המכללה האקדמית להנדסה סמי שמעון

עבודת הגשה מס' 4

תאריך הגשה הוא **19/01/2023 עד 23:55**

- ✓ ניתן להכין את המטלה **בזוגות**
- ✓ רק **חבר אחד** בצמד **יגיש** בפועל את העבודה
- ✓ יש להגיש את הפתרון **תחת שם המכיל את מספרי ת"ז של כל המגישים**.
- ✓ יש להגיש קובץ בפורמט PY.
- ✓ **חובה** להשתמש בשמות הפונקציות המוגדרות.
- ✓ שימו לב, הפלט של דוגמאות ההרצה הוא בהתאם לסביבת הפיתוח Python IDLE.
- ✓ חובה לכל פונקציה להוסיף doc strings .
- ✓ הגשה דרך **מודל** בלבד!

- ✓ **אישורי ההארכה** יינתנו ע"י מרצה בלבד !

* הערה חשובה: קיים הבדל עקרוני בין הדפסה לבין החזרה של ערך מפונקציה! ברירת המחדל בהיעדר הוראת הדפסה מפורשת היא **החזרה בלבד**.

המכללה האקדמית להנדסה סמי שמעון

חלק 1 (Python) OOP ומימוש מערכת אובייקטים (Shmython)

שאלה 1

ממשו מחלקות(בסוגרים שמות לשדות) ופונקציה הבאות ב-**Python**

א) **Item** – מתארת פריט בתפריט במסעדה הכוללת שם(name), מחיר(price) וכמות קלוריות(calories). יש למשמ בנאי ופונקציות גנריות- **str** ו-**repr**.

ב) **Order** – מתארת הזמנה במסעדה הכוללת שם של מזמין(name) ורשימת פריטים בהזמנה(order_list) (רשימה של **Item**). יש למשמ בנאי ופונקציות גנריות- **str** ו-**repr**. בנוסף יש למשמ מתודות: **add_items** שמקבלת תפריט ואינדקס או רצף של אינדקסים ומוסיפה פריט או פריטים להזמנה לפי אינדקס/ים בתפריט. **remove_item** שמקבלת אינדקס בהזמנה ומורידה פריט מהזמנה. **total** מחשבת סכום לתשלום של ההזמנה. **calories** מחשבת סה"כ קלוריות בהזמנה. לפונקציה גנריות- **str** ביצירת מחרוזת יש להוסיף סכום לתשלום וכמות קלוריות של ההזמנה.

ג) **Restaurant** - מתארת מסעדה הכוללת שם(name), תפריט(menu) ורשימת הזמנות(orders). יש למשמ בנאי ופונקציה גנרית-**repr**. בנוסף יש למשמ מתודות: **add_menu** שמקבלת פריט ומוסיפה אותו לתפריט. **remove_menu** שמקבלת אינדקס ומוחקת פריט מתפריט. **print_menu** שמדפיסה פריטים בתפריט. **add_orders** שמוסיפה הזמנה או הזמנות לרשימת הזמנות. **print_orders** שמדפיסה רשימת הזמנות.

ד) פונקציה **min_calories** שמקבלת כפרמטר מסעדה ומחזירה הזמנה עם כמות קלוריות מינימלית או הודעה שאין הזמנות.

