשוטות - ניתן לממש את רב המתודות בשורת קוד יחידה.

דגש 3 – נא לקרוא את <u>כל העבודה</u> לפני שאתם מתחילים לכתוב את הקוד.

חלק א'

מחלקת חיה (Animal)

המחלקה חיה (Animal) מוגדרת על ידי השדות הבאים:

- **nick_name** – כינוי בעל החיים המהווה מזהה חד חד ערכי. כלומר, אין שני מופעים (instance/object) עם כינוי זהה.
- **price** – מחיר החיה בחנות.
- **power** – כוח החיה. שדה פרטי.
- **type** – טיפוס החיה. שדה זה מוגדר בצורה דיפולטית.

השלימו את מימוש המחלקה Animal:

- ***def __init__(self, nick_name, price, power, type):***

בנאי המאתחל את שדות החיה.

<u>קלט:</u>

- **nick_name (str)** – מחרוזת לא ריקה.
- **price (int or float)** - על המחיר להיות גדול מ-0. אחרת עליכם לזרוק שגיאה מסוג ValueError. עליכם לשמור את הערך הנ"ל בתצורת float.
- **power (int or float)** - הכוח חייב להיות גדול מ-0 וקטן או שווה ל-100. אחרת עליכם לזרוק שגיאה מסוג ValueError. עליכם לשמור את הערך הנ"ל בתצורת float.
- **type (str)** – טיפוס החיה.

- ***__repr__(self):***

השיטה תחזיר מחרוזת שתייצג את החיה.

<u>פלט:</u>

- **(str)** - ב-bold מסומנות המילים הקבועות

Name: nick_name**, Price:** price **NIS**, **Power:** power

לדוגמא –

Name: parrot_1, Price: 8.0 NIS, Power: 9.0

- ***_get__power(self):***

השיטה תחזיר את כוח החיה.

<u>פלט:</u>

- **Power (float)**

- ***_set__power(self, new_power):***

השיטה תבצע השמה חדשה של כוח החיה.

<u>קלט:</u>

- **new_power (int or float)**– הכוח יישמר בתצורה של float. יש לוודא כי הערך החדש גדול מ-0 וקטן או שווה ל-100. אחרת לא תתבצע ההשמה (אין צורך לזרוק שגיאה במקרה זה).

- ***win(self):***

השיטה תחזיר מחרוזת כי החיה ניצחה במשחק עם חיה אחרת.

<u>פלט:</u>

- **(str)**- ב-bold מסומנות המילים הקבועות

nick_name **winner**

לדוגמא –

parrot_1 **winner**

- ***loss(self):***

השיטה תחזיר מחרוזת כי החיה הפסידה כלומר, כוח החיה המנצחת גבר על כוח החיה המפסידה.

<u>פלט:</u>

- **(str)**- ב-bold מסומנות המילים הקבועות

nick_name **loser**

לדוגמא –

parrot_1 loser

- ***__ge__(self, other):***

השיטה תחזיר True אם השדה power גדול או שווה לזה של חיה אחרת (other). במקרה ולא ניתן להעריך את יחסי הכוחות תיזרק שגיאה מסוג ValueError.

<u>קלט:</u>

- **other (object)**– אובייקט מולו משווים.

<u>פלט:</u>

- **(bool)**

- ***__eq__(self, other):***

השיטה תחזיר bool המציין אם שני אובייקטים שווים. שני אובייקטים שווים אם יש להם את אותו nick_name. . במקרה ולא ניתן להעריך אם האובייקטים שווים (עליכם לבצע בדיקת תקינות קלט) יוחזר False.

<u>קלט:</u>

- ***other* (object)**– אובייקט מולו משווים.

<u>פלט:</u>

- **(bool)**

- ***get_type(self):***

השיטה את הטיפוס של החיה, התואם לשדה type.

<u>פלט:</u>

  - **(str)**

מחלקת יונקים (Mammal)

המחלקה יונק (Mammal) יורשת ממחלקת Animal.

ממשו את המחלקה Mammal:

- ***__init__(self, nick_name, price, power, type):***

בנאי המאתחל את שדות החיה (כמתואר במחלקת Animal).

- ***speak(self):***

שיטה המציינת את קול החיה.

<u>פלט:</u>

  - **(str)**- ב-bold מסומנות המילים הקבועות

nick_name **says**

לדוגמא –

dog_1 says

מחלקת כלבים (Dog)

המחלקה כלב (Dog) יורשת ממחלקת Mammal.

