מבנה תכנית בשפת python, מר עידן פלדברג.

<u>מטרה</u>: הקניית מיומנויות לכתיבת תכנית ב-python ומתן דגשים על מבנה התכנית.

<u>דרישות קדם</u>:

- ייבוא ושימוש במודולים מובנים בשפת התכנות python.
- הגדרה, מימוש וזימון פונקציות.

ייבוא מודולים מובנים לתכנית: • ניתן גם לא לייבא אף מודול. • ניתן לייבא מודול אחד או יותר.	import <module name>
הגדרת ערכי קבועים: • ניתן גם לא להגדיר אף קבוע. • ניתן להגדיר קבוע אחד או יותר. • שם קבוע יכיל אותיות גדולות בלבד מופרדות על – ידי קווים תחתונים	<constant name> = <constant value>
הגדרת פונקציות: • ניתן גם לא להגדיר אף פונקציה. • ניתן להגדיר פונקציה אחת או יותר.	def <function name>(<parameters>): <function body>
הגדרת פונקציית main: • זוהי הפונקציה הראשית ממנה ניתן לזמן את הפונקציות המוגדרות לפניה	def main(): <main body>
קטע עיקרי עם זימון פונקציה ראשית main: • התנאי מתקיים אם מריצים את קובץ התכנית ישירות ב-command line. • התנאי לא מתקיים אם מייבאים את קובץ התכנית מתוך קובץ תכנית אחר.	if __name__ == '__main__': main()

דוגמה למבנה תכנית:

```python
import math
MUL_FACTOR = 2


def calc_circle_perimter(radius):
    """ הפונקציה מקבלת רדיוס מעגל ומחשבת ומחזירה את היקפו """
    return MUL_FACTOR * math.pi * radius


def main():
    """ פונקציה ראשית הקולטת רדיוס מעגל ומחשבת ומדפיסה את היקפו """
    radius = float(raw_input("enter circle's radius: "))
    print calc_circle_perimter(radius)


if __name__ == '__main__':
    main()
```

דגשים, שימו לב ל:

- ייבוא המודול המתמטי math לשימוש בקבוע הפאי עבור חישוב היקף מעגל.
- פונקציה ראשית, main, המשתמשת בפונקציה לחישוב היקף מעגל על – פי רדיוס.
- זימון המותנה של הפונקציה הראשית main.
- לתיעוד הפונקציות.
- שתי שורות רווח בין פונקציה לפונקציה, ובין כל מרכיב עיקרי בתכנית לפונקציות.
- שמות המשמעותיים הניתנים למשתנים ולפונקציות.
- שמות קבועים: אותיות גדולות בלבד מופרדות בקווים תחתונים כדי להפריד בין תתי מילים משמעותיות בתוך שם הקבוע.

main מזמן את main?

נניח שתכנית זו שמורה בקובץ בשם perimeter.py,

אז פונקציית ה-main תופעל אם נריץ את הקובץ ישירות ב-command line כך: python perimeter.py:

```
C:\Users\user\Desktop>python perimeter.py
enter circle's radius: 0.5
3.14159265359

C:\Users\user\Desktop>_
```

לעומת זאת, פונקציית ה-main לא תופעל אם נריץ קובץ אחר המבצע ייבוא של הקובץ perimeter.py באמצעות ההוראה import perimeter השמורה בקובץ other_program.py:

```python
import perimeter
```

```
C:\Users\user\Desktop>python other_program.py

C:\Users\user\Desktop>
```

יש לשים לב שכאשר מייבאים קובץ מודול לקובץ תכנית אחר, הוראת import גורמת להרצת קטעי הקוד העיקריים שאינם תחת פונקציות, שהרי פונקציות אינן מבוצעות אם אין מזמנים אותן. לכן, היות שזימון פונקציית main נמצא בקטע קוד עיקרי שאינו תחת פונקציה, הרי שהוא מתבצע רק כתוצאה מקיום תנאי הוראת ה-if התוחמת אותו.

מערך שיעור תרגיל מעבדה FizzBuzz

מר תומר טלגם, גב' ריקי יפה, ד"ר דורון זהר

למה ללמד וללמוד Python בכיתה ט'?

- שפה חדשנית, שפת הסייבר.
- שפה קלה ללמידה וקריאה.
- המעבר בין סקראצ', JS, Python:
  - בכל הסביבות אנו לא מגדירים את טיפוס המשתנה.
  - השוני - אנו עוברים מסיבה גרפית לסביבת תכנותית.
- מוטיבציה להמשך לימודי העמ"ט בחט"ב והן לקראת בחירת המגמה בתיכון.
- הכנה ללימודים בתיכון.

איפה מתכנתים בפייתון?

הדרכה להתקנת פייתון: https://app.box.com/s/inca5vch0i1nr20glwnd

תרגיל כיתה

הכירו את פראן סלאק

https://www.youtube.com/watch?v=kBMvfkG0CeY

פראן סלאק האיש הכי בר מזל בעולם זכה בגיל 72 בסכום רב של כסף.

פראן רוצה לחלק את סכום זכייתו בין חבריו לפי סדר קרבתם כך שהרחוק ביותר ממנו יקבל שקל אחד וכל חבר הבא בתור יקבל בשקל אחד יותר מאשר קודמו.

כלומר החבר הרחוק ביותר יקבל 1 ש"ח, השני הרחוק ביותר 2 ש"ח וכך הלאה.

כתבו תכנית בפייתון הקולטת את סכום זכייתו.

על התכנית לחשב ולהדפיס מהו מספר החברים להם יחלק פראן בר המזל את כספו וכמה יקבל החבר האחרון.

הצעת פתרון:

```
winning_prize = int( raw_input("Enter The Winning Prize: ") )
gift = 0
friends = 0
sum = 0
while winning_prize > sum + gift:
   gift += 1
   sum += gift
   friends += 1
   print "Friend number ",friends," got ",gift, "money spent ", sum
print friends
```

הסבר:

```
winning_prize = int( raw_input("Enter The Winning Prize: ") )
```

למעשה זה משפט השמה כלומר אנחנו שמים ערך במשתנה.

איזה ערך? הערך שמוחזר מלהמיר ל-INT כלומר למספר שלם את הקלט שמגיע מהמשתמש ע"י הפקודה ()raw_input. שימו לב שמשתנה בפייתון מוגדר ללא סוג אך מקבל את הסוג מהערך שבתוכו במקרה זה רצינו מספר שלם. שימו לב גם לצורת הכתיבה אותיות קטנות, מילים מופרדות בקו תחתי.

עוד שלושה משפטי השמה עבור כל מה שנרצה לזכור:

```
gift = 0
friends = 0
sum = 0
```

gift – הסכום הניתן כרגע

friends – מונה צובר עבור חברים

sum – סכום הכסף שהוצא עד כה

```
while winning_prize > sum + gift:
        gift += 1
        sum += gift
        friends += 1
```

יש לשים לב כי בלולאת while כי אין סוגריים ולאחר הגדרת התנאי ישנם נקודתיים. למעשה שפת פייתון כתובה כמו שפה מדוברת, כך שאם רוצים שמשהו יהיה תחת נושא מסוים בכתיבה נוסיף נקודתיים.

פייתון יודעת מה בתוך מה לפי הזחות, כלומר רווחים/tab. בגוף הלולאה נגדיר פקודות להוספת ערכים למשתנים. שימו לב כי gift+=1 זהה ל- gift = gift + 1.

print "Friend number ",friends," got ",gift, "money spent ", sum

השורה האחרונה היא שורת הדפסה והיא מקבלת מספר ערכים להדפסה. הערכים מופרדים בפסיק.

תרגיל ראשון

יש לבקש מהתלמידים לבנות את שני קטעי הקוד הבאים (אין צורך בהגדרת מבנה הלולאה. עליהם לחפש את מבנה הלולאה ברשת):

- כתבו קטע תכנית בפייתון המדפיס את כל המספרים בין 1 ל-100 (כולל).
- כתבו קטע תכנית בפייתון המדפיס את כל המספרים בין 100 ל-1 (כולל).

פתרונות:

```
for x in xrange(1, 101):
    print x
```

```
for x in xrange(101, 0, -1):
    print x
```

מומלץ לרשום את מבנה הלולאה על הלוח על פי הנחיית התלמידים.

תרגיל שני

סדרת פיבונאצ'י היא הסדרה ששני איבריה הראשונים הם 1, 1 וכל איבר לאחר מכן שווה לסכום שני קודמיו. בהתאם לכך, איבריה הראשונים של הסדרה הם :

$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144...$. (קישור)

כתבו קטע תכנית בפייתון המדפיס את 12 הערכים הראשונים בסדרת פיבונצ'י.

