המכללה האקדמית להנדסה סמי שמעון

עבודת הגשה מס' 4

תאריך הגשה – 05/09/2021

הוראות הגשה: (אי קיום הוראות אלו עלול לגרום להורדת ציון!)

1. יש להגיש עד התאריך 05/08/2021 בשעה 23:55 למטלה הקשורה ב-Moodle בלבד.
2. אין להגיש בשום פנים ואופן למייל של מרצה או מתרגל - אך ורק ב-Moodle.
3. דחיית העבודה ניתנת רק במקרה של מילואים או אישור מחלה. אישורי ההארכה יינתנו ע"י מרצה בלבד.
4. יום איחור של בהגשת עבודת הגשה יעלה 3% מהציון. שבוע איחור יעלה 20% מהציון. אחרי שבוע של איחור, העבודה לא תתקבל. אין ערעורים במקרה של איחור!
5. ניתן להגיש את העבודה או ביחיד(מומלץ) או בזוגות. יש לרשום את כל השותפים לעבודה בתוך הקובץ.
6. תוכניות יש להגיש בקובץ עם סיומת PY ושאלות תיאורטיות בקובץ PDF מכווצים יחד ב-ZIP/RAR.
7. את העבודה בזוגות יש להגיש על ידי סטודנט אחד עם שם הקובץ שיהיה מורכב מהמילה "HW4" ושני מספרי ת"ז מופרדים בקו תחתון ביניהם.
   לדוגמא: HW4_123456789_123456789.zip
8. כל שאלה בנוגע לעבודה יש להפנות אך ורק לאחראי על העבודה במייל:
   finkm@ac.sce.ac.il . פניות בכל בדרך אחרת – לא יענו! בפנייה, יש לציין את : שם הקורס ופרטים מזהים.
9. פתרון שלא יעבוד בהרצה ב-IDLE בגרסה-python 3.8.x יקבל 0. בדקו היטב שאין שגיאות syntax.
10. במקרה של העתקה מלאה או חלקית של העבודה (מסטודנטים אחרים, מ-Internet או מכל מקום אחר), יינתן ציון 0 על העבודה של כלל הסטודנטים המעורבים והם יעלו לוועדת משמעת.
11. חובה לכל פונקציה להוסיף doc strings.
12. אין להשתמש בנושאים שטרם נלמדו.

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

1) ממש מחלקות הבאות ב-**Python**:
   - א) **Date** - תאריך כולל פירוט של שנה, חודש ויום בחודש.
   - ב) **Time** - זמן כולל שעה ודקה.
   - ג) **CalendarEntry** - רשומה בלו"ז כוללת רשימת מטלות (עם זמן התחלה וזמן סיום) בתאריך מסוים. אין שתי מטלות עם בדיוק אותם זמנים!

הערה 1: יש להגדיר פונקציות גנריות **repr** ו-**str** על הטיפוס האלו.
הערה 2: יש להשתמש בצמד של מחרוזות המייצגות מופעים של מחלקת **Time**(פלט של מימוש __**str**__) בתור מפתח ייחודי של מטלה(**task**)(ראה דוגמת הרצה).
הערה 3: יש להדפיס מטלות ממוינות בסדר של זמן התחלה.

דוגמת הרצה מחייבת:

```
>>> today = Date(2021, 8, 15)
>>> today
Date(2021, 8, 15)
>>> today.year
2021
>>> print(today)
‘15th of August, 2021’
>>> todo = CalendarEntry(2021, 8, 15)
>>> t = Time(10,0)
>>> str(t)
‘10:00’
>>> todo.addTask(“PPL lecture”, t, Time(13,0))
>>> todo.addTask(“PPL homework#4”, Time(14,0), Time(16,0))
>>> todo.tasks
{(’14:00’, ’16:00’): 'PPL homework#4', (‘10:00’, ’13:00’): 'PPL lecture'}
>>> print(todo)
Todo list for 15th of August, 2021:
1. 10:00-13:00 - PPL lecture
2. 14:00-16:00 – PPL homework#4
```

2) ממש אותן מחלקות ב –**Shmython**:
   - א) **Date** - יש לממש פונקציה **make_date_class**()
   - ב) **Time** - יש לממש פונקציה **make_time_class**()
   - ג) **CalendarEntry** - יש לממש פונקציה **make_calentry_class**()

הערה 1: יש לצרף קוד של מערכת אובייקטים (**basic.py**) שנבנה בכיתה להרצה.
הערה 2: אין לממש פונקציה **repr**, אך יש לממש את __**str**__ בטיפוס **Time** שתחזיר מחרוזת המייצגת זמן (לשימוש במילון של מטלות).
הערה 3: אין לתמוך בהדפסה של מטלות כמו בדוגמה הקודמת.

