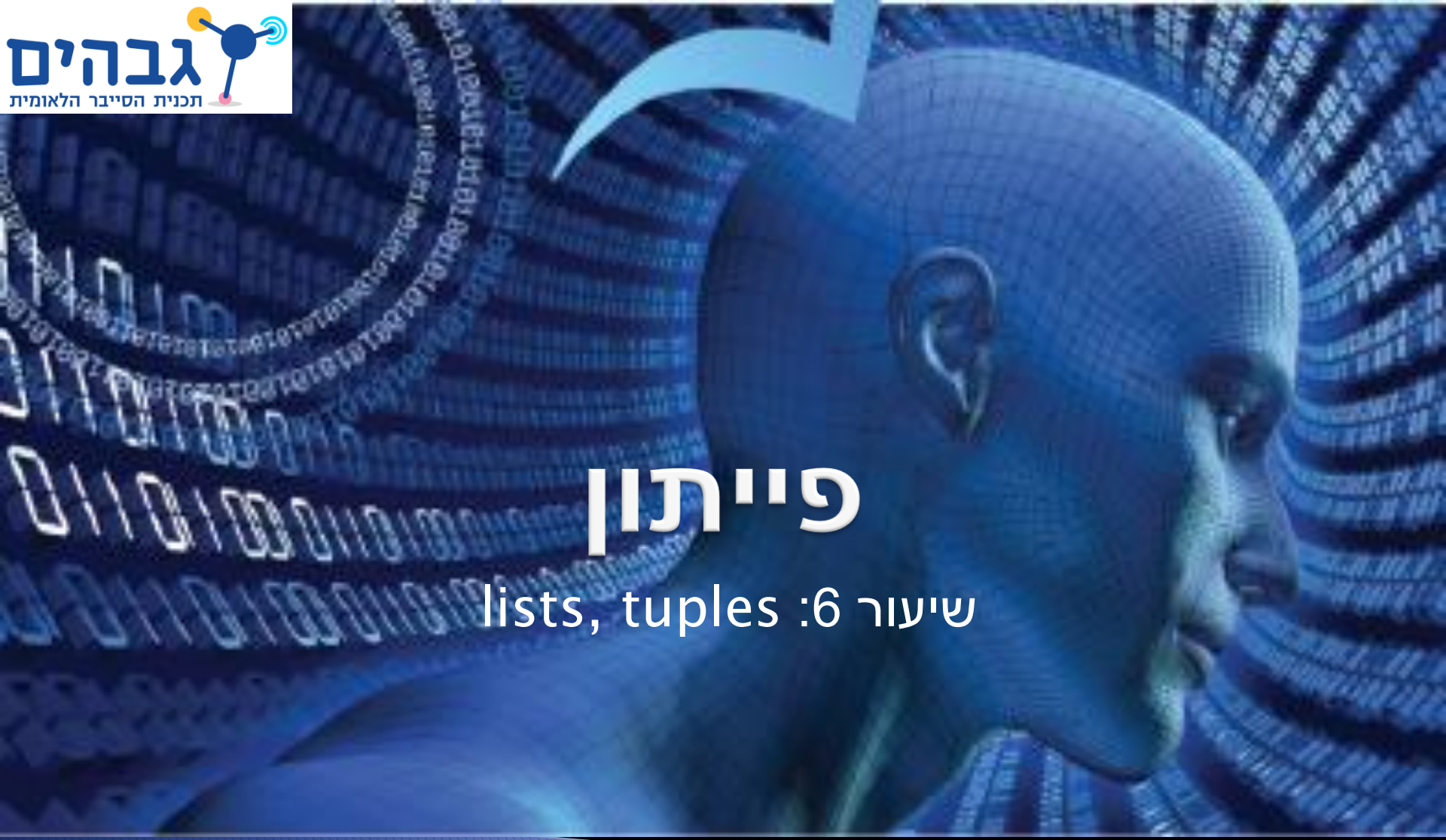
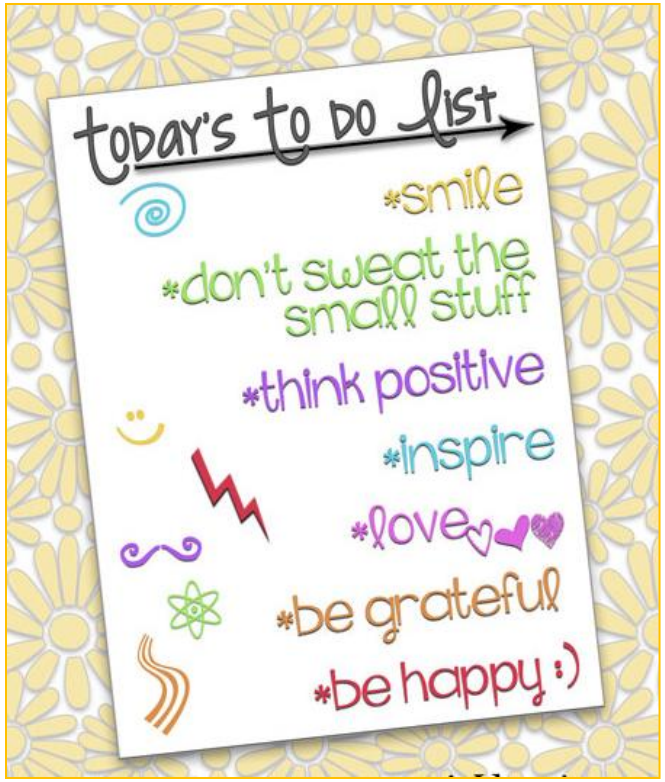