פתרון ראשון

```
#Fibonacci
a = 1
b = 1
for i in xrange (1, 13):
   print a
   old_a = a
   a = b
   b = old_a + b
```

ניתן לפתור זאת גם בדרך הפייתונית הבאה:

```
#Fibonacci
a , b = 1 , 1
for i in xrange (1, 13):
   print a
   old_a, a = a , b
   b += old_a
```

תרגיל שלישי

כתבו קטע תכנית בפייתון המדפיס את כל המספרים בין 1 ל-100 (כולל) על פי הכללים הבאים:

- אם המספר כפולה של 3 (כלומר המספר מתחלק ב-3 ללא שארית) יש להדפיס "Fizz"
- אם המספר כפולה של 5 יש להדפיס "Buzz"
- אם המספר כפולה של 3 וגם 5 יש להדפיס "FizzBuzz"
- בכל מקרה אחר יש להדפיס את המספר עצמו.

בתרגיל זה יש לתת תלמידים לפתור את התרגיל על פי הידע הקיים (יתכן מכל שפה אחרת שלמדו בעבר).

פתרון אפשרי:

```
# Fizz Buzz
for x in range(1,  101):
    if x%3 == 0 and x%5== 0:
        print "FizzBuzz"
     elif x%3 == 0 :
        print "Fizz"
    elif x%5 == 0:
        print "Buzz“
```

```
    else:
        print x
```

מומלץ להתעכב על הדרך הפתרון השגויה והנפוצה הבאה בה תלמידים מתחילים לפתור את התרגיל לפי האפשרויות בשאלה (מספר המתחלק גם ב-3 וגם ב-5 ידפיס במקרה זה רק את Fizz):

```
# Fizz Buzz
for x in range(1,  101):
    if x%3 == 0 :
        print "Fizz"
    elif x%5 == 0:
        print "Buzz"
    elif x%3 == 0 and x%5== 0:
        print "FizzBuzz"
    else:
        print x
```

המשך תרגיל שלישי

עתה יש להקשות ולבקש לפתור את התרגיל על פי הגבלות הבאות, בפתרון ניתן להשתמש:

לולאה **אחת** בלבד.

שני תנאים if פשוטים ללא שימוש אפשרות else.

מותר להשתמש בתנאי המכיל תנאים לוגיים (and, or).

אפשרות פתרון ראשונה

```
for x in xrange(1,101):        #   for x in range(1,101):
   st_number = x
   st1 = ""
   if x % 3 == 0:
           st1 += "fizz"
           st_number = ""
   if x % 5 == 0:
           st1 += "buzz"
           st_number = ""
   print str(st_number) + str(st1)
```

אפשרות פתרון שנייה

```
for x in xrange(1,101):
  msg = ""
  if x % 3 == 0:
        msg += "fizz"
  if x % 5 == 0:
        msg += "buzz"
  print msg or str(x)
```

אפשרות פתרון שלישית ורביעית ללא תנאים:

```
for x in xrange(1,101):

  print (“Fizz” *(x %3 == 0) + ”Buzz” * (x % 5 == 0) or x)

for x in range(1,101):
  print '\n'.join(['Fizz'*(x % 3 == 0) + 'Buzz'*(x % 5 == 0) or str(x)])
```

אפשרות פתרון חמישי:

```
print '\n'.join(['Fizz'*(x % 3 == 0) + 'Buzz'*(x % 5 == 0) or str(x) for x in range(1,101)])
```

זה פתרון זהה לפתרון הקודם אך פשוט כתוב בשורה אחת (הסבר על הרעיון - Python One-Liners).

הסבר:

- תצרף לstring את המילה Fizz אם x מתחלק בשלוש ללא שארית
- תצרף Buzz אם xמתחלק בחמש ללא שארית
- תחזיר את ה-string או x (כלומר x כשהstring ריק)

.

דף עבודה בנושא string, מר עידן פלדברג

התרגילים מבוססים על Googe For Education , קישור

יש לפתור את הפעולות והזימונים הבאים על פי ההנחיות בכל שאלה. בכל פעולה יש להגדיר תיעוד (#) בצד כל שורת קוד.

 # A. donuts

Given an int count of a number of donuts, return a string

of the form 'Number of donuts: <count>', where <count> is

the number

passed in. However, if the count is 10 or more, then use the # word 'many'

instead of the actual count.

So donuts(5) returns 'Number of donuts: 5'

and donuts(23) returns 'Number of donuts: many'

פתרון ראשון: להשתמש בהוראת תנאי if else פשוט במבנה קלאסי ומסורתי:

```
def donuts(count):
  if count < 10:
    return 'Number of donuts: ' + str(count)
  else:
    return 'Number of donuts: many'
```

א. יש להשלים את שורות הקוד החסרות. ראו פתרון מודגש בצהוב בשורות החסרות בגוף הפונקציה.

ב. כתבו את מבנה הפעולה בשורה אחת, בדרך פייתונית:

הפתרון המוצג **בסעיף ובדרך א'** הוא בגישה קלאסית ומסורתית של תנאי if else פשוט המתפרש על פני מספר שורות, לעומת זאת נציג מספר דרכים אחרות לפתרון גוף הפונקציה בגישה פייתונית וחדשנית יותר:

פתרון שני: אפשר לוותר על רישום ההוראה else, שכן ברור שאם לא התקיימה הוראת ה-return ב-if, משמע התנאי לא התקיים ויש לקיים את הוראת ה-return ב-else מבלי לרשום else וללא הזחה. בדרך פתרון זו חסכנו רק את המילה else:

```
  if count < 10:
    return 'Number of donuts: ' + str(count)
  return 'Number of donuts: many'
```

<u>פתרון שלישי</u>: להשתמש בצורת רישום מיוחדת של הוראת if else המותאמת לשורה אחת:

```
return 'Number of donuts: ' + (str(count) if count < 10 else 'many')
```

<u>הסבר</u>: אם ערכו של count קטן מ-10, ערכו של count משורשר לתחילית 'Number of dounts', אחרת תת המחרוזת 'many' משורשרת לתחילית זו. שימו לב לסוגריים עגולים כדי ליצור סדר קדימות תקין בשרשור. יתרון דרך הפתרון הוא בשורה אחת.

<u>פתרון רביעי</u>: להשתמש בקשר הלוגי or בשורה אחת וללא if else כלל:

```
return 'Number of donuts: ' + ('many' * (count >= 10) or str(count))
```

<u>הסבר</u>: אם ערכו של count גדול או שווה ל-10 תת המחרוזת 'many' משורשרת לתחילית 'Number of donuts', אחרת ערכו של count משורשר לתחילית. אם התנאי מתקיים, שרשור בו עם מחרוזת שווה למחרוזת עצמה, שערכה הלוגי הוא True, ואז לא מגיעים לחלק ה-or. ואם התנאי לא מתקיים שרשור בו עם מחרוזת שווה למחרוזת ריקה, שערכה הלוגי הוא False, ולכן כן מגיעים לחלק ה-or שמחזיר את ערכו של count. שימו לב לסוגריים עגולים כדי ליצור סדר קדימות תקין בשרשור. יתרונות דרך הפתרון הם בשורה אחת וללא if else בכלל.

```
# B. both_ends
#

def both_ends(s):
  if len(s) < 2:
    return ''
  first2 = s[0:2]
  last2 = s[-2:]
  return first2 + last2
```

א. מה יודפס עבור כל אחד מהזימונים הבאים:

```
print both_ends('spring')
print both_ends('Hello')
print both_ends('a')
print both_ends('xyz')
```

ב. יש להשלים את מטרת ביצוע הקוד בראש הפעולה.

ג. כתבו את מבנה הפעולה בשורה אחת, בדרך פייתונית:

פתרון:

א.

```
print both_ends('spring') spng
print both_ends('Hello') Helo
print both_ends('a') מחרוזת ריקה
print both_ends('xyz') xyyz
```

ב. הפונקציה מקבלת מחרוזת, ומחזירה מחרוזת המורכבת משרשור שני התווים הראשונים במחרוזת ושני התווים האחרונים במחרוזת, אם אורך המחרוזת המתקבלת לפחות 2 תווים, ומחרוזת ריקה אחרת.

ג.

```
return (s[:2] + s[-2:]) * (len(s) >= 2)
```

<u>הסבר</u>: אם תנאי אורך מחרוזת גדול מ-2 מתקיים מוחזרת מחרוזת המורכבת משרשור בין 2 התווים הראשונים של המחרוזת לבין 2 התווים האחרונים של המחרוזת, ואם התנאי לא מתקיים מוחזרת מחרוזת ריקה. הרעיון הוא גם לא להשתמש במשתני עזר / ביניים בדרך פתרון זו.

```
# C. fix_start
#

def fix_start(s):
  front = s[0]
  back = s[1:]
  fixed_back = back.replace(front, '*')
  return front + fixed_back
```

א. מה יודפס עבור כל אחד מהזימונים הבאים:

```
print fix_start('babble')
print fix_start('aardvark')
print fix_start('google')
print fix_start('donut') donut
```

ב. יש להשלים את מטרת ביצוע הקוד בראש הפעולה.

ג. כתבו את מבנה הפעולה בשורה אחת, בדרך פייתונית:

פתרון:

א.