דוגמה להרצה(מחייבת):

```
>>> item1=Item('Big Mac',10,550)
>>> item1.price
10
>>> item1
Item('Big Mac',10,550)
>>> print(item1)
(Big Mac:10$:550cal)
>>> item2=eval(repr(item1))
>>> item2
Item('Big Mac',10,550)
>>> menu=[Item('McDouble',10,400),Item('Big_Mac',12,550),Item('McChicken',10,400),Item('Fries',5,320),
Item('Cappuccino',3,160),Item('Coca-Cola',5,210)]
>>> order1=Order('David')
>>> order1.add_items(menu,(1,3,5))
>>> order1
Order('David',[Item('McDouble',10,400), Item('McChicken',10,400), Item('Cappuccino',3,160)])
>>> print(order1)
(David, ((McDouble:10$:400cal),(McChicken:10$:400cal),(Cappuccino:3$:160cal)), total:23$,calories:960cal)
>>> order1.remove_item(1)
Remove Item: (McDouble:10$:400cal)
>>> order2=eval(repr(order1))
>>> order2.name='Tali'
>>> print(order2)
(Tali, ((McChicken:10$:400cal),(Cappuccino:3$:160cal)), total:13$, calories:560cal)
>>> order3=Order('Jim', menu, (2,4,5,6))
>>> print(order3)
(Jim, ((Big_Mac:12$:550cal),(Fries:5$:320cal),(Cappuccino:3$:160cal),(Coca-Cola:5$:210cal)), total:25$,
calories:1240cal)
>>> rest1=Restaurant('BurgerPoint', menu)
>>> rest1
Restaurant('BurgerPoint',[Item('McDouble',10,400), Item('Big_Mac',12,550), Item('McChicken',10,400),
Item('Fries',5,320), Item('Cappuccino',3,160), Item('Coca-Cola',5,210)],[])
>>> rest1.add_menu(Item('Green_Salad',12,434))
>>> rest1.remove_menu(1)
Remove Item from menu: (McDouble:10$:400cal)
>>> rest2=eval(repr(rest1))
>>> rest2.name='BurgerSheva'
>>> rest2
```

המכללה האקדמית להנדסה סמי שמעון

```
Restaurant('BurgerSheva',[Item('Big_Mac',12,550), Item('McChicken',10,400), Item('Fries',5,320),
Item('Cappuccino',3,160), Item('Coca-Cola',5,210), Item('Green_Salad',12,434)],[])
>>> rest1.print_menu()
 Menu:
 1) Big_Mac       12$  550cal
 2) McChicken     10$  400cal
 3) Fries          5$  320cal
 4) Cappuccino     3$  160cal
 5) Coca-Cola      5$  210cal
 6) Green_Salad 12$  434cal
>>> rest1.add_orders(Order('David',rest1.menu,(2,4,5,6)))
>>> rest1.add_orders((Order('Tali',rest1.menu,(1,3,5)),Order('Jim',rest1.menu,(1,2,3,5))))
>>> rest1.print_orders()
(David, ((McChicken:10$:400cal),(Cappuccino:3$:160cal),(Coca-Cola:5$:210cal),
(Green_Salad:12$:434cal)), total:30$, calories:1204cal)
(Tali, ((Big_Mac:12$:550cal),(Fries:5$:320cal),(Coca-Cola:5$:210cal)), total:22$, calories:1080cal)
(Jim, ((Big_Mac:12$:550cal),(McChicken:10$:400cal),(Fries:5$:320cal),(Coca-Cola:5$:210cal)), total:32$,
calories:1480cal)
>>> print(min_calories(rest1))
(Tali, ((Big_Mac:12$:550cal),(Fries:5$:320cal),(Coca-Cola:5$:210cal)), total:22$, calories:1080cal)
```

שאלה 2

ממשו אותן מחלקות ב-**Shmython** ופונקציה:

א) **ItemClass** - יש לממש פונקציה make_item_class() שכוללת בנאי ופונקציה **__str__**.

ב) **OrderClass** - יש לממש פונקציה make_order_class() שכוללת בנאי, פונקציה **__str__** ופונקציות **add_items**, **remove_item**, **total**, **calories**. לפונקציה **__str__** ביצירת מחרוזת יש להוסיף סכום לתשלום וכמות קלוריות של הזמנה.

ג) **RestaurantClass** - יש לממש פונקציה make_restaurant_class() שכוללת בנאי, פונקציה **__str__** ופונקציות **add_menu**, **remove_menu**, **print_menu**, **add_orders**, **print_orders**.

ד) פונקציה **min_calories1** עם אותה פעולה שיש לפונקציה בשאלה **1**.

<u>הערה</u>: יש לצרף קוד של מערכת אובייקטים שנבנה בכיתה להרצה – **make_class**.