ממשו את המחלקה Dog:

- ***__init__(self, nick_name, price, power, , type="Dog"):***

בנאי המאתחל את שדות החיה (כמתואר במחלקת Animal).

- ***speak(self):***

שיטה המציינת את קול החיה.

<u>פלט:</u>

  - **(str)**- ב-bold מסומנות המילים הקבועות

nick_name **says woof woof**

לדוגמא –

dog_1 says woof woof

רמז: כבר ראינו את ההדפסות האלו במתודות אחרות (הורשה).

- ***win(self):***

השיטה תחזיר מחרוזת כי החיה ניצחה.

<u>פלט:</u>

- **(str)**- ב-bold מסומנות המילים הקבועות

nick_name **says woof woof** nick_name **winner**

לדוגמא –

dog_1 says woof woof dog_1 winner

רמז: כבר ראינו את ההדפסות האלו במתודות אחרות (הורשה).

מחלקת חתולים (Cat)

המחלקה חתול (Cat) יורשת ממחלקת Mammal.

ממשו את המחלקה Cat:

- ***__init__(self, nick_name, price, power, type="Cat"):***

בנאי המאתחל את שדות החיה (כמתואר במחלקת Animal).

- ***speak(self):***

שיטה המציינת את קול החיה.

<u>פלט:</u>

- **(str)** - ב-bold מסומנות המילים הקבועות

nick_name **says meow**

לדוגמא –

cat_1 says meow

מחלקת זוחלים (Reptiles)

המחלקה זוחלים (Reptiles) יורשת ממחלקת Animal.

ממשו את המחלקה Reptiles:

- ***__init__(self, nick_name, price, power, type):***

בנאי המאתחל את שדות החיה (כמתואר במחלקת Animal).

- ***move(self):***

זוחלים הם בעלי חיים בעלי אנרגיה מוגבלת, כאשר הם נעים במהירות היכולת שלהם לשחק יורדת משמעותית. כאשר מתודה זאת מופעלת הכוח של החיה קטן לחצי מכוחו הנוכחי.

- ***__ge__(self, other):***

אם השיטה מופעלת על שני זוחלים אז שני הזוחלים יפעילו את שיטת move ואז תבוצע השוואה לפי מדיניות האב. אחרת (השיטה הופעלה על זוחל וחיה ממחלקה אחרת) המדיניות תהייה לפי הכוח של החיה.

<u>קלט:</u>

- ***other* (object)**– אובייקט מולו משווים.

<u>פלט:</u>

- **(bool)**

מחלקת נחשים (Snake)

המחלקה נחש (Snake) יורשת ממחלקת Reptiles.

ממשו את המחלקה Snake:

- ***__init__(self, nick_name, price, power, type="Snake"):***

בנאי המאתחל את שדות החיה (כמתואר במחלקת Animal).

- ***move(self):***

כאשר מתודה זאת מופעלת הכוח הנוכחי של החיה משתנה ועולה פי 2.5. כוח החיה לא יכול להיות גדול מ-100. ולכן, אם הכוח החדש עולה על 100 לא תתבצע השמה של הכוח החדש.

למשל – עבור כוח ששווה ל-99, והפעלת מתודה seed הכוח של החיה יישאר 99 (מאחר ו-100<2.5*99).

מחלקת צבים (Turtle)

המחלקה צב (Turtle) יורשת ממחלקת Reptiles.

ממשו את המחלקה Turtle:

- ***__init__(self, nick_name, price, power, type="Turtle"):***

בנאי המאתחל את שדות החיה (כמתואר במחלקת Animal).

- ***loss(self):***

השיטה תחזיר מחרוזת כי החיה הפסידה.

<u>פלט:</u>

- **(str)**- ב-bold מסומנות המילים הקבועות

nick_name **loser I always lose**

לדוגמא –

turtle_1 loser I always lose

מחלקת עופות (Bird)

המחלקה עופות (Bird) יורשת ממחלקת Animal. עוף (Bird) מוגדר על ידי השדה:

❖ **Fly** – שדה בוליאני המציין אם החיה יכולה לעוף. ערך ברירת המחדל הוא False.

ממשו את המחלקה Bird:

- ***__init__(self, nick_name, price, power, type):***

בנאי המאתחל את שדות החיה (כמתואר במחלקת Animal) ואת השדה fly.

- ***__ge__(self, other):***

אם האובייקט האחר, other אינו יכול לעוף אז יוחזר True. אחרת ההשוואה תתבצע על פי מדיניות האב.