ב. print fix_start('babble') ba**le

ג. print fix_start('aardvark') a*rdv*rk

ד. print fix_start('google') goo*le

ה. print fix_start('donut') donut

ב. הפונקציה מקבלת מחרוזת, ומחזירה מחרוזת חדשה המתקבלת מהמחרוזת המתקבלת על – ידי החלפת כל מופע של התו הראשון במחרוזת בתו *, למעט התו הראשון.

ג.

return s[0] + s[1:].replace(s[0], '*')

<u>הסבר</u>: הרעיון לא להשתמש במשתני עזר / ביניים בדרך פתרון זו. פעולת ההחלפה לתו ה-* נעשית לאחר פעולת ה-slicing לנטילת תת המחרוזת ללא התו הראשון.

דרך פתרון נוספת:

return s[0] + s.replace(s[0], '*')[1:]

<u>הסבר</u>: גם בדרך פתרון זו הרעיון לא להשתמש במשתני עזר / ביניים. בדרך פתרון זו לעומת זאת, פעולת ה-slicing לנטילת תת המחרוזת ללא התו הראשון נעשית לאחר פעולת ההחלפה לתו ה-*.

 # D. MixUp

Given strings a and b, return a single string with a and b

separated

by a space '<a> <b>', except swap the first 2 chars of each

string.

e.g. 'mix', pod' -> 'pox mid'

'dog', 'dinner' -> 'dig donner'

Assume a and b are length 2 or more.

def mix_up(a, b):

א. יש להשלים את שורות הקוד.

ב. יש להריץ ולבדוק את הזימונים הבאים. בצד כל זימון מוגדרת התוצאה הרצויה.

print mix_up('mix', 'pod'), 'pox mid'

print mix_up('dog', 'dinner'), 'dig donner'

print mix_up('gnash', 'sport'), 'spash gnort'

print mix_up('pezzy', 'firm'), 'fizzy perm'

פתרון:
א.

```
return b[:2] + a[2:] + ' ' + a[:2] + b[2:]
```

 # E. verbing

```
# Given a string, if its length is at least 3,
# add 'ing' to its end.
# Unless it already ends in 'ing', in which case
# add 'ly' instead.
# If the string length is less than 3, leave it unchanged.
# Return the resulting string.
def verbing(s):
  if len(s) >= 3:
    if_______________
                ___________
    else:
                ___________
  return s
```

א. יש להשלים את שורות הקוד.

ב. יש להריץ ולבדוק את הזימונים הבאים. לצד כל זימון מוגדרת התוצאה הרצויה.

```
  print verbing('hail'), 'hailing'
  print verbing('swiming'), 'swimingly'
  print verbing('do'), 'do'
```

ג. כתבו את מבנה הפעולה בשורה אחת, בדרך הפייתונית.

פתרון:
א.

```
def verbing(s):
  if len(s) >= 3:
    if s.endswith('ing'):
                return s + 'ly'
    else:
                return s + 'ing'
  return s
```

ג.

return s + (len(s) >= 3)*(s.endswith('ing')*'ly' or 'ing')

<u>הסבר</u>: למחרוזת s משרשרים 'ing' או 'ly' אם אורך המחרוזת גדול או שווה 3, 'ly' אם המחרוזת מסתיימת ב-'ing', ו-'ing' אחרת. אם אורך המחרוזת קטן מ-3 משרשרים למחרוזת מחרוזת ריקה ולכן במקרה זה מוחזרת המחרוזת המקורית ללא שינוי.

 # F. not_bad

```
#
def not_bad(s):
  n = s.find('not')
  b = s.find('bad')
  if n != -1 and b != -1 and b > n:
      s = s[:n] + 'good' + s[b+3:]
  return s
```

א. מה יודפס לכל אחת משורות הפלט

```
print not_bad('This movie is not so bad')
print not_bad('This dinner is not that bad!')
print not_bad('This tea is not hot')
print not_bad("It's bad yet not")
```

ב. יש להשלים תיעוד בצד כל שורת קוד ולהשלים את מטרת ביצוע הקוד בראש הפעולה.

ג. כתבו את מבנה הפעולה בשורה אחת, בדרך פייתונית:

פתרון:

א.

```
print not_bad('This movie is not so bad') This movie is good
print not_bad('This dinner is not that bad!') This dinner is good!
print not_bad('This tea is not hot') This tea is not hot
print not_bad("It's bad yet not") It's bad yet not
```

ב. הפונקציה מקבלת מחרוזת ומחליפה תת מחרוזת המתחילה ב-'not' ומסתיימת ב-'bad' בתת המחרוזת 'good'.

```
def not_bad(s):
  n = s.find('not')    # not אינדקס המופע הראשון של תת המחרוזת
  b = s.find('bad')   # bad אינדקס המופע הראשון של תת המחרוזת
  if n != -1 and b != -1 and b > n:
```

```
    s = s[:n] + 'good' + s[b+3:]    # good החלפה לתת המחרוזת
  return s
```

ג.

return (s[:s.find('not')] + 'good' + s[s.find('bad')+3:]) * ('not' in s and 'bad' in s and s.find('not') < s.find('bad')) or s

<u>הרעיון</u>: אם המחרוזת מכילה רצף המתחיל ב-'not' ומסתיים ב-'bad' הוא מוחלף ב-'good', אחרת מוחזרת המחרוזת המקורית כמו שהיא ללא שינוי.

<u>הערה</u>: אפשר גם להוריד את התנאי 'bad' in s, שכן הוא מוכל בבדיקת **s.find('not') < s.find('bad')**, כי אם התנאי מתקיים הרי ש-'bad' במחרוזת. (שהרי אז אינדקס המציאה גדול מ-1-, משמע תת המחרוזת נמצאת במחרוזת).

<u>פתרון אחר נוסף וקצר קצת יותר</u>:

return s.replace(s[s.find("not"):s.find("bad")+3], "good") * ('not' in s and s.find('not') < s.find('bad')) or s

 # G. front_back

לפני הרצה

#

אחרי הרצה

#

```
def front_back(a, b):
  a_middle = len(a) / 2
  b_middle = len(b) / 2
  if len(a) % 2 == 1:  # add 1 if length is odd
    a_middle = a_middle + 1
  if len(b) % 2 == 1:
    b_middle = b_middle + 1
  return a[:a_middle] + b[:b_middle] + a[a_middle:] + b[b_middle:]
```

א. יש לקרוא את קטע הקוד ולהגדיר בשורת 'לפני הרצה' את מטרת הפעולה

ב. יש להריץ את הקטע לכל אחד מהפלטים הבאים ולרשום את הפלט המתקבל:

```
print front_back('abcd', 'xy')
print front_back('abcde', 'xyz')
print front_back('Kitten', 'Donut')
```

ג. יש להשלים את מטרת קטע הקוד (לתקן בהתאמה להגדרה הראשונית).

ד. כתבו את מבנה הפעולה בשורה אחת, בדרך פייתונית:

פתרון:

ב. יש להריץ את הקטע לכל אחד מהפלטים הבאים ולרשום את הפלט המתקבל:

print front_back('abcd', 'xy') abxcdy

print front_back('abcde', 'xyz') abcxydez

print front_back('Kitten', 'Donut') KitDontenut

ג. יש להשלים את מטרת קטע הקוד (לתקן בהתאמה להגדרה הראשונית).

הפעולה מקבלת 2 מחרוזות, ומחזירה מחרוזת המורכבת משרשור החצאים הראשונים של כל מחרוזת ומשרשור החצאים האחרונים של כל מחרוזת. אם אורך מחרוזת אי זוגי, התו האמצעי משויך לחצי הראשון.

ד.

return (((a[:len(a)/2]) * (len(a) % 2 == 0)) or a[:len(a)/2+1]) + (((b[:len(b)/2]) * (len(b) % 2 == 0)) or b[:len(b)/2+1]) + (((a[len(a)/2:]) * (len(a) % 2 == 0)) or a[len(a)/2+1:]) + (((b[len(b)/2:]) * (len(b) % 2 == 0)) or b[len(b)/2+1:])

הרעיון: לשרשר בין חצאים ראשונים של כל מחרוזת לבין חצאים אחרונים של כל מחרוזת, תוך הבחנה בין אורך זוגי לבין אורך אי זוגי של כל מחרוזת, והתחשבות בהתאם בתו האמצעי במחרוזת באורך אי זוגי.

פתרון נוסף (קצר יותר):

```
return a[:(len(a)+1)/2]+b[:(len(b)+1)/2]+a[(len(a)+1)/2:]+b[(len(b)+1)/2:]
```

מערך שיעור - חילוק עם שארית (מודולו %), מר מני עבודי, מר אסף צדיק
דף למורה

מטרות השיעור

1. התלמיד ידע לעשות העברה של פעולת החלוקה בשארית למצבי שארית חלוקה בשלמים אחרים הקיימים בחיי היום-יום.
2. התלמיד יכיר את פעולת החלוקה בשארית (module) בשפת פייתון.
3. התלמיד ידע ליישם את פעולת החלוקה בשארית בשלמים על מגוון בעיות במדעי המחשב, לרבות מחשבון בסיסים וצופן קיסר (צופן הזזה).

