**עבודת הגשה מס' 4**

תאריך הגשה הוא **14/01/2022 עד 14:59**

* 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!
* כל שאלה בנוגע לתרגיל יש להפנות אך ורק לאחראי על התרגיל - מיכאל באימייל: [**finkm@ac.sce.ac.il**](file:///C:\Users\marinal\Downloads\finkm@ac.sce.ac.il) . פניות בכל בדרך אחרת – לא יענו! בפנייה, יש לציין את : שם הקורס (**PPL**), מסי עבודה (**HW4**) ופרטים מזהים.
* **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**.

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

**שאלה 1**

ממש מחלקות הבאות ב: **Python**-

1. **Time**- תאריך כולל פירוט של שעות ודקות. יש לממש בנאי ופונקציות גנריות- **str** ו-**repr**.
2. **Event** – ביקור בתחנה למטופל שכולל זמן ביקור(**Time**) ושם של תחנה. יש לממש בנאי ופונקציות גנריות- **str** ו-**repr**.
3. **MedicalRecord**- שמייצג תיק רפואי בחדר מיון של בית חולים. תיק כולל שם של מטופל, מספר מזהה ורישום כל התחנות שעובר מטופל(**Event**). את תחנות יש לשמור במילון אשר זמן ביקור בתחנה בצורת מחרוזת יהיה מפתח. יש לממש בנאי שמקבל שם ומספר מזהה, פונקציה גנריות **repr** ומתודות: **add** שמוסיפה תחנה שעבר מטופל לתיק רפואי, **view** להדפסה תיק רפואי שכולל כל התחנות שעבר מטופל הממוינות לפי סדר עלה של זמנים בהתאם לדוגמת הרצה.

**דוגמה להרצה(מחייבת):**

>>> time1=Time(8,2)

>>> time1

Time(8,2)

>>> print(time1)

08:02

>>> time2=eval(repr(time1))

>>> print(time2)

08:02

>>> event1=Event(time1,'registration')

>>> event1

Event(Time(8,2),'registration')

>>> event2=eval(repr(event1))

>>> print(event2)

08:02 – registration

>>> record1=MedicalRecord('David',1)

>>> record1.add('08:02','registration')

>>> record1.data

{'08:02': Event(Time(8,2),'registration')}

>>> record1.add('09:15','doctor checkup')

>>> record1.add('08:45','doctor checkup')

>>> record1.add('09:00','procedure')

>>> record1.add('11:00','doctor checkup')

>>> record1.add('09:25','radiography')

>>> record1.add('11:30','hospital discharge')

>>> record1.add('10:30','blood test')

>>> record1.view()

name: David

ID: 1

08:02 - registration

08:45 - doctor checkup

09:00 - procedure

09:15 - doctor checkup

09:25 - radiography

10:30 - blood test

11:00 - doctor checkup

11:30 - hospital discharge

>>> record1

MedicalRecord('David',1,{'08:02': Event(Time(8,2),'registration'), '09:15': Event(Time(9,15),'doctor checkup'), '08:45': Event(Time(8,45),'doctor checkup'), '09:00': Event(Time(9,0),'procedure'), '11:00': Event(Time(11,0),'doctor checkup'), '09:25': Event(Time(9,25),'radiography'), '11:30': Event(Time(11,30),'hospital discharge'), '10:30': Event(Time(10,30),'blood test')})

**שאלה 2**

ממש אותן מחלקות ב -**Shmython**:

1. **TimeClass**- יש לממש פונקציה make\_time\_class() שכוללת בנאי ופונקציה **\_\_str\_\_**.
2. **EventClass** - יש לממש פונקציה make\_event\_class() שכוללת בנאי ופונקציה **\_\_str\_\_**.
3. **MedicalRecordClass**-יש לממש פונקציה make\_medical\_record\_class() שכוללת בנאי ופונקציות: **add** ו-**view**.