<u>דוגמה להרצה(מחייבת)</u>:

```
>>> item1=ItemClass['new']('Big Mac',10,550)
>>> item1['get']('price')
10
>>> item1['get']('__str__')()
'(Big Mac:10$:550cal)'
>>> menu1=[ItemClass['new']('McDouble',10,400),ItemClass['new']('Big_Mac',12,550),
ItemClass['new']('McChicken',10,400),ItemClass['new']('Fries',5,320),ItemClass['new']('Cappuccino',3,160)
, ItemClass['new']('Coca-Cola',5,210)]
>>> order1=OrderClass['new']('David')
>>> order1['get']('add_items')(menu1,(1,3,5))
>>> order1['get']('__str__')()
'(David,((McDouble:10$:400cal),(McChicken:10$:400cal),(Cappuccino:3$:160cal)), total:23$,calories:960cal)'
>>> order1['get']('remove_item')(1)
Remove Item from order: (McDouble:10$:400cal)
>>> order1['get']('__str__')()
'(David, ((McChicken:10$:400cal),(Cappuccino:3$:160cal)), total:13$, calories:560cal)'
>>> order2=OrderClass['new']('Tali',menu1,(2,4,5,6))
>>> order2['get']('__str__')()
'(Tali, ((Big_Mac:12$:550cal),(Fries:5$:320cal),(Cappuccino:3$:160cal),(Coca-Cola:5$:210cal)), total:25$,
calories:1240cal)'
>>> rest1=RestaurantClass['new']('BurgerPoint',menu1)
>>> rest1['get']('__str__')()
```

המכללה האקדמית להנדסה סמי שמעון

```
'(BurgerPoint,((McDouble:10$:400cal),(Big_Mac:12$:550cal),(McChicken:10$:400cal),(Fries:5$:320cal),(Cap
puccino:3$:160cal),(Coca-Cola:5$:210cal)),())'
>>> rest1['get']('add_menu')(ItemClass['new']('Green_Salad',12,434))
>>> rest1['get']('remove_menu')(1)
Remove Item from menu: (McDouble:10$:400cal)
>>> rest1['get']('print_menu')()
 Menu:
 1) Big_Mac        12$  550cal
 2) McChicken      10$  400cal
 3) Fries           5$  320cal
 4) Cappuccino      3$  160cal
 5) Coca-Cola       5$  210cal
 6) Green_Salad  12$  434cal
>>> rest1['get']('add_orders')(OrderClass['new']('David',rest1['get']('menu'),(2,4,5,6)))
>>> rest1['get']('add_orders')((OrderClass['new']('Tali',rest1['get']('menu'),(1,3,5)),
OrderClass['new']('Jim',rest1['get']('menu'),(1,2,3,5))))
>>> rest1['get']('print_orders')()
(David, ((McChicken:10$:400cal),(Cappuccino:3$:160cal),(Coca-Cola:5$:210cal),
(Green_Salad:12$:434cal)), total:30$, calories:1204cal)
(Tali, ((Big_Mac:12$:550cal),(Fries:5$:320cal),(Coca-Cola:5$:210cal)), total:22$, calories:1080cal)
(Jim, ((Big_Mac:12$:550cal),(McChicken:10$:400cal),(Fries:5$:320cal),(Coca-Cola:5$:210cal)), total:32$,
calories:1480cal)
>>> min_calories1(rest1)['get']('__str__')()
'(Tali, ((Big_Mac:12$:550cal),(Fries:5$:320cal),(Coca-Cola:5$:210cal)), total:22$, calories:1080cal)'
```

המכללה האקדמית להנדסה סמי שמעון

שאלה 3

א) שינו את המימוש הקיים של הפונקציה **make_class**(לתת שם לפונקציה **make_class1**), כך שלא תהיה אפשרות לשנות טיפוס תכונות(שדות) של אובייקטים(מופעים) (ראה דוגמה להרצה). יש להשתמש במנגנון **<u>טיפול בחריגות</u>** במקרים שיש שינוי בטיפוס.

```
def make_account_class():
    return make_class1({'interest' : 0.05})
def make_save_account_class():
    def init(self, owner):
        self['set']('owner',owner)
        self['set']('balance',0)
    return make_class1({'__init__' : init, 'interest' : 0.03}, Account)
Account = make_account_class()
SaveAccount = make_save_account_class()
```

<u>דוגמה להרצה(מחייבת):</u>

```
>>> b = SaveAccount['new']('Bob')
>>> b['get']('owner')
'Bob'
>>> b['set']('bank','Leumi')
>>> b['get']('balance')
0
>>> b['set']('balance','Bob')
The balance attribute can be given a new value of type <class 'int>'
>>> b['set']('bank',5)
The bank attribute can be given a new value of type <class 'str>'
>>> b['set']('interest',1)
>>> b['set']('interest',0.75)
The interest attribute can be given a new value of type <class 'int'>
```

ב) שינו את המימוש הקיים של הפונקציה **make_class**(לתת שם לפונקציה **make_class1**), כך שתתמוך ביצירת עותקים של אובייקטים(מופעים) (ראה דוגמה להרצה).