הנחות

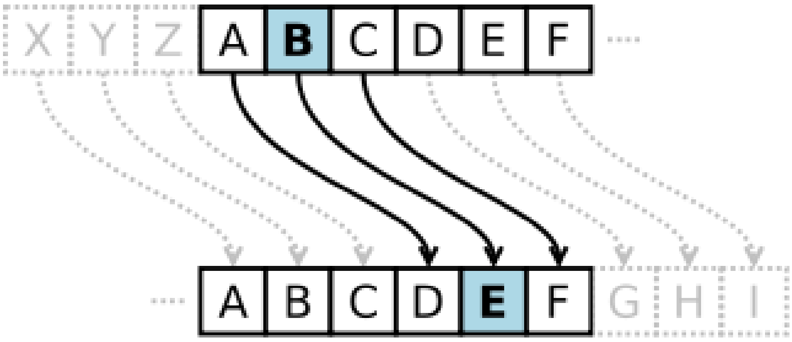
1. התלמיד נחשף לפעולת החלוקה בשארית בלימודיו הקודמים במדעי המחשב (לרבות בתכנית הלימודים בכיתה ז', עם תוכנת ה- scratch ובתכנית הלימודים בכיתה ח', במסגרת תכנות צד לקוח בשפת Java script).
2. התלמיד מכיר פקודות קלט/פלט/השמה בסיסיות בשפת פייתון.

מהו חילוק עם שארית - קישור לסרטון

במתמטיקה:

- שארית (חילוק) – חלק מהמחולק המתקבל לאחר ביצוע פעולת חילוק, אותו לא ניתן לחלק בשנית ולקבל תוצאה שלמה.

צופן קיסר:



הפעולה שמבצע צופן קיסר היא הזזת כל אות מספר מקומות בכיוון נתון לאורך האלף-בית. הדוגמה הזו היא עם היסט (הזזה) בגודל 3 ועם אלף-בית אנגלי. האות B בטקסט הופכת לאות E בטקסט המוצפן.

בתחום הקריפטוגרפיה, **צופן קיסר**, הידוע גם כ**צופן היסט**, **קוד קיסר** או **היסט קיסר** או **הסטה קיסרית**, הוא אחד הצפנים הפשוטים והידועים בעולם ההצפנה. זהו סוג של **צופן** החלפה שבו כל אות בטקסט מוחלפת על ידי אות הנמצאת בהיסט קבוע כלשהו ממנה באלף-

בית. למשל אם נקבע את ההיסט להיות 3, האות A תוחלף באות D, האות B תוחלף באות E וכך הלאה. הכינוי קיסר נובע מכך **שיוליוס קיסר** נהג להשתמש בצופן על מנת לתקשר עם מפקדיו.

צופן קיסר מכונה כך על שם **יוליוס קיסר**, שלפי דבריו של **סווטוניוס**, עשה בו שימוש עם היסט של 3 כדי להסתיר את תוכנן של הודעות צבאיות :

> "אם היה לו מסר מסווג, הוא רשם אותו בקוד, כלומר, על ידי שינוי סדר האותיות של האלף-בית, כך שאף מילה לא הייתה מובנת. אם מישהו חשק לפענח את ההודעות ולהבין את משמעותן, היה עליו להחליף אותיות."

א״ב רגיל ←	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
א״ב מוצפן ←	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

<u>שאלות לדיון</u>:

- מהו גודל ההיסט של הצופן הנ"ל?
- מהו גודל ההיסט של פענוח הצופן הנ"ל?
- האם גודל ההיסט של פעולת ההצפנה ופעולת הפענוח זהים? יש לנמק מדוע.
- האם אופן ההזזה של האות A (לאות D) ואופן ההזזה של האות X (לאות A) זהים? במה הם דומים ובמה הם נבדלים?
- מה הבעייתיות עם ההזזה של האותיות X/Y/Z וכיצד ניתן להתמודד עם בעיה זו?

יש להציג לתלמידים את פעולת החלוקה בשארית (%) פעולת המודולו (mod) במענה לבעיית ההזזה של האותיות X-Z בהינתן גודל היסט 3.
המורה ישכלל את פעולת המודולו בהינתן היסט <u>כלשהו</u>.
המורה ידגים לתלמידים פעולות מודולו בחיי היום יום: חלוקת קוביות שוקולד, שעון אנלוגי, פעולת חיבור בין מספרים שלמים,וכיו"ב.

<u>דגשים</u>:

- כיצד לייצג את המילה המקורית (plain)? רמז: קליטה למחרוזת
- כיצד לייצג את המילה המוצפנת (cipher)? רמז: מחרוזת ריקה
- מה המנגנון להזיז כל אות במילה בגודל קבוע? רמז: לולאה
- כיצד נדע לזהות הזזה "בעייתית"? רמז: כמו הזזה בגודל 3 של האות X?

לאחר מכן, התלמידים יתחלקו לזוגות ויחלו בפתרון האלגוריתם.

דגשים:

- האם גודל ההזזה בפעולת הפענוח זהה לגודל ההזזה בפעולת ההצפנה? רמז: כן. מדובר בהצפנה סימטרית.
- מה המשמעות של הזזה בכיוון "הפוך"? רמז: גודל ההזזה הוא נגדי ולכן פעולת הפענוח היא הפרש ולא סכום.
- כיצד נדע לזהות פענוח "בעייתי" של אות ? רמז: כמו הזזה בגודל 3 של האות A?

הצעה לפתרון

```
__authors__ = "Menny Abudi & Asas Zadik"
import string

# Constants
MODULO = 26
DICTIONARY = string.uppercase

def encrypt(plain, offset):
  cipher = ""
  for letter in plain:
      cipher += str(DICTIONARY[(DICTIONARY.find(letter) + offset) %
MODULO])
  return cipher


def decrypt(cipher, offset):
  plain = ""
  for letter in cipher:
      plain += str(DICTIONARY[(DICTIONARY.find(letter) - offset) %
MODULO])
  return plain


def main():
  plain = raw_input("Enter a word to encrypt: ")
  offset = int(raw_input("Enter the encryption offset: "))
  cipher = encrypt(plain, offset)
  original = decrypt(cipher, offset)
  print "The plain word is ", plain
```

```
print "The encrypted word is ", cipher
print "The decrypted word is ", original


if __name__ == '__main__':
  main()
```

חילוק עם שארית (מודולו %) - דף עבודה לתלמיד

צופן הזזה הוא צופן שבו כל אות מוחלפת באות הנמצאת בהיסט (shift) קבוע בסדר הא״ב.

לדוגמה: עבור shift = 3

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

האות A במסר המקורי תוחלף באות D , האות B תוחלף באות E , האות Z תוחלף באות C , וכך הלאה. לדוגמה: המסר "HELLO " יוצפן ל–"KHOOR".

תהליך הפענוח הפוך לתהליך ההצפנה: כל אות במסר המוצפן תוחלף באות המתאימה על–פי ההיסט שנקבע.

א. לפניכם המילה DOJRULWKP שמוצפנת בצופן הזזה:

ידוע כי בהצפנה השתמשו בהיסט shift = 3. פענחו את המילה הזאת:

ב. אם ערכו של shift לא ידוע (הקיפו את התשובה הנכונה):

1. לא ניתן יהיה לפענח את המילה כי יש אינסוף אפשרויות.
2. ניתן יהיה לפענח את המילה אם עוברים על כל האפשרויות בין 1 ל– 13.
3. ניתן יהיה לפענח את המילה בשיטת "כוח גס"
4. ניתן יהיה לפענח את המילה אם המפתח הציבורי ידוע והמפתח הפרטי לא ידוע.

ג. מחשבים מתרגמים הודעות למספרים, ומבצעים פעולות חשבוניות (חיבור או חיסור) כדי להזיז את האותיות. בטבלה שלפניכם מיוצגות האותיות A – Z על–ידי הקודים המספריים שלהם בבסיס 10 :

A	65	H	72	O	79	V	86
B	66	I	73	P	80	W	87
C	67	J	74	Q	81	X	88
D	68	K	75	R	82	Y	89
E	69	L	76	S	83	Z	90
F	70	M	77	T	84		
G	71	N	78	U	85		

לדוגמה: אם צריך להזיז את A ב– 3 , המחשב פועל באופן הזה:

- הוא ממיר את האות A לערכה המספרי (65)
- הוא מוסיף את המספר 3 ומקבל את המספר 68
- הוא ממיר את ערך הקוד 68 לאות, ומתקבלת האות D

text[k] מסמל את האות שנמצאת במיקום ה– k במילה text . מספר המקומות מתחיל מ– 0 .