<u>קלט:</u>

- ***other* (object)**– אובייקט מולו משווים.

מחלקת תוכים (Parrot)

המחלקה תוכי (Parrot) יורשת ממחלקת Bird.

ממשו את המחלקה Parrot:

- ***__init__(self, nick_name, price, power, type="Parrot"):***

בנאי המאתחל את שדות החיה (כמתואר במחלקת Animal). התוכי יוגדר על ידי יכולת תעופה (fly=True).

חלק ב'

Design pattern

"בהנדסת תוכנה, תבנית עיצוב (באנגלית: Design pattern) היא פתרון כללי לבעיה שכיחה בעיצוב תוכנה. תבנית עיצוב אינה עיצוב סופי שניתן להעבירו הישר לקוד, אלא תיאור או תבנית לדרך לפתרון בעיה, שעשויה להיות שימושית במצבים רבים. תבניות עיצוב מונחות עצמים מציגות לרוב יחסים וקשרי גומלין בין מחלקות או אובייקטים, בלי לפרט את המחלקות או אובייקטי היישום הסופיים המעורבים. אלגוריתמים אינם נחשבים כתבנית עיצוב, כיוון שהם פותרים בעיות חישוביות ולא בעיות עיצוב." (ההגדרה לקוחה מויקיפדיה). הנושא של "תבניות עיצוב" נלמד בהמשך התואר בקירוס ניתוח ותיכון, בתרגיל הנוכחי תחוו טעימה של הנושא באמצעות מימוש תבנית עיצוב מוכרת בשם Factory.

Factory

תבנית העיצוב Factory מאפשרת לייצר אובייקטים מבלי לציין את שם המחלקה המדויקת של האובייקט שייווצר. הדבר נעשה על ידי קריאה לאובייקט Factory במקום לבנאי של האובייקט שאותו נרצה ליצור. המוטיבציה העיקרית מאחורי תבנית עיצוב זו היא ליצור מחלקה אחת שתפקידה ליצר סוגים שונים של אובייקטים החולקים תכונות ופונקציונליות משותפות. התוצאה היא גמישות מוגברת ושימוש חוזר בקוד. מחלקת ה-Factory תקבל מידע אודות אובייקט נדרש, תיצור אותו ותחזיר את האובייקט מהסוג שצוין. לדוגמא, מחלקה בשם ShapeFactory יוצרת את כל האובייקטים מטיפוס shape. השיטה create_shape מקבלת את שם האובייקט שנדרש וקוראת לבנאי המתאים.

```
class ShapeFactory:
    def create_shape(self, name):
        if name == 'circle':
            radius = input("Enter the radius of the circle: ")
            return Circle(float(radius))

        elif name == 'rectangle':
            height = input("Enter the height of the rectangle: ")
            width = input("Enter the width of the rectangle: ")
            return Rectangle(int(height), int(width))

        elif name == 'square':
            width = input("Enter the width of the square: ")
            return Square(int(width))
```

אתם מוזמנים להרחיב על תבנית עיצוב Factory כאן.

מחלקת מפעל חיות (AnimalFactory)

השלימו את מימוש המחלקה AnimalFactory:

- ***create(type_animal, nick_name, price, power):***

שיטה היוצרת חיה לפי הפרמטרים. בנוסף, השיטה תדפיס את ההודעה המצוינת בטבלה לפי סוג החיה שנוצרה. כאשר אין חיה מתאימה וחוזר None לא יודפס דבר.

קלט:

- **type (str)**– שם החיה – על פי הטבלה הבאה (השם מתאים לשם של השדה type של כל מחלקה):

Print	Instance	type
Dog created	Dog	Dog
Cat created	Cat	Cat
Parrot created	Parrot	Parrot
Snake created	Snake	Snake
Turtle created	Turtle	Turtle

- **nick_name (str)** – מחרוזת לא ריקה.
- **Price (int or float)** - על המחיר להיות גדול מ-0. אחרת עליכם לזרוק שגיאה מסוג ValueError. עליכם לשמור את הערך הנ"ל בתצורת float.
- **Power (int or float)** - הכוח חייב להיות גדול מ-0 וקטן או שווה ל-100. אחרת עליכם לזרוק שגיאה מסוג ValueError. עליכם לשמור את הערך הנ"ל בתצורת float.