דוגמת הרצה מחייבת:

```
>>> Date = make_date_class()
>>> today = Date[‘new’](2021, 8, 15)
>>> today[‘get’](‘year’)
2021
>>> CalendarEntry = make_calentry_class()
>>> todo = CalendarEntry[‘new’](2021, 8, 15)
>>> Time = make_time_class()
>>> t = Time[‘new’](10,0)
>>> t[‘get’](‘__str__’)()
‘10:00’
>>> todo[‘get’](‘addTask’)(“PPL lecture”, t, Time[‘new’](13,0))
>>> todo[‘get’](‘addTask’)(“PPL homework#4”, Time[‘new’](14,0), Time[‘new’](16,0))
>>> todo[‘get’](‘tasks’)
{(’14:00’, ’16:00’): 'PPL homework#4', (‘10:00’, ’13:00’): 'PPL lecture'}
```

חלק 2: פונקציות גנריות (generic functions)

3) יש לממש שלוש מחלקות המייצגות מטבע: **Euro**, **Dollar**, **Shekel** בבנק ישראל. כל מופע של מטבע תאותחל עם סכום במטבע המקורי, אך ניתן לקבל את ערכו בשקלים (**amount**) . פעולת חיבור (**add**) בין **2** מופעים שלהם תחזיר סכום שלהם בשקלים. יש להגדיר מילון עזר עם שערי המרה בין מטבעות חוץ לשקל. יש לממש פונקציות הנדרשות להדפסה ותצוגה של מפרש. נא להימנע מחזרות בקוד.

הערה: יש ליישם שיטת העמסת אופרטור (ממשק משותף)

דוגמת הרצה מחייבת:

```
>>> rates ={('dollar', 'nis'): 3.22,('euro','nis'): 3.81}
>>> s = Shekel(50)
>>> d = Dollar(50)
>>> e = Euro(50)
>>> d.amount()
161.0
>>> e.amount()
190.5
>>> d + s
Shekel(211.0)
>>> print(d + s)
211.0nis
>>> z=eval(repr(d))
>>> print(z)
50.0$
>>> print(s)
50.0nis
>>> print(e)
50.0€
```

4) יש לממש פונקציה גנרית **apply** בהינתן שם של פעולה, ושמות טיפוסים של ארגומנטים. פונקציה תבצע את הפעולות חיבור וחיסור בין מטבעות שונות (אותם סוגים כמו בסעיף **3**) ותחזיר תוצאה כמופע של מחלקה (מטבע) המייצגת את הארגומנט הראשון (שמאל).

הערה: יש ליישם שיטה **dispatch on type**.

הערה: ניתן להיעזר במילון **rates** המורחב.

דוגמת הרצה מחייבת:

```
>>> apply('add', Shekel(50), Dollar(20))
Shekel(114.4)
>>> rates[('euro','dollar')] = 1.183
>>> apply('add', Dollar(50), Euro(20))
Dollar(73.66)
>>> apply('sub', Dollar(50), Euro(20))
Dollar(26.34)
```

5) יש לממש פונקציה גנרית **coerce_apply**, שבהינתן שם של פעולה ושמות טיפוסים של ארגומנטים (אותם טיפוסים כמו בסעיף **3**) תבצע את הפעולה לאחר המרה של מטבע חוץ לשקלים.

הערה: יש ליישם שיטה **coercion**.

הערה: התוצאה תמיד תהיה מטיפוס **Shekel**.

דוגמת הרצה מחייבת:

```
>>> coercions[('dollar','nis')](Dollar(50))
Shekel(161.0)
>>> rates[('euro','dollar')] = 1.183
>>> coerce_apply('add', Shekel(50), Dollar(20))
Shekel(114.4)
>>> coerce_apply('sub', Dollar(50), Euro(20))
Shekel(84.8)
```

חלק 3: חריגות (Exceptions)

6) שאלה זאת מתייחסת לשאלה **5** בעבודת בית מס' **3** (מימוש של **parking**).
יש לשדרג את המימוש שלכם ולהוסיף טיפול בחריגות (**ValueError**, **TypeError**, **IndexError**) של Python במקרים:

א) בהפעלת חניה(**parking**) יש לבדוק שתשלום עבור שעת חניה מספר חיובי ויש מקום לחניה בכל אחד מסוגי חניונים.

ב) בכניסה רכב לחניה (**'start_parking'**) מספר רכב ערך לא תקין(לדוגמא מחרוזת) וגם סוגי חניון מרשיה האפשרית.

ג) הדפסת פרטים של כל הרכבים הנמצאים בחניונים(**'print_list'**) לא יהיה צרך בבדיקה סוף רצף **prn['end']()**.