לדוגמה: עבור: text = "yesterday"

0	1	2	3	4	5	6	7	8
y	e	s	t	e	r	d	a	y

test[3] = t ו־ text[8] = y

לפניכם אלגוריתם להצפנה של מילה מסוימת בעזרת צופן הזה. השלימו את החסר באלגוריתם הזה:

1. k ← 0.
2. קלוט מילה להצפנה לתוך המשתנה text.
3. קלוט ערך היסט למשתנה shift.
4. הצב במשתנה len את אורך המילה שב– text.
5. כל עוד k > len בצע:

 5.1 הצב במשתנה code את הערך המספרי של התו text[k].

 5.2 ___________ ← new_code.

 5.3 אם _____________

 5.3.1 _____________

 5.4 הצב ב- text[k] את התו המתאים לקוד המספרי (הערך האסקי) של new_code.

 5.5 k ← k + 1.

6. הדפס משתנה text.

ד. כתבו בתוך הטבלה את תהליך ההצפנה עבור המילה: "text" = WAZE" ב– shift = 2 על– פי האלגוריתם הנתון.

k	Code	new_code	Text

<u>דף תשובות למורה:</u>

א. ALGORITHM.

ב. התשובה הנכונה היא 3.

ג.

shift + code

90 < new_code

new_code ← new_code - 26

ד.

k	Code	new_code	Text
0	87	89	YAZE
1	65	67	YCZE
2	90	92 – 26 = 66	YCBE
3	69	71	YCBG

שיעור בנושא מחרוזת string, מר שי שגב

הדרכים להגדרת מחרוזת כש- **my_name** מוגדר כמחרוזת:

גרש בודד	my_name = 'David Ben Gurion'
גרשיים כפולים	my_name = "David Ben Gurion"
באמצעות 3 גרשים בודדים	my_name = '''David Ben Gurion'''
באמצעות 3 גרשיים כפולים	my_name = """David Ben Gurion"""

אין הבדל בין שימוש בגרש בודד לעומת הכפול, אך מקובל להשתמש בגרש בודד.

נשתמש ב- 3 גרשיים לטובת מחרוזת ארוכה, הכוללת שורות ועימוד.

שרשור מחרוזות:

f_name = ‘David ’

l_name = ‘Ben Gurion’

print f_name + l_name   # הדפסת מחרוזת

Output: David Ben Gurion

מציאת אורך מחרוזת:

print len(f_name)

Output: 6   שימו לב לתו הרווח בסוף המילה:

הפונקציה **join** פועלת באופן הבא:

הכנסת מחרוזת נתונה לתוך רצף תווים.

print ‘shai.join(‘cool?’)

Output: cshaioshaioshailshai?

הפונקציה **split** מפצלת מחרוזת כברירת מחדל לפי רווחים או לפי תו נתון:

Print (‘shai cool?’).split()

Output: 'shai', 'cool?'

print ('shay cool?').split('y')

Output: 'sha', ' cool?'

print ('doron cool?').split('o')

Output: 'd', 'r', 'n c', '', 'l?'

<u>פונקציות מחרוזת שונות:</u>

הפונקציה	תיאור
('str1').upper	החזרת המחרוזת באותיות גדולות
('str1').lower	החזרת המחרוזת באותיות קטנות
('str1').join('str2')	שרשור מחרוזות
('str1').split()	פיצול מחרוזת
.startswith('shai')	פונ' בוליאנית הבודקת האם מחרוזת מתחילה בערך נתון
.endswith('cool')	פונ' בוליאנית הבודקת האם מחרוזת מסתיימת בערך נתון
cmp('st1','st2')	השוואה בין 2 מחרוזות: אם המחרוזות זהות יוחזר 0, אם ערך קידוד התאים גדול במחרוזת הראשונה יוחזר 1, אם ערך קידוד התאים גדול במחרוזת השנייה יוחזר 1-,
len('str')	החזרת אורך המחרוזת

דוגמאות:

str1 = 'doron cool'

print str1.replace('doron', 'shai')

Output: shai cool

print (shai cool').upper()

Output: SHAI COOL

print ('alon cool').startswith('alon')

Output: True

משימות תרגול פונקציות

משימת הצפנת מקום המפגש:

יש להוסיף את האות O - לאחר כל אות במשפט המקורי (כולל תו רווח).

לדוגמה, אם המשפט המקורי הוא:

meeting at the cinema

יש להדפיס את המשפט הבא (הדגשת האותיות באדום היא להמחשה בלבד):

moeoeotoionogo oaoto otohoeo ocoionoeomoao

הצעת פתרון

```
str1 = 'meeting at the cinema'
str2 = 'o'.join(str1)
print str2
```

או בשורה בודדת:

```
print 'o'.join('meeting at the cinema')
```

משימת פענוח הצופן:

התקבל טקסט המוצפן הבא: moeoeotoionogo oaoto otohoeo ocoionoeomoa

יש לרשום פקודה שתסיר את כל האותיות O מהטקסט.

הצעת פתרון

```
print ('moeoeotoionogo oaoto otohoeo ocoionoeomoa').split('o')
```

או באופן כללי

```
print (string_text).split('o')
```

Output:

'm', 'e', 'e', 't', 'i', 'n', 'g', ' ', 'a', 't', ' ', 't', 'h', 'e', ' ', 'c', 'i', 'n', 'e', 'm', 'a'

<u>טיפול במחרוזות על ידי חיתוכים ורצפים</u>

בפייתון ישנה אפשרות לסרוק מחרוזת בצורה נוחה על פי מבנה הבא:

String[strart : end : interval]

לדוגמה:

```
str1 = 'meeting at the cinema'
print str1[0:7]
```

Output: meeting

להדפסת כל אות שנייה נשתמש באינטרוול, ניתן להשאיר את מציין הסוף ריק.

```
print str1[0::2]
```

Output: meiga h iea

להדפסת המחרוזת בצורה הפוכה, נשתמש באינטרוול של 1-

```
print str1[::-1]
```

Output: amenic eht ta gniteem

הדפסת המילה הראשונה במחרוזת meeting הפוך

```
print str1[7::-1]
```

Output: gniteem

תרגיל:

קלטו את שמכם המלא (שם פרטי ושם משפחה) למחרוזת my_name ופעלו לפי הוראות **ההדפסה** הבאות:

- הדפיסו את שמכם במלואו
- הדפיסו את שמכם כך שתופיע אות כן ואות לא.
- הדפיסו את שמכם מהסוף להתחלה
- הדפיסו רק את שמכם הפרטי
- הדפיסו רק את שם המשפחה
- הדפיסו בצורה הפוכה את שמכם הפרטי
- הדפיסו בצורה הפוכה את שם המשפחה

- הדפיסו מחרוזת חדשה המורכבת מ-2 אותיות ראשונות בשם הפרטי ו-2 אותיות ראשונות בשם המשפחה
- הדפיסו מחרוזת חדשה המורכבת מ-2 אותיות אחרונות בשם הפרטי ו-2 אותיות אחרונות בשם המשפחה

הצעת פתרון

my_name = ‘shai segev’

print my_name

print my_name[::] או

my_name = 'shai segev'

print my_name[::2]

my_name = 'shai segev'

print my_name[::-1]

my_name = 'shai segev'

print my_name[:4:]

my_name = 'shai segev'

print my_name[::2]

my_name = 'shai segev'

print my_name[5::]

Output: segev

my_name = 'shai segev'

print my_name[0:2] + my_name[5:7]

Output: shse

תרגיל סיסמאות, גב' ריקי יפה, ד"ר דורון זהר

נגדיר סיסמת כניסה חוקית לאתר כסיסמה העומדת בדרישות הבאות:

- אורך 8 תווים לפחות
- מתחילה באות אנגלית (A-z)
- לפחות שני תווים מספריים
- לפחות שני תווים z-A
- כל סימן מיוחד כגון %, $ אסור. מותר מכף תחתון או נקודה.
- אסור רווחים.

כתבו פעולה המקבלת סיסמה ומחזירה True אם הסיסמה עומדת בכל הקריטריונים, אחרת מחזירה False.