פלט:

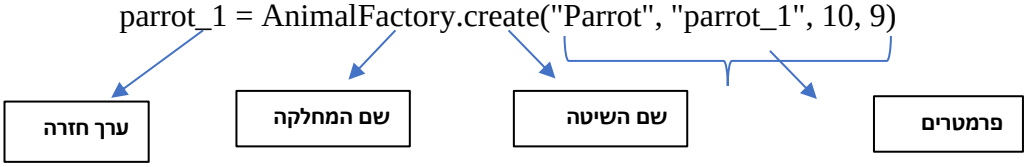
- **(Animal or None)** - השיטה תחזיר את החיה שנוצרה. במידה וה-type לא תואם לאף type של חיה יוחזר None.

רמז- עליכם להשתמש בתנאי שמשתנה לפי type החיה בדיוק כמו בדוגמא של הצורות. לאחר שמצאת את ה-type המתאים עליכם לקרוא לבנאי של המחלקה המתאימה.

לדוגמא –

קריאה תתבצע בצורה הבאה:

```
parrot_1 = AnimalFactory.create("Parrot", "parrot_1", 10, 9)
```



במקרה זה תודפס למסך ההודעה "Parrot created" ויוחזר האובייקט לתוך משתנה parrot_1.

דגש – קריאה לבנאי של כל אחת מהמחלקות היורשות ממחלקת Animal תעשה רק על ידי המחלקה AnimalFactory. כלומר, בכל מקום בקוד בו אתם יוצרים אובייקט מטיפוס Animal עליכם ליצור אותו באמצעות המחלקה AnimalFactory.

רמז -לאחר שתממשו את כל המשימה, חשבו האם כתבתם מתודה שגורמת ליצירת אובייקט של חיה. אם כן, תקנו את יצירתו על ידי המחלקה AnimalFactory

חלק ג'

מחלקת חנות (Shop)

המחלקה חנות (Shop) מוגדרת על ידי השדות הבאים:

- **name (str)**– שם החנות.
- **(float)(balance** – היתרה הכספית של החנות. בעזרת יתרה זאת החנות יכולה לקנות חיות נוספות על מנת להוסיף אותן לחנות. התקציב יהיה גדול או שווה ל-0 (הניחו קליט תקין).
- **animal_list (dictionary)**- מילון שמכיל את רשימת החיות בחנות. ה-key הינו שם החיה (nick_name) וה-value הינו ה- instance של האובייקט (Animal). באתחול החנות הרשימה תהייה ריקה. לא ניתן יהיה לגשת למילון מחוץ למחלקה.

השלימו את מימוש המחלקה Shop:

- ***__init__(self, name, balance):***

בנאי המאתחל את שדות החנות.

<u>קלט:</u>

- **name (str)** – מחרוזת לא ריקה.
- **balance (int or float)** - היתרה הנוכחית של החנות.

- ***get_name(self):***

השיטה תחזיר מחרוזת שתייצג את שם החנות.

<u>פלט:</u>

- **(str)** - ב-bold מסומנות המילים הקבועות

name

לדוגמא –

My shop

- ***__add__(self, other):***

השיטה תוסיף חיה\חיות לחנות (אין לבצע העתקה לחיה אלא להוסיף את המצביע). החנות תנסה להוסיף את המספר המקסימלי של החיות לפי אילוצי תקציב כתלות במחיר (price) החיה. תקציב החנות יכול להגיע עד ל-0.

<u>קלט:</u>

- **Other (Animal or list)** - השיטה יכולה לקבל חיה בודדת או רשימה של חיות. הניחו כי לא תקבלו שם של חיה שקיימת ברשימה .

<u>פלט:</u>

- **(int)** - מספר החיות שהחנות רכשה.

לדוגמא –

עבור חיות

1. כלב בשם dog_1 במחיר 10.
2. חתול בשם cat_1 במחיר 5.
3. נחש בשם snake_1 במחיר 5.

ו- balance חנות 10 ₪.

השיטה תוסיף את החיה במקום השני (חתול) והחיה במקום השלישי (נחש).

הניחו כי לא יהיה מקרה בדיקה על המצב בסגנון הבא:

לדוגמא –

עבור חיות

1. כלב בשם dog_1 במחיר 10.
2. חתול בשם cat_1 במחיר 5.
3. נחש בשם snake_1 במחיר 5.

ו- balance חנות 5 ₪.

אין מדיניות איזו חיה תתווסף לחנות (האם חיה במקום השני או השלישי). ולכן הניחו כי מצב זה לא ייבדק.