דוגמת הרצה מחייבת:

```
>>> park1=parking(-10,3,3,3)
the price value is bad
>>> park1=parking(10,0,3,3)
parking places error
>>> park1
>>> park1=parking(10,3,3,3)
>>> park1['start_parking']('aaa','VIP')
incorrect car number
>>> park1['start_parking'](223,'VIP1')
VIP1 is incorrect prking type
>>> park1['start_parking'](222,'Regular')
>>> park1['start_parking'](223,'Regular')
>>> park1['next_time']()
>>> park1['start_parking'](224,'Regular')
>>> park1['start_parking'](225,'VIP')
>>> prn=park1['print_list']()
>>> prn
{'next': <function parking.<locals>.print_list.<locals>.next at 0x03B6D618>}
>>> for _ in range(6):
prn['next']()
car: 222, parking type: Regular, parking time: 2
car: 223, parking type: Regular, parking time: 2
car: 224, parking type: Regular, parking time: 1
car: 225, parking type: VIP, parking time: 1
no car
no car
```

חלק 4. מבני נתונים רקורסיבי (recursive data structures)

7)

א) אתם מתבקשים לממש מחלקה בשם **Expr** המייצגת ביטוי אריתמטי, כאשר ארגומנטים יכולים להיות ביטויים. המחלקה חייבת להכיל מימוש של **repr**. העלים יכולים להיות מכל טיפוס שונה מ-**Expr**.

ב) יש לממש פונקציה רקורסיבית **buildExprTree** שבהינתן **tuple** שבנוי מ-**3** אלמנטים(שם אופרטור במקום הראשון וארגומנטים במקום השני והשלישי) המייצג ביטוי אריתמטי, תבנה ותחזיר מופע של **Expr** עבור ביטוי.

דוגמת הרצה מחייבת:

```
>>> exp = buildExprTree(('add', ('mul', 2, 3), 10))
>>> exp
Expr('add',Expr('mul',2,3),10)
```

חלק 5:מפרש (Interpreter)

8) אתם מתבקשים להרחיב את המפרש כך:

א) שהמחשבון יתמוך בנוסף למספרים שלמים(עשרוניים) גם במספרים מבסיסים: 2(**b**), 8(**q**), 16(**h**). כל מספר שלם יכול להיות רשום לפי בסיס. לדוגמה: **1afh**, **101101b**(תו בסוף המספר מציין בסיס). על המחשבון לזהות אם מספר נתון באחד משלושת הבסיסים, להפוך ולאחסן כמספר שלם עשרוני.

דוגמת הרצה מחייבת:

```
calc> add(1001b, ah)
19
calc> add(1001b, mul(1ah, 21q))
451
```

רמז: פונקציה **int**, בנוסף להפיכת מחרוזת למספר שלם גם יכולה לבצע את הפעולה לפי בסיסים. לדוגמה:

```
>>> int('10010',2)
18
```

ב) שלמחשבון תהיה אפשרות לבצע פעולת משלים ל-(**n-1**),במקרה שלנו **9**,ע"י אופרטור **compl** או סימן **!** אשר מקבל מספר שלם ומחזיר משלים ל-**9**(לכל ספרה) עבורו.
לדוגמה: **1564** -> **8435**, **12083** -> **87916**
את החישוב עצמו יש לבצע ע"י השלמת **pipeline**

int(".join((<1>(lambda x:<2>, list(<3>)))))

דוגמת הרצה מחייבת:

```
calc> compl(12083)
87916
calc> !(12083,12,24)
TypeError: compl requires exactly 1 argument
calc> compl(12083.12)
TypeError: 12083.12 is not <class int>
calc>
```

חלק 6: שאלות תיאורטיות

9)

א) במערכת אובייקטים שבנינו בכיתה (של **Shmython**) לא ניתן להוסיף פונקציות למאפיינים של מופע (**instance attributes**)

ב) במפרש של מחשבון (**calculator**) שכתבנו בכיתה פונקציות **calc_eval** ו-**calc_apply** עובדות בצורת רקורסיה הדדית.

ג) מטרת **memoization** היא לייעל חישוב רקורסיבי ע"י הקטנת ניצול זיכרון על חשבון זמן ריצה.

ד) ב- **Python 3.x** משתמשים בהצהרה **nonlocal** על מנת לעדכן תוכן של מבנה נתונים (למשל מילון) בסביבה כוללת (לא כולל מסגרת גלובאלית).

ה) במפרש של מחשבון (**calculator**) שכתבנו בכיתה פונקציית **calc_eval** מממשת **רקורסית עץ** על מנת להעריך את הארגומנטים של הביטוי המוערך.

ו) **פולימורפיזם בהורשה** ממומש בעזרת ממשק משותף בין הטיפוסים.

ז) פונקציות גנריות (**generic functions**) המוגדרות על פרמטרים מטיפוסים זרים (**parametric polymorphism**) ממומשות ע"י הגדרת ממשק משותף בין הטיפוסים של ארגומנטים.

בהצלחה !