<u>דוגמה להרצה(מחייבת):</u>

```
>>> b = SaveAccount['new']('Bob')
>>> b['set']('bank','Leumi')
>>> b['get']('owner')
'Bob'
>>> b['get']('balance')
0
>>> c=SaveAccount['copy'](b)
>>> c['get']('owner')
'Bob'
>>> c['set']('owner','Jim')
>>> b['get']('owner')
'Bob'
```

```
>>> c['get']('owner')
'Jim'
>>> c['set']('balance',100)
>>> b['get']('balance')
0
>>> c['get']('balance')
100
>>> c['set']('bank','Discount')
>>> b['get']('bank')
'Leumi'
>>> c['get']('bank')
'Discount'
```

המכללה האקדמית להנדסה סמי שמעון

חלק 2: פונקציות גנריות (Generic Functions)

שאלה 4

א) ייצרו שלוש מחלקות(**Shekel, Dollar, Euro**) שמייצגות מטבעות שונות.

דוגמה להרצה(מחייבת):

```
>>> Shekel(100)
Shekel(100)
>>> Dollar(50)
Dollar(50)
>>> Euro(80)
Euro(80)
>>> print(Shekel(100),Dollar(50),Euro(80))
100₪ 50$ 80€
```

נתון קוד הבאה:

```
rates ={('dollar', 'nis'): 3.45,('euro','nis'): 3.67}
rates[('euro','dollar')]=rates[('euro','nis')]/rates[('dollar', 'nis')]
def type_tag(x):
    return type_tag.tags[type(x)]
type_tag.tags = {Shekel: 'nis',Dollar: 'dollar', Euro: 'euro'}
```

ב) אתם מתבקשים לממש פעולה חיבור('**add**') בין אובייקטים מטיפוסים: **Shekel**, **Dollar**, **Euro** בעזרת פונקציה גנרית **apply**. יש להחזיר תוצאת פעולה ב-**Shekel** אם יש פעולה עם שקלים ולהחזיר תוצאה ב-**Dollar** אם יש פעולה בין **Euro** לבין **Dollar**.

דוגמה להרצה(מחייבת):

```
>>> apply('add',Euro(100),Shekel(200))
Shekel(567.0)
>>> apply('add',Dollar(30),Shekel(50))
Shekel(153.5)
>>> print(apply('add',Euro(30),Dollar(50)))
81.9$
```

ג) אתם מתבקשים לממש פעולה חיבור('**add**') וחיסור('**sub**') בין אובייקטים מטיפוסים: **Shekel**, **Dollar**, **Euro** בעזרת פונקציה גנרית **coerce_apply**. יש להחזיר תוצאת פעולה ב-**Shekel** אם יש פעולה עם שקלים ולהחזיר תוצאה ב-**Dollar** אם יש פעולה בין **Euro** לבין **Dollar**.

דוגמה להרצה(מחייבת):

```
>>> coerce_apply('add',Euro(100), Euro (200))
Euro(300)
>>> coerce_apply('add',Dollar(30),Shekel(50))
Shekel(153.5)
>>> print(coerce_apply('add',Euro(30),Dollar(50)))
81.9$
>>> coerce_apply('sub',Euro(100),Shekel(200))
Shekel(167.0)
>>> print(coerce_apply('sub',Shekel(200),Dollar(100)))
-145.0₪
>>> print(coerce_apply('add',Euro(50),Dollar(30)))
83.2$
```

ד) שינו את מערכת כך שהיא תאפשר לבצע פעולות הבאות:

דוגמה להרצה(מחייבת):

```
>>> Euro(100) + Euro (200)
Euro(300)
>>> Dollar(30) + Shekel(50)
Shekel(153.5)
>>> print(Shekel(200) + Euro(100))
567.0₪
>>> print(Euro(30) + Dollar(50))
81.9$
>>> Euro(100) - Shekel(200)
Shekel(167.0)
>>> print(Shekel(200) - Dollar(100))
-145.0₪
```

חלק 3: מבני נתונים רקורסיביים (Recursive Data Structures)

שאלה 5

נתון מימוש של מחלקה בשם **Tree** המייצגת עץ כללי. לכל צומת בעץ יש ערך פנימי (**value**) ורשימת בנים (**nodes**). העלים הם מופעים של **Tree** ללא רשימת ילדים.