- ***get__animals(self):***

השיטה תחזיר את מילון החיות. שימו לב כי לא נרצה לאפשר שינוי של רשימת החיות מבחוץ.

<u>פלט:</u>

- **(dictionary)**

- ***sell (self, nick_name):***

בשיטה זאת החנות מוכרת חיה. השיטה תסיר חיה מהמיליון של החנות ותוסיף לתקציב, balance, את מחיר החיה.

<u>קלט:</u>

- **nick_name (str)**

<u>פלט:</u>

- **(Animal or None)** – השיטה תחזיר את החיה שהוסרה. במקרה ולא ניתן להסיר את החיה יוחזר None.

- ***num_of_animals(self):***

השיטה תחזיר את כמות החיות בחנות.

<u>פלט:</u>

- **(int)**

- ***play(self,animal_1, animal_2):***

השיטה תאפשר לשתי חיות לצאת לגינת המשחק ולשחק. השיטה תבדוק אם חיה בשם animal_1 גוברת על חיה בשם animal_2. עבור החיה המנצחת תופעל שיטת win ועבור החיה המפסידה תופעל שיטת loss.

<u>קלט:</u>

- ***animal_1* (str)** - שם חיה.
- ***animal_2* (str)**– שם חיה.

<u>פלט:</u>

- **(str or False)** – השיטה תחזיר מחרוזת המייצגת את הפלט של מתודות win ו-loss. במקרה ואחת החיות שרוצים לבצע איתם את המשחק לא קיימת בחנות יוחזרו False.

לדוגמא –

shop.play(snake_1, parrot_1)

הפלט:

snake_1 winner**n**parrot_1 loser

שימו לב כי יש \n בין ההדפסה של הניצחון וההפסד.

דוגמת הרצה נוספת תוצג בהמשך.

חלק ד'

Unit test

זאת הפעם הראשונה שהינכם מתמודדים עם מספר רב של קבצים, שיטות ומחלקות. על מנת שתוכלו לבדוק את התרגיל שלכם בצורה קלה ונוחה.

הכלי מאפשר לבצע בדיקות לקוד שלכם בצורה קלה ונוחה. הכלי ישמש אתכם הן בהמשך התואר שלכם והן בתעשייה ולכן אנחנו מעודדים אתכם לקרוא עליו באינטרנט ולהרחיב את הידע שלכם בנושא מלבד הפירוט בעבודה ובתרגול.

לעבודה סופקה לכם מחלקת Test אשר מממשת מספר בדיקות שיאפשרו לכם לבדוק את עבודתכם.

<u>ניתן להגדיר test בצורה הבאה:</u>

נגדיר פונקציה ובתוך הפונקציה נבצע תרחיש מסוים שבודק שיטה בקוד. למשל, בדוגמא אנו בודקים כי הבנאי של המחלקה shop מבוצע כהלכה.

על מנת לבצע את ה-test עלינו להשתמש באובייקט self ובשיטת השוואה.

למשל בדוגמא,

```
self.assertEqual(shop.get_name(), "My shop", msg="The name is not the same")
```

שיטת השוואה | ערך צפוי (expected) | ערך אמיתי (actual) | אופציונאלי (הודעה שתוצג אם הערך האמיתי והצפוי לא אותו דבר)

מוזמנים לקרוא על כל המתודות להשוואה בקישור .

```python
def test_1(self):
    """
    Check init method of shop
    """
    shop = Shop("My shop", 10000)
    self.assertEqual(shop.get_name(), "My shop",
                     msg="The name is not the same")
    self.assertEqual(shop.balance, 10000,
                     msg="The balance is not the same")
    self.assertEqual(shop.get__animals(), {},
                     msg="The dict is not the same")
```

ניתן להריץ טסט בודד על ידי לחיצה על החץ הירוק בצד שמאל. לאחר ההרצה נקבל: פלט כי הטסט עבר.

```
Run: pytest for test.Test.test_1
Tests passed: 1 of 1 test – 0 ms
Test Results 0 ms
Launching pytest with arguments test.py::te

============================= test session
platform win32 -- Python 3.8.3, pytest-5.4.
cachedir: .pytest_cache
rootdir: C:\Users\shirs\Google Drive\סודרות
collecting ... collected 1 item

test.py::Test::test_1 PASSED

============================== 1 passed in

Process finished with exit code 0
9: Git  4: Run  6: Problems  TODO  Terminal  Python Console
```