פייתון

שיעור 6: lists, tuples

ברק גונן

list



- list היא אוסף של איברים
  - לא בהכרח מאותו סוג
  - סוגריים מרובעים מציינים list
  - האיברים מופרדים בפסיק

```
stam = [11, 'aaaa', 36.5, True]
```

לולאת for על list

- נניח שאנחנו רוצים להדפיס את כל האיברים ב-list
- כדי לעבור על איברי list בזה אחר זה:
  - המילה **for**
  - שם משתנה כלשהו, שישמש כ-**iterator**
    - עוברים איתו (איטרציה) על איברי ה-list
  - המילה **in**
  - שם ה-list

```
for element in stam:
    print element
```

```
11
aaaa
36.5
True
```

תרגיל - summer

- ישנה פונקציה מובנית בפייתון בשם sum. היא מקבלת רשימה של מספרים ומחזירה את סכום האיברים שברשימה.
- ממשו פונקציה חדשה בשם summer, שגם היא מחזירה את סכום האיברים ברשימה, אך יכולה לפעול על טיפוסים שונים.
  - הערה: הפונקציה לא צריכה לתמוך ברשימה שבה האיברים אינם מאותו type, לדוגמה ['a', 1, 'c'] אינה רשימה חוקית.

```
print summer([10, 11, 12, 0.75])
print summer([True, False, True, True])
print summer(['aa', 'bb', 'cc'])
print summer([[1, 2, 3, 'a'], [4, 'b', 'c', 'd']])
```

```
33.75
3
aabbcc
[1, 2, 3, 'a', 4, 'b', 'c', 'd']
```

- קרדיט: עומר רוזנבוים, שי סדובסקי

Mutable-Immutable

- נבין את המושגים ע"י בחינת ההבדלים בין רשימה למחרוזת
- רשימה - אפשר לשנות ערכים
- מחרוזת – אי אפשר לשנות ערכים

```
>>> my_list = ['a', 'b', 'c']
>>> my_string = 'abc'
>>> my_list[1]
'b'
>>> my_string[1]
'b'

>>> my_list[1] = 'e'
>>> my_list
['a', 'e', 'c']
>>> my_string[1] = 'e'

Traceback (most recent call last):
  File "<pyshell#128>", line 1, in <module>
    my_string[1] = 'e'
TypeError: 'str' object does not support item assignment
```

Mutable – Immutable

- שינוי של רשימה לא משנה את ה-ID שלה
- מחרוזת- יצירת מחרוזת חדשה משנה ID

```
>>> my_str = 'Hello'
>>> id(my_str)
50183344L
>>> my_str = 'World'
>>> id(my_str)
31698456L
```

```
>>> my_list = [1, 2, 3]
>>> id(my_list)
49207944L
>>> my_list[1] = 4
>>> id(my_list)
49207944L
```