א) השלימו פונקציה __**repr**__ בהגדרה של מחלקה:

```
class Tree():
    def __init__(self, value, nodes=None):
        self.value = value
        self.nodes = nodes

    def __repr__(self):
            <1>
```

ב) השלימו פונקציה **BuildTree**, שבהנתן עץ (**tree**) מיוצג כ-**tuple** מחזירה עץ חדש (מופע של **Tree**) כך שעלים הם בעלי ערכים וערך לכל צומת פנימית סכום ערכים של עלים בתת-עץ שמתחיל מהצומת. הפונקציה חייבת להיות **רקורסיבית!**

```
def BuildTree(tree):
    if        <1>:
        return        <2>
    <3>
    return    <4>
```

ג) כתבו פונקציה **MaxValue**, שבהנתן עץ (**tree**) הפונקציה מחזירה את הערך הגדול ביותר שיש לעלים.

```
def MaxValue (tree):
```

דוגמה להרצה(מחייבת):

```
>>> t=BuildTree((((2, 3), (4, (5, 6, (8, 2))))))
>>> t
Tree(30,[Tree(5,[Tree(2), Tree(3)]), Tree(25,[Tree(4), Tree(21,[Tree(5), Tree(6), Tree(10,[Tree(8),
Tree(2)])])])])
>>> MaxValue(t)
8
```

המכללה האקדמית להנדסה סמי שמעון

חלק 4: מפרש (Interpreter) וחריגות(Exceptions)

שאלה 6

אתם מתבקשים להרחיב/לעדכן את המפרש באופן הבא(יש לטפל בחריגות הרלוונטיות בכל הסעיפים):

א) שהמחשבון יתמוך בפעולת **type** שתחזיר טיפוס של אובייקט.

דוגמה להרצה(מחייבת):

```
calc> type(5)                calc> type(add(2,3,4))
<class int>                  <class Exp>
calc> type(3.5)              calc> type('Python')
<class float>                SyntaxError: unexpected
```

ב) שהמחשבון יתמוך בפעולת **ror** שמבצעת **סיבוב מעגלי ימינה** למספר(שלם או ממשי) לכמות ספרות. למספר ממשי לבצע סיבוב מעגלי ימינה גם לחלק השלם וגם למספר אחרי ניקודה בנפרד. את הפעולה יש לבצע בעזרת **Pipeline** עם שימוש בפונקציות (**map**, **filter**, **reduce**) וטיפוסים מובנים שלמדנו.

דוגמת להרצה(מחייבת):

```
calc> ror()
TypeError: ror requires exactly 2 argument
calc> ror(12345,3.5)
TypeError: 3.5 is not <class int>
calc> ror(3-,12345)
ValueError: -3 is not positive number
calc> ror(12,3)
ValueError: The number 12 is not possible for rotate
calc> ror(1234.2,3)
ValueError: The number 1234.2 is not possible for rotate
calc> ror(12345,3)
34512
calc> ror(12345.678,2)
45123.786
```

חלק 5: שאלות תיאורטיות

א) במערכת תכנות מונחה עצמים שמימשנו בכיתה (**Shmython**) ניתן להוסיף תכונות (שדות) חדשות גם לאובייקטים (מופעים) וגם למחלקות.

ב) פונקציה אשר מממשת שיטת **memorization** לא שומרת ערך מוחזר לכל ארגומנט שהיא קיבלה קודם, אלא מבצעת תהליך חישוב חדש בדומה ל-**recursive tree**.

ג) במימוש פונקציה גנרית **coerce_apply** מילון **coerce_apply.implementations** צריך לכלול פונקציה אחת עבור כל שילוב טיפוסים הנתמכים.

ד) במנגנון טיפול בחריגות שיש ב-**Python** כאובייקט חריגה ניתן "לזרוק"(ע"י **raise**) כל אובייקט מכל טיפוס אפשרי, מובנה או לא מובנה בשפה.

בהצלחה !