אם למשל נשנה את הטסט לטסט שגוי:

```
def test_1(self):
    """
    Check init method of shop
    """
    shop = Shop("My shop", 10000)
    self.assertEqual(shop.get_name(), "????????",
                     msg="The name is not the same")
    self.assertEqual(shop.balance, 10000,
                     msg="The balance is not the same")
    self.assertEqual(shop.get__animals(), {},
                     msg="The dict is not the same")
```

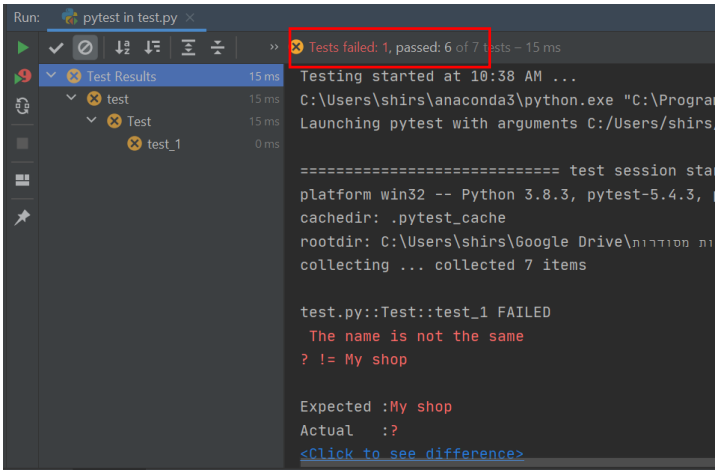
כאשר נריץ נקבל הודעה כי הטסט נכשל.

```
Run: pytest for test.Test.test_1
Tests failed: 1 of 1 test – 0 ms
Test Results 0 ms
  test 0 ms
    Test 0 ms
      test_1 0 ms
collecting ... collected 1 item

test.py::Test::test_1 FAILED                                    [100%]
 The name is not the same
???????? != My shop

Expected :My shop
Actual   :????????
<Click to see difference>

self = <test.Test testMethod=test_1>, first = 'My shop', second = '????????'
msg = 'The name is not the same'

    def _patched_equals(self, first, second, msg=None):
```

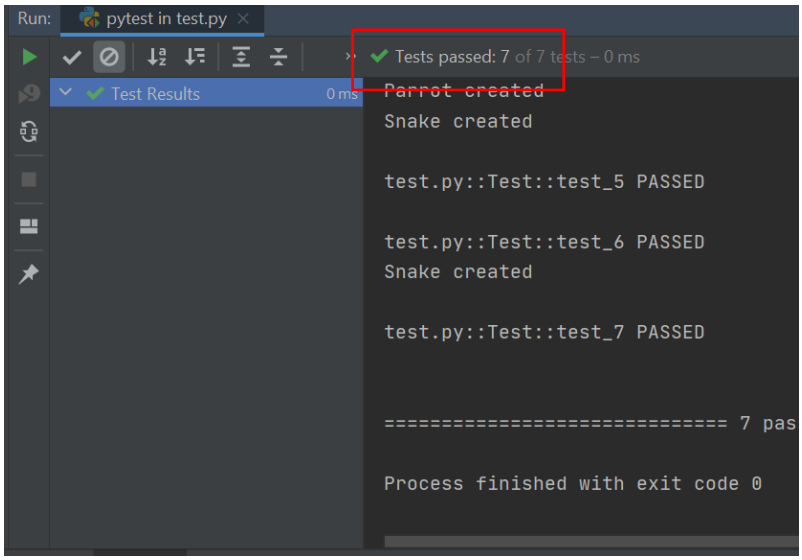
על מנת להריץ את כל הטסטים לחצו על החץ הירוק בצד שמאל.

```
82  if __name__ == "__main__":
83      unittest.main()
84
```

מצב שבו טסט אחד נכשל ו-6 עברו:



מצב שבו כל הטסטים עברו:



סיפקנו לכם שבעה טסטים שונים על מנת שתוכלו לבדוק את עבודתכם.

אנחנו מעודדים אתכם להוסיף טסטים נוספים על מנת לבדוק את עבודתכם. שימו לב כי **אנחנו <u>לא</u> מאשרים לשתף טסטים בינכם.**

ניתן לראות במחלקת ה-Test את אחד היתרונות של תבנית העיצוב Factory בכך שאנו נדרשים לעשות import אחד בלבד למחלקת AnimalFactory על מנת ליצור את כל החיות, מאשר לעשות import לכל מחלקה בנפרד.

בהצלחה ועבודה מהנה ! 😊