אפשרות פתרון ראשונה :

```
PASSWORD_LENGTH = 8

def correct_password(password):

   """ Get password and return True if it stands in standards else return False """

   password = str(password)

   if len(password) < PASSWORD_LENGTH or password[0] < 'A' or password > 'z':

      return False

   count_digits = 0

   count_letters = 0

   for char in password:

      if char >= '0' and char <= '9':

         count_digits += 1

      elif char >= 'A' and char >= 'Z' or char > 'a' and char <= 'z':

         count_letters += 1

      # between z to A there are a few invalid chars

      elif not (char == '_' or char == '.'):
```

```
            return False
    return count_digits > 1 and count_letters > 1
```

אפשרות פתרון שנייה :

```
PASSWORD_LENGTH = 8
def correct_password_2(password):
    """ Get password and return True if it stands in standards  else return False
"""
    password = str(password)
    if len(password) < PASSWORD_LENGTH or password[0] < 'A' or
password[0] > 'z':
        return False
    count_digits, count_letters = 0 , 0
    for char in password:
        if char.isdigit():
            count_digits += 1
        elif char.isalnum():
            count_letters += 1
        # between z to A there are a few invalid chars
        elif not (char == '_' or char == '.'):
            return False
    return count_digits > 1 and count_letters > 1
```

נוריד שני חסמים בבדיקת הסיסמה:

נגדיר סיסמת כניסה חוקית לאתר כסיסמה העומדת בדרישות הבאות:

- אורך 8 תווים לפחות
- מתחילה באות אנגלית (A-z)
- לפחות שני תווים מספריים

- לפחות שני תווים A-z

כתבו פתרון פייתוני קצר.

```
PASSWORD_LENGTH = 8

def correct_password_2(password):

    """ Get password and return True if it is stands in standards else return False
"""

    password = str(password)

    return  = len(password) >= PASSWORD_LENGTH and \

                  ([x for x in password if x.isdigit()]) >= 2 and \

                  len([x for x in password if x.isalpha()]) >= 2
```

תרגיל בניית מחשבון, ד"ר דורון זהר

בדומה לפעולות ב-#C חקרו את מבנה הפעולות/מתודות בפייתון.

להלן הצעת מבנה פעולה:

```
# define functions
def <function_name> (parameter1, parameter2, …):
   """This function ……."""
   …..
   return <result>
```

מה מייצגים החלקים השונים במבנה הפעולה:

```
# ______________________
def ___________________ (____________, _____________, …):
      _____________________
      _____________________
   return ______________
```

תרגיל

במחשבון קיימות הפעולות הבאות:

חיבור, חיסור, כפל, חילוק, חזקה, שורש שלם.

כתבו תכנית המדמה מחשב הקולט שני מספרים שלמים וקוד לכל אחת מהפעולות הבאות:

1. חיבור, **2**. חיסור, **3**. כפל, **4**. חזקה, **5**. שורש שלם

בהתאם לקוד הנקלט תודפס התוצאה.

חלקו את התכנית לתת פעולות.

א. פעולה add המקבלת שני מספרים ומחזירה את סכומם.
ב. פעולה subtract המקבלת שני מספרים ומחזירה את הפרש המספר והמספר השני.
ג. פעולה multiply המקבלת שני מספרים ומחזירה את מכפלתם
ד. פעולה divide המקבלת שני מספרים ומחזירה את התוצאה המדויקת של חלוקת המספר הראשוני במספר השני.
ה. פעולה power המקבלת שני מספרים שלמים x ו-y. על הפעולה להחזיר את x^y (x בחזקת y). בפתרון אין להשתמש במחלקת math.
ו. פעולה sqrt המקבלת מספר שלם ומחזירה את השורש הריבועי השלם שלו. אם למספר אין שורש שלם יוחזר None

יש לשמור על מוסכמות כתיבה והגדרת docstring לכל פעולה.

הצעת פתרון:

```
# Program make a simple calculator that can add, subtract, multiply and divide
using functions

# define functions
def add(x, y):
   """This function adds two numbers"""
   return x + y

def subtract(x, y):
   """This function subtracts two numbers"""
   return x - y

def multiply(x, y):
   """This function multiplies two numbers"""
   return x * y

def divide(x, y):
   """This function divides two numbers"""
   return x / y

# take input from the user
print("Select operation.")
print("1.Add")
print("2.Subtract")
print("3.Multiply")
print("4.Divide")

choice = input("Enter choice(1/2/3/4):")

num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))

if choice == '1':
   print(num1,"+",num2,"=", add(num1,num2))
```

```
elif choice == '2':
   print(num1,"-",num2,"=", subtract(num1,num2))

elif choice == '3':
   print(num1,"*",num2,"=", multiply(num1,num2))

elif choice == '4':
   print(num1,"/",num2,"=", divide(num1,num2))
else:
   print("Invalid input")
```

ממשק – *operations_basses.py*

הקובץ הינו ממשק המכיל פעולות שונות בין בסיסים. להלן נתוני הקובץ המכיל תיעוד והסבר קצר על כל פעולה.

הממשק כולל את ארבעת הפעולות הבאות:

```
# the function gets Strings of binary number, and current base
# returns string of decimal representation according the current base
# if the string is not correctly, the function return empty string
def any_base_to_dec(number, base):
  if checking_input(number, base) and \
      check_digits_in_base(number, base):
    b = int(base)
    digits = number[::-1]
    dec_number, power = 0, 0
    for digit in digits:
      dec_number += int(digit) * (b**power)
      power += 1
    return str(dec_number)
  return ''
```

הפעולה מקבלת מספר ובסיס נוכחי שבו מיוצג המספר. הפעולה תחזיר את הייצוג של המספר ע"פ הבסיס העשרוני. הייצוג העשרוני יוחזר כמחרוזת. במידה והמספר או הבסיס לא תקינים, תוחזר מחרוזת ריקה.

```
# get Strings of decimal number, and other base
# returns string of new representation according the other base
# if the string is not correctly, the function return empty string
def dec_to_any_base(str_number, base):
  if checking_input(str_number, base):
```

```
        b, n = int(base), int(str_number)
        dec_number = ''
        while n != 0:
            dec_number += str(n % b)
            n /= b
        dec_number = dec_number[::-1]
        return dec_number
    return ''
```

הפעולה מקבלת מספר בייצוג דצימלי ובסיס חדש. הפעולה תחזיר את הייצוג של המספר ע"פ הבסיס החדש. הייצוג העשרוני יוחזר כמחרוזת. במידה והמספר או הבסיס לא תקינים, תוחזר מחרוזת ריקה.

```
# get Strings of number, and current base
# returns true if all digits are smaller than the current base
def check_digits_in_base(number, current_base):
    correct = True
    for digit in number:
        d, b = int(digit), int(current_base)
        correct = (d < b) & (d >= 0)
```

הפעולה מקבלת מספר ובסיס נוכחי. הפעולה תחזיר True רק אם כל הספרות של המספר הם בין 0 ל- בסיס. למשל:

עבור המספר 124 בבסיס 3 יוחזר False, כיון שהספרה 4>3, והיא לא יכולה להיות ספרה בבסיס 3.

יוחזר false גם עבור מספרים שליליים.

```
# get Strings of number, and base
# returns true if all characters are digits
def checking_input(str_num, base):
    return (str_num.isdigit())&(base.isdigit())
```

הסבר ופירוט הקובץ – calculator.py

קובץ זה מורכב משני חלקים:

חלק ראשון: הגדרת מחלקה בשם **Calculator** המייצגת אובייקט של מחשבון המרות הכולל את החלון הראשי, פקדים (תכונות המחלקה) ופעולות אריתמטיות שונות שהתבצעו במחשבון.

חלק שני: תכנית ראשית (main). התכנית הראשית **יוצרת** מופע של המחלקה הנ"ל **ומפעילה** בו מתודה הנקראת **run**. בעת **יצירת** המופע, **יופעל בנאי (constructor)** אשר **יצור** את כל הפקדים של המחשבון **ויסדר** אותם על החלון. לאחר מכן נריץ את הפעולה **run** **והמחשבון יוצג** על המסך.

למחלקה **Calculator** **יש** 8 תכונות שהם למעשה אובייקט החלון הגרפי יחד עם כל הפקדים (widgets) שנמצאים על החלון.

שמות התכונות (השדות) הם:

1. ***root*** – אובייקט החלון.

---

2. ***button_number*** – פקד מטיפוס Entry button. זו תיבת טקסט שאליה המשתמש יקיש את המספר הנוכחי.
3. ***button_base*** - פקד מטיפוס Entry button. זו תיבת טקסט שאליה המשתמש יקיש את הבסיס הנוכחי או החדש (תלוי בפעולה שנבחרה ע"י המשתמש).
4. ***button_result*** - פקד מטיפוס Entry button. זו תיבת טקסט שבה יוצג המספר בייצוג הרצוי.