הערה: יש לצרף קוד של מערכת אובייקטים שנבנה בכיתה להרצה. מומלץ לשנות שם לפונקציה ל-**make\_class1**.

**דוגמה להרצה(מחייבת):**

>>> time1=TimeClass['new'](8,2)

>>> time1['get']('hours')

8

>>> time1['get']('\_\_str\_\_')()

'08:02'

>>> event1=EventClass['new'](time1,'registration')

>>> event1['get']('\_\_str\_\_')()

'08:02 - registration'

>>> record1=MedicalRecordClass['new']('David',1)

>>> record1['get']('add')('08:02','registration')

>>> record1['get']('data')['08:02']['get']('\_\_str\_\_')()

'08:02 - registration'

>>> record1

{'get': <function make\_class1.<locals>.new.<locals>.get at 0x037DEF58>, 'set': <function make\_class1.<locals>.new.<locals>.set at 0x037DEFA0>}

>>> record1['get']('add')('09:15','doctor checkup')

>>> record1['get']('data')

{'08:02': {'get': <function make\_class1.<locals>.new.<locals>.get at 0x03160100>, 'set': <function make\_class1.<locals>.new.<locals>.set at 0x03160148>}, '09:15': {'get': <function make\_class1.<locals>.new.<locals>.get at 0x03160220>, 'set': <function make\_class1.<locals>.new.<locals>.set at 0x03160268>}}

>>> record1['get']('add')('08:45','doctor checkup')

>>> record1['get']('add')('09:00','procedure')

>>> record1['get']('add')('11:00','doctor checkup')

>>> record1['get']('add')('09:25','radiography')

>>> record1['get']('add')('10:30','blood test')

>>> record1['get']('add')('11:30','hospital discharge')

>>> record1['get']('view')()

name: David

ID: 1

08:02 - registration

08:45 - doctor checkup

09:00 - procedure

09:15 - doctor checkup

09:25 - radiography

10:30 - blood test

11:00 - doctor checkup

11:30 - hospital discharge

**שאלה 3**

1. שינו את המימוש הקיים של הפונקציה **make\_class**, כך שלכל מחלקה יהיה מאפיין של שם המחלקה (**name**). שתהיה אפשרות לקבל שם המחלקה גם דרך מחלקה וגם דרך האובייקט(ראה דוגמה להרצה).

def **make\_account\_class**():

def **\_\_init\_\_**(self, owner, ID):

self['set']('owner', owner)

self['set']('ID', ID)

return **make\_class**('Account',{ '\_\_init\_\_' : \_\_init\_\_ , 'interest' : 0.05})

Account = **make\_account\_class**()

def **make\_saving\_account\_class**():

**interest**=0.075

**type\_acc**='saving'

def **strAccount**(self):

s=self['get']('name')+'(owner:'+self['get']('owner')+',ID:'+str(self['get']('ID'))

return s+',interest:'+ str(self['get']('interest'))+')' make\_class('SaveAccount',locals(),Account)

SaveAccount = **make\_saving\_account\_class**()

**דוגמה להרצה(מחייבת):**

>>> **Account**['get'](**'name'**)

'Account'

>>> **SaveAccount**['get'](**'name'**)

'SaveAccount'

>>> acc=**SaveAccount**['new']('Bob',1)

>>> acc['get'](**'name'**)

'SaveAccount'

1. שינו את המימוש הקיים של הפונקציה **make\_class**, כך שדרך כל אובייקט או מחלקה תהיה אפשרות לקבל כל התכונות ומתודות לכל השרשרת הורשה של מחלקות ממחלקת בסיס. הפעלה דרך אובייקט תאפשר לקבל בנוסף כל התכונות של אובייקט.