- רשימה היא Mutable – ניתנת לשינוי
- מחרוזת היא Immutable- בלתי ניתנת לשינוי

בדיקה אם איבר ב-list

- המילה in
- יחזר True או False
- ניתן לשלב עם פקודת if

```
shopping_list = ['Cheese', 'Melons', 'Oranges', 'Apples', 'Sardines']
if 'Apples' in shopping_list:
    print 'There it is!'
```

append, pop

- append : הוספת איבר ל-list
  - מוסיף את האיבר בסוף ה-list
  - משנה את הרשימה
- pop : הוצאת איבר מ-list
  - מקבל אינדקס כפרמטר (0 הוא האינדקס הראשון)
  - מוציא את האיבר באינדקס הנ"ל

```
>>> stam = [1, 2, 'a']
>>> stam.append('b')
>>> stam
[1, 2, 'a', 'b']

>>> stam.pop(2)
'a'
>>> stam
[1, 2, 'b']
```

sort / sorted

- sort על list מבצע מיון
  - ברירת המחדל: מיון מהקטן לגדול
  - מיון מחרוזות: לפי סדר הופעתן במילון
- אפשר לתת ל-sort פרמטרים
  - reverse
- sorted היא כמו sort, אך יוצרת list חדש

```
>>> stam = [3, 1, 7, 2]
>>> stam.sort()
>>> stam
[1, 2, 3, 7]
>>> stam = ['cat', 'dog', 'apple', 'elephant']
>>> stam.sort()
>>> stam
['apple', 'cat', 'dog', 'elephant']
```

Custom Sorting

- מהם הפרמטרים ש-sort מקבלת?

```
>>> help(stam.sort)
Help on built-in function sort:

sort(...)
    L.sort(cmp=None, key=None, reverse=False)
```

- נקבע reverse=True:

```
>>> stam = [3, 1, 7, 2]
>>> stam.sort(reverse=True)
>>> stam
[7, 3, 2, 1]
```

- נעביר כפרמטר key:

```
>>> words = ['aaa', 'c', 'zz', 'bbb']
>>> words.sort(key=len)
>>> words
['c', 'zz', 'aaa', 'bbb']
```

Custom Sorting – תרגיל מודרך

- אפשר להעביר בתור key שם של פונקציה אותה הגדרנו
- בצעו מיון לפי התו האחרון:
  - הגדירו פונקציה בשם last, שמקבלת מחרוזת ומחזירה את התו האחרון שבמחרוזת
  - קיראו ל-sorted עם הפרמטר key=last

split, join

- פקודת join:
  - מקבלת list מחרוזת כלשהי
  - מחזירה מחרוזת שהיא חיבור האיברים ב-list, בין כל שני איברים מופיעה מחרוזת הקלט
- פקודת split מבצעת פעולה ההפוכה ל-join

```
>>> list_str1 = ['aaa', 'bbb', 'ccc']
>>> list_str2 = ':'.join(list_str1)
>>> list_str2
'aaa:bbb:ccc'
>>> list_str3 = list_str2.split(':')
>>> list_str3
['aaa', 'bbb', 'ccc']
```

תרגיל - פאף

- כיתבו סקריפט שמקבל כפרמטר כתובת אינטרנט ומדפיס את תתי הספריות שלה.
  לדוגמה עבור הקלט
  www.cyber.org.il/networks/class/ex1
  יודפס networks class ex1

tuple

- מוגדר ע"י סוגריים עגולים

```
my tuple = (1, 2, 'a')
```

- tuple הוא immutable
- שימושי להחזרת מספר ערכים מפונקציה ע"י return יחיד

```
def silly():
    return 'hi','there'

first, second = silly()
```

מתי tuple ולא list?

- זיכרון:
  - פייתון מניח שאם הגדרנו list נרצה להגדיל אותה
  - מקצה ל-list זכרון "ספייר"
  - ל-tuple מוקצה זיכרון בדיוק בגודל הנדרש
- מניעת שינויים:
  - נשמור ב-tuple דברים שלא נרצה שהתוכנית תשנה

תרגילים מסכמים

- List1.py
- List2.py