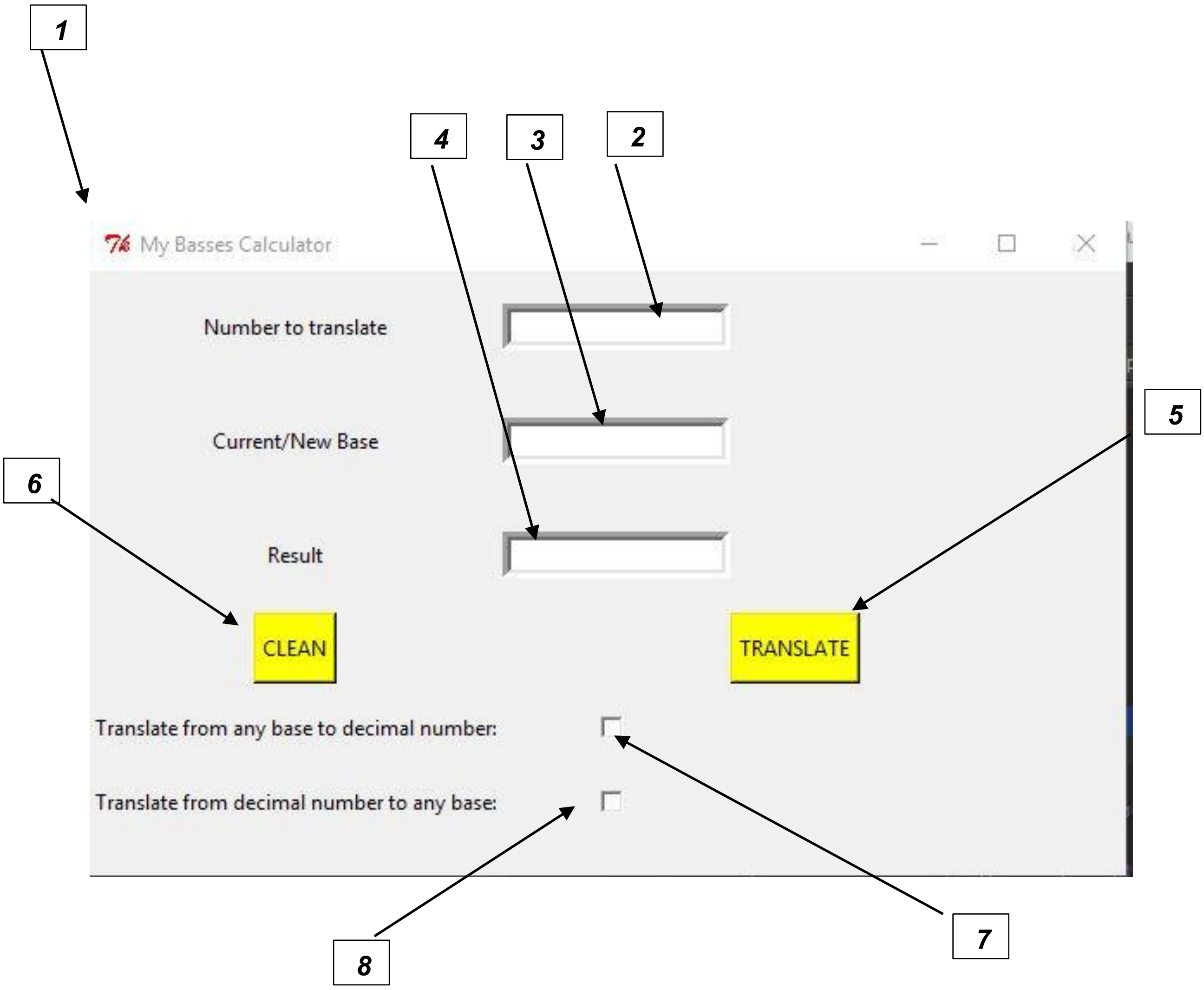
---

5. ***b1*** – לחצן רגיל (מופעל בעזרת לחיצת עכבר) אשר המשתמש ילחץ עליו כאשר ירצה לבצע את פעולת ההמרה (לאחר הכנסת הקלט לתיבות הטקסט).
6. ***b2*** – לחצן רגיל (מופעל בעזרת לחיצת עכבר) אשר המשתמש ילחץ עליו כאשר ירצה לבצע ניקוי של תיבות הטקסט.

---

7. ***checkVar1*** – שדה זה מכיל 0 או 1. 1 במצב בו המשתמש לחץ על האפשרות של המרת מספר מבסיס כל שהוא לבסיס עשרוני. 0, אחרת.
8. ***checkVar2*** – שדה זה מכיל 0 או 1. 1 במצב בו המשתמש לחץ על האפשרות של המרת מספר עשרוני לייצוג בבסיס כל שהוא. 0, אחרת.

להלן תמונה של החלון המחשבון יחד עם כל הפקדים המסומנים עליו ע"פ המספרים שמצוינו בעמוד הקודם:



בעמודים הבאים יובא הקוד של הקובץ הנ"ל יחד עם התעיוד כפי שצריך להיות בפייתון. בנוסף, יובאו הסברים נוספים על אופן בניית המחלקה כפי שהיא ע" עקרונות בסיסיים בתכנות מונחה עצמים.

קוד המחלקה:

```
from Tkinter import*

from operations_basses import*

import tkMessageBox
```

הערה!

אופן הגדרת מחלקה בפייתון קצת שונה מאשר ב- Java כו – C#.

את התכונות יש להגדיר בבנאי עצמו וניגשים אליהם בעזרת המילה השמורה **self**. המילה השמורה **self** מקבילה למילה this שמאפשרת לנו גישה לתכונות ופעולות לאובייקט ב- java.

בשונה מ- this, שב- java, לא ניתן לגשת לתכונות המחלקה לא שימוש ב- **self.**

כאמור, בבנאי שיופיע למטה, אנו נגדיר את התכנות ונאתחל בפקדים השונים.

שורות יצירת הפקדים יסומנו בצהוב.

```
 #this class describe a window of  calculator
class Calculator:
 #    constructor which build the window and all widgets
    def __init__(self):
        self.root = Tk()     # יצירת החלון
        self.root.title("My Basses Calculator")


#       update the length:600 width:350 of current window
        self.root.geometry("600x350")


#       creating all widgets: texts, entry box and buttons
#       create the three texts of the entry buttons
        text = Label(self.root, text="Number to translate", height=4)
        text.grid(row=1, column=1)
        text = Label(self.root, text="Current/New Base", height=4)
```

```
text.grid(row=2, column=1)

text = Label(self.root, text="Result", height=4)

text.grid(row=3, column=1)


#	create entry buttons

self.button_number = Entry(self.root, bd=5)

self.button_number.grid(row=1, column=2)

self.button_number.delete(0, "end")

self.button_base = Entry(self.root, bd=5)

self.button_base.grid(row=2, column=2)

self.button_base.delete(0, "end")

self.button_result = Entry(self.root, bd=5)

self.button_result.delete(0, "end")

self.button_result.grid(row=3, column=2)

#	create the "TRANSLATE" button in the right side of window

self.b1 = Button(self.root, text="TRANSLATE", height=2, bg="yellow",
command=self.translate_number)

self.b1.grid(row=4, column=8)

#	create the "CLEAN" button in the right side of window

self.b2 = Button(self.root, text="CLEAN", height=2, bg="yellow",
command=self.delete_number)

self.b2.grid(row=4, column=1)

text = Label(self.root, text="Translate from any base to decimal number:",
height=3)

text.grid(row=5, column=1)

text = Label(self.root, text="Translate from decimal number to any base:",
height=2)

text.grid(row=6, column=1)

self.checkVar1 = IntVar()

self.checkVar2 = IntVar()
```

```
        c1 = Checkbutton(self.root, variable=self.checkVar1, command=
self.act_checkbox)
        c1.grid(row=5, column=2)
        c2 = Checkbutton(self.root, variable=self.checkVar2, command=
self.act_checkbox)
        c2.grid(row=6, column=2)
```

עבור כל אחד מהשדות נעשה אתחול לפקד מסוים בחלון. יש לשים לב שהטקסטים הם גם פקדים, אך הם לא מהווים תכונות באובייקט החלון וזו מסיבה פשוטה שהם אינם מחזירים ערך אלא פקדים סטטיים שנועדו להסבר עבור המשתמש.

יש לשים לב, שבשורות הבנאים של הפקדים הוכנסו כפרמטרים שמות לפעולות שהוגדרו במחלקה. מבצעים זאת בעזרת המילה השמורה **command**.

בנוסף, אופן השמת הפקדים על גבי החלון נעשה באמצעות הפעולה **grid**. פעולה זו מאפשרת לסדר את הפקדים על גבי החלון בצורת טבלה (תאים ועמודות).

```
    def act_checkbox(self):
        if self.checkVar1.get() == 1:
            self.checkVar2.set(0)
        if self.checkVar2.get() == 1:
            self.checkVar1.set(0)
```

המחשבון מאפשר למשתמש לבחור את סוג ההמרה שיש לבצע.

בעת הגדרת שני הפקדים (ה- **checkbox**), יצרנו שני משתנים אשר משמשים כאינדיקטורים למצבי הפקדים. ניתן לגשת לערכם, וכן ניתן לשנות אותו. בעת שמשנים את ערך המשתנה, זה גורר שינוי מצב של אותו הפקד אליו הוא מצביע. במידה ומשנים את ערכו ל – 0 הפקד יהפוך את מצבו ל – **"לא מסומן"**.

בנוסף, בעת הגדרת הפקד, יש להעביר כפרמטר לבנאי שם של פעולה שתתבצע אוטומטית כל פעם שתתרחש לחיצה כל על הפקד.

הפעולה **act_checkbox** הינה פעולה שהועברה כפרמטר בשורה של יצירת פקדי ה – checkbox (כדאי שתעיינו לעיל).