**דוגמה להרצה(מחייבת):**

class: SaveAccount

methods:

strAccount

attributes:

interest: 0.075

type\_acc: saving

name: SaveAccount

object attributes:

owner: Bob

ID: 1

>>> acc=**SaveAccount**['new']('Bob',1)

>>> Account[**'info'**]()

class: Account

methods:

\_\_init\_\_

attributes:

interest: 0.05

name: Account

>>> acc[**'info'**]()

class: Account

methods:

\_\_init\_\_

attributes:

interest: 0.05

name: Account

**חלק 2: פונקציות גנריות (generic functions )**

**שאלה 4**

ייצרו שלוש מחלקות(**Feets**, **Inches**, **Centimeters**) שמייצגות יחידות מדידות שונות. יחס בין יחידות מדידה:

1 **ft** = 30.48 **cm**, 1 **inch** = 2.54 **cm**, 1 **ft** = 12 **inch**.

יש לממש פונקציה גנרית **apply** שבהינתן שם של פעולה ושני הארגומנטים כאובייקטים מטיפוסים יחידות מדידה מחשבת ומחזירה את התוצאה של הפעולה על הארגומנטים. יש להחזיר אובייקט מטיפוס יחידת מדידה הראשון שפונקציה מקבלת. יש לתמוך ב-2 פעולות:

1. חיבור (**add**)
2. כפל (**mul**)

**דוגמה להרצה(מחייבת):**

>>> **apply**('**add**',**Inches**(1),**Centimeters**(150))

**Inches**(60.05511811023622)

>>> **apply**('**add**',**Centimeters**(100),**Feets**(1.5))

**Centimeters**(145.72)

>>> **apply**('**add**',**Feets**(2),**Inches**(5))

**Feets**(2.4166666666666665)

>>> **apply**('**sub**',**Inches**(1.5),**Centimeters**(100))

**Inches**(-37.87007874015748)

>>> **apply**('**sub**',**Feets**(2),**Inches**(5))

**Feets**(1.5833333333333333)

>>> **apply**('**sub**',**Centimeters**(100),**Inches**(15))

**Centimeters**(61.9)

**הערה:** יש לתמוך ב"ערבוב" של כל הטיפוסים. יש לספק פונקציות לפעולות אריתמטיות עבור כל קומבינציית טיפוסים אפשרית. אין להמיר את כל הטיפוסים לטיפוס אחד!

**שאלה 5**

יש לממש פונקציה גנרית **coerce\_apply** שבהינתן שם של פעולה ושני הארגומנטים כאובייקטים מטיפוסים יחידות מדידה מחשבת ומחזירה את התוצאה של הפעולה על הארגומנטים, ע"י המרה של אחד מאובייקטים לטיפוס של אובייקט השני.

**הערה:** תוצאה תהיה באחד מטיפוסים שפונקציה מקבלת כארגומנטים!

**דוגמה להרצה(מחייבת):**

>>> **coerce\_apply**('**add**',**Inches**(1),**Centimeters**(150))

**Inches**(60.05511811023622) # or **Centimeters**(152.4)

>>> **coerce\_apply**('**add**',**Centimeters**(100),**Feets**(1.5))

**Centimeters**(145.72) # or **Feets**(4.7808398950131235)

>>> **coerce\_apply**('**add**',**Feets**(2),**Inches**(5))

**Feets**(2.4166666666666665) # or **Inches**(29)

>>> **coerce\_apply**('**sub**',**Inches**(1.5),**Centimeters**(100))

Inches(-37.87007874015748) # or **Centimeters**(-96.19)

>>> **coerce\_apply**('**sub**',**Feets**(2),**Inches**(5))

**Feets**(1.5833333333333333) # or **Inches**(19)

>>> **coerce\_apply**('**sub**',**Centimeters**(100),**Inches**(15))

**Centimeters**(61.9) # or **Inches**(24.37007874015748)

**חלק 3: חריגות Exceptions))**

**שאלה 6**

שאלה זאת מתייחסת לשאלה בעבודת בית מס' **3** )מימוש של **make\_medical\_Record**).

יש לשדרג את המימוש שלכם ולהוסיף טיפול בחריגות (**ValueError**, **TypeError**, **IndexError**) של Python במקרים:

1. בהפעלת **make\_medical\_Record** יש לבדוק שפונקציה מקבלת פרמטרים מטיפוסים מתאמים.
2. בהוספת תחנה יש לבדוק שזמן נתון בפורמט **'XX:XX'** והוא חוקי.
3. בהוספת תחנה יש לבדוק שתחנה מרשימה תחנות הקיימות וקבלה ושחרור חולה ניתן להוסיף פעם אחד בלבד.
4. בהדפסה תחנות (**'next'**) יש לטפל שמקום ברשימת תחנות קימות.

**דוגמה להרצה(מחייבת):**

>>> mr=**make\_medical\_Record**(1,'David')

<class 'TypeError'> : invalid parameters type

name: <class 'int'>

ID: <class 'str'>

>>> mr=**make\_medical\_Record**('David',1)

>>> mr['addData']('11:30:20','registration')

<class 'TypeError'> : invalid time format XX:XX

time: 11:30:20

>>> mr[**'addData'**]('11:3','registration')

<class 'TypeError'> : invalid time format XX:XX

time: 11:3

>>> mr[**'addData'**]('11:62','registration')

<class 'TypeError'> : invalid time value

time: 11:62

>>> mr[**'addData'**]('11:30','registration')

>>> mr[**'addData'**]('11:40','registration')

<class 'ValueError'> : this event is present

event: registration

>>> mr[**'addData'**]('11:40','abcd')

<class 'ValueError'> : this event not possible

event: abcd

>>> mr[**'addData'**]('12:50','doctor checkup')

>>> mr[**'addData'**]('11:40','doctor checkup')

>>> mr[**'addData'**]('12:40','procedure')

>>> mr[**'addData'**]('13:30','radiography')

>>> mr[**'addData'**]('13:40','blood test')

>>> mr[**'addData'**]('15:00','hospital discharge')

>>> pr=mr[**'printRecord'**]()

David 1

>>> pr[**'next'**]()

'11:30:registration'

>>> for \_ in range(8):

pr[**'next'**]()

'11:40:doctor checkup'

'12:40:procedure'

'12:50:doctor checkup'

'13:30:radiography'

'13:40:blood test'

'15:00:hospital discharge'

<class 'IndexError'> : list index out of range

<class 'IndexError'> : list index out of range

**חלק 4:מבני נתונים רקורסיביים ( Recursive Data Structures )**

**שאלה 7**

נתון מימוש של מחלקה בשם **Tree** המייצגת עץ בינארי. לכל צומת בעץ יש ערך (**entry**), לכל צומת פנימי יש שני בנים (**left, right**). עלים הם מופעים של **Tree** ללא בנים.

class **Tree**():

def \_\_init\_\_(self, entry, left=None, right=None):

self.entry = entry

self.left = left

self.right = right

def **\_\_repr\_\_**(self):

if not self.left and not self.right:

return "Tree({0})".format(repr(self.entry))

return "Tree({0},{1},{2})".format(repr(self.entry),repr(self.left),repr(self.right))

1. השלימו פונקציה **build \_tree*,*** בהינתן **tuple** שבנוי מ-**3** אלמנטים לצומת פנימי ו-**tuple** מאלמנט **אחד** לעלה, בונה ומחזירה מופע של **Tree**. הפונקציה חייבת להיות **רקורסיבית**!

def **build\_ tree**(tree):

if **<1>:**

return **<2>**

return **<3>**

1. כתבו שתי פונקציות **max\_tree**(מקבלת מופע של **Tree** ומחזירה ערך מקסימלי) ו-**min\_tree** (מקבלת מופע של **Tree** ומחזירה ערך מינימלי).
2. כתבו פונקציה **is\_BST\_tree**,שבהנתןעץ (**tree**) פונקציה המחזירה **True** אם עץ הוא עץ חיפוש בינארי(**BST**), אחרת **False**. עץ חיפוש בינארי הוא עץ בינארי כאשר כל ערכי הצמתים בתת העץ הימני שלו גדולים מערך הצומת וכל ערכי הצמתים בתת העץ השמאלי שלו קטנים מערך הצומת.