פעולה זו דואגת שהמשתמש לא יוכל לבחור בשתי האפשרויות בו זמנית (אך יש מצב שהמשתמש לא בחר באף אופציה).

```
#   this function is executed when the user type on "TRANSLATE" button
  def translate_number(self):
     if (self.checkVar1.get() == 0) & (self.checkVar2.get() == 0):
        tkMessageBox.showinfo("Warning:", "You should select an option !!!")
     else:
        self.button_result.delete(0, "end")
        n = self.button_number.get()
        b = self.button_base.get()
        if self.checkVar1.get() == 1:
           r = str(any_base_to_dec(n, b))
        else:
           r = str(dec_to_any_base(n, b))
        self.button_result.insert(0, r)
```

הפעולה ***translate_number*** מתבצעת כל פעם שהמשתמש לחץ על לחצן ה- "TRANSLTE".
שם הפעולה הועבר כפרמטר לשורת זימון הבנאי של פקדי הלחצנים.

פעולה זו מבצעת מספר דברים:

1. בודקת האם המשתמש בחר את אחת האופציות של ההמרות. במידה ולא, תקפוץ הודעת חלון למשתמש על הבעיה, והפעולה לא תמשיך להתבצע.
2. מוחקים את הפלט הקיים על תיבת הטקסט של התוצאה.
3. מחלצים את הקלט מתוך תיבות הטקסט: מספר ובסיס.
4. מבצעים את אחת הפעולות: any_base_to_dec או dec_to_any_base בהתאם למצבי לחצני ה- checkbox.
5. מדפיסים את מחרוזת התוצאה לתיבת הטקסט – result.

הערה! במידה והמשתמש לא הכניס כלל קלט או שהקלט לא תקין, המחרוזת שתתקבל תהייה ריקה. דבר זה מטופל כבר בתוך פעולת ההמרה והבדיקה מהממשק operations_basses.

```
#    this function is executed when the user type on "CLEAN" button
  def delete_number(self):
      self.button_number.delete(0, "end")
      self.button_base.delete(0, "end")
      self.button_result.delete(0, "end")
```

הפעולה ***delete_number*** מתבצעת כל פעם שהמשתמש לחץ על לחצן ה- "CLEAN". שם הפעולה הועבר כפרמטר לשורת זימון הבנאי של פקדי הלחצנים.

הפעולה מוחקת את כל המידה מתיבות הטקסט באמצעות הפעולה delete הקיימת כבר באובייקט של תיבת טקסט.

הערה! הפעולה delete מקבלת שני פרמטרים. הראשון זה אינדקס הראשון שממנו רוצים לבצע את המחיקה. והשני הוא האינדקס האחרון שעד אליו תתבצע המחיקה.

במידה ורוצים למחוק את כל השורה, יש להכניס 0 בפרמטר הראשון, ובפרמטר השני את המחרוזת "end".

```
#    run the window of the calculator
   def run(self):
       self. root.mainloop()
```

הפעולה run עוטפת את הפעולה mainloop ששייכת לאובייקט החלון - ***Tk***. הפעולה ***mainloop*** הינה לולאה שמאפשרת לחלון "להקשיב" לערכי הפקדים הקיימים בו, ובהתאם לכך לאפשר למשתמש לבצע את הפעולות שברצונו לעשות.

```
def main():
# creating a calculate window object
   pass
   obj_calculator = Calculator() \\ יצירת מופע של מחשבון והפעלת בנאי
   obj_calculate.run() \\  הפעלת המחשבון באמצעות הפעולה Run
if __name__ == '__main:'__
   main()
```

דוגמה לשאלות בחינה בפייתון, ד"ר דורון זוהר

<u>**שאלה ראשונה**</u> (20 נק')

ציון חוקי הנו ציון בין 40 ל-100 (כולל). כל ציון שאינו בטווח זה מוגדר כציון שגוי.

לפניך מספר תנאים. מטרת כל תנאי להדפיס Correct אם הציון חוקי, ו- Mistake אחרת.

לכל תנאי יש להקיף בעיגול 'תקין' אם בתנאי יודפס Correct לציון תקין ובכל מקרה אחר שגוי.
לתנאי שגוי יש להסביר בקצרה את הסיבה.

1	if num > 40 and num < 100:
	   print "Correct"
	else:
	   print "Mistake"
		תקין / שגוי
		הסבר:

2	if num > 39 and num <= 100:
	   print "Correct"
	else:
	   print " Mistake"
		תקין / שגוי
		הסבר:

3	if num >= 40 or num <= 100:
	   print "Correct"
	else:
	   print " Mistake"
		תקין / שגוי
		הסבר:

4	if num >= 39 or num <= 100:
	   print "Correct"
	else:
	   print " Mistake"
		תקין / שגוי
		הסבר:

5	if num <= 39 or num > 100:
	   print " Mistake"
	if num > 39 and num < 101:
	   print "Correct"
		תקין / שגוי
		הסבר:

<u>שאלה שנייה</u> (18 נק')

לפניך מספר שורות קוד. לכל שורה יש להציג את הפלט המתקבל:

	שורת קוד	הפלט המתקבל
1	print len("Daniel")	
2	print "Danie".find("a")	
3	print "b" in "Daniel"	
4	print "Daniel".upper()	
5	print "Daniel"[:: -1]	
6	print "ILovePython"[2::2]	

שאלה שלישית (24 נק')

לפניך מספר קטעי קוד המשלבים לולאות. מהו הפלט המתקבל לאחר הרצת כל קוד:

	קוד	פלט
1	for x in range(5): print x	
2	for x in range(1,10,3): print x	
3	for x in range(9, 6, -1): print x	

	קוד	פלט

4	x = 1
	while x < 5:
	    print x**2
	    x = x + 1	

5	x = 769
	new_x = 0
	while x != 0:
	    new_x = new_x * 10 + x % 10
	    x = x / 10
	print new_x	

6	text = "ABCD" for x in text: print chr(ord(x) + 3 % 26)	

<u>שאלה רביעית</u> (20 נק')

לפניך קטע הקוד הבא:

```
num = int(raw_input("Please enter a positive number"))
secret = ""
while (num > 0):
        secret = str(num % 2) + secret
        num = num / 2
print secret
```

א. מה יודפס בעקבות הרצת קטע הקוד עבור הקלט **6**?

____

ב. מה יודפס בעקבות הרצת קטע הקוד עבור הקלט **13**?

____

ג. אריק הריץ את קטע הקוד והקליד מספר כלשהו. בשורת הפלט התקבל המספר 1001.

מה המספר העשרוני אותו הקליד אריק? ______________________________

ד. מה מבצע קטע התכנית. הסבר קצר,

____

____

חלק ב' - יש לבחור את אחת מהשאלות.

<u>שאלה חמישית</u> (18 נק')

שפת Robber היא שפת משחק אשר בה כל אות עיצור מוכפלת וביניהם מתווספת האות o .

לדוגמה: המשפט "have fun" יהפוך ל "hohavove fofunon"

הערה: התו רווח נשאר ללא שינוי

יש לכתוב פעולה המקבלת מחרוזת ומחזירה את המחרוזת המקודדת בשפת Robber .

הדרכה: האותיות היחידות באנגלית שהן לא עיצור הן: aeiou (אותיות תנועה)

לפניך 2 פתרונות חלקיים שכתבו דנה ודני, כל אחד פתר בדרכו שלו. יש להשלים את ההוראות החסרות **באחד** מהפתרונות, של דנה או של דנה.

פתרון א' הפתרון של דנה

def **robber_code**(st):
code = "aeiou" // אותיות תנועה
new_code = ""
for x in ________:
if _______________ or x == " ":
new_code = _______________

```
        else:

            new_code = ________________

    return new_code
```

פתרון ב' הפתרון של דני

```
def robber_code(st):

     code = ['a','e','i','u','o']

    new_code = ""

    for x in ___________:

    if __________________ or x == " ":

            new_code = ____________________

    else:

               new_code = _____________________

     return new_code
```

<u>שאלה שישית</u> (18 נק')

כתובת IP מורכבת מ-4 חלקים (קטטה) המופרדים ע"י נקודה. כל חלק (קטטה) הנו מספר עשרוני בין 0 ל-255 (כולל).

לדוגמה:

192.168.0.17 כתובת IP חוקית

192.890.0.17 כתובת IP לא חוקית

192.890.0.17.89 כתובת IP לא חוקית

192.890 כתובת IP לא חוקית

לפניך פעולה המקבלת מחרוזת המייצגת כתובת IP המורכבת מספרות המופרדות ע"י נקודה. על הפעולה להחזיר True אם הכתובת שהתקבלה חוקית, אחרת False.

בפעולה הושמטו מספר הוראות, יש להשלים את החסר כדי שהפעולה תבצע את מטרתה.

```
def check_IP(ip_add):
```

```
temp = ______________________________
if len(temp) != 4:
    return _______________
for x in temp:
    if _________________________________
        return False
return _______________
```