def **is\_BST\_tree** (tree):

**דוגמה להרצה(מחייבת):**

>>> tree1 = (12, (6, (2,), (8,)), (15, (14,), (18,)))

>>> t1=**build\_tree**(tree1)

>>> t1

**Tree**(12, **Tree**(6, **Tree**(2), **Tree**(8)), **Tree**(15, **Tree**(14), **Tree**(18)))

>>> tree2 = (12, (6, (2,), (8,)), (15, (7,), (20,)))

>>> t2=**build\_tree**(tree2)

>>> t2

**Tree** (12, **Tree**(6, **Tree**(2), **Tree**(8)), **Tree**(15, **Tree**(7), **Tree**(20)))

>>> **max\_tree**(t1)

18

>>> **min\_tree**(t2)

2

>>> **is\_BST\_tree**(t1)

True

>>> **is\_BST\_tree**(t2)

False

**חלק 5:מפרש ( Interpreter)**

**שאלה 8**

אתם מתבקשים להרחיב/לעדכן את המפרש באופן הבא:

1. שהמחשבון יתמוך בחישוב שורש ריבועי(**sqrt** או סימן **V**).
2. תהיה אפשרות לעגל מספר עם אופרטור **round** או סימן **~** אשר שמקבל **2** מספרים **n** ו-**k** ומחזיר מספר הראשון(**n**) מעוגל בעל **k** ספרות אחרי נקודה. ניתן להשתמש בפונקציה מובנת של Python בשם **round** עם חתימה זהה.
3. שלמחשבון תהיה אפשרות לבצע חישוב **משקל** של מספר שלם (**weight** או **&**). פעולת חישוב משקל מוגדרת כפעולת משלים ל-**10** עבור ספרות של מספר שלם ללא אפסים.

לדוגמה: **1564** -> **9546**, **120083** -> **9827**

את החישוב עצמו יש לבצע ע"י השלמת **pipeline:**

**int(''.join( (<1> (lambda n: <2>,<3> (lambda n: <4>,list(<5>) ) ) ) ) )**

**דוגמת להרצה(מחייבת):**

calc> **round**(div(1,**sqrt**(add(30,6))),3)

0.167

calc> **V**(4,5)

TypeError: V requires exactly 1 argument

calc> **~**(1.333333333,3.5)

TypeError: ~ A second parameter must be integer

calc> **weight** (14075)

9635

calc> **&** (add (30,6), sub (6,30))

TypeError: & requires exactly 1 argument

calc> **weight** (140.75)

TypeError: 140.75 is not <class int>

**הערה לסעיפים א', ב' ו-ג':**

יש לטפל בחריגות הרלוונטיות לפעולות אלו.

**חלק 6: שאלות תיאורטיות**

1. המימוש הנוכחי (שהוצג בכיתה) של מחשבון לא מאפשר הרחבתו לתמיכה במשתנים (עם שינוים בקוד הקיים).
2. במערכת אוביקטים שמימשנו בכיתה (**Shmython**) פונקציה **get** של אובייקט יכולה להחזיר תכונה כלשהי של אובייקט **חוץ ממתודה**.
3. מטרת פונקציה **memo** (memoized function) היא לייעל חישוב רקורסיבי ע"י הקטנת ניצול זיכרון על חשבון זמן ריצה.
4. מערכת אובייקטים שכתבנו בכיתה (**Shmython**) יש תמיכה בפונקציות **גנריות** מסוג **פולימורפיזם בהורשה**.
5. מערכת אובייקטים שכתבנו בכיתה (**Shmython**) תפעיל בנאי דפולטיבי (**default**) בזמן יצירת אובייקט חדש אם מתכנת "שכח" להגדיר בנאי במחלקה המייצגת ובמחלקות אב.

**בהצלחה !**