מחרוזות
ב python

אילנית קיי

הגדרת מחרוזת

- מחרוזת: אוסף של תווים כלשהם
  - אפשר להגדיר עם גרש יחיד או גרשיים

```
var1 = 'Hello World'
var2 = "I am programming with Python "
```

מכיוון שהתווים מסודרים ברצף, ניתן להתייחס אל כל אחד מהם באמצעות אינדקס המציין את מרחקם מהתו הראשון, או לחילופין באמצעות אינדקס שלילי המציין את מרחקם מסוף המחרוזת.

לדוגמה עבור המחרוזת הבאה:

```
v = 'Water'
```

אנדקסי המיקום יהיו:

0	1	2	3	4
W	a	t	e	r
-5	-4	-3	-2	-1

ניתן לגשת לתו מסוים על ידי ציון מקומו במחרוזת. לדוגמה:

הוראת הפלט	הפלט שיוצג
print(v[0])	W
print(v[-5])	W
print(v[2])	t
print(v[-2])	e
print(v[3])	e
print(v[-3])	t
print(v[4] + v[1] + v[2])	rat

שרשור מחרוזות באמצעות האופרטור +

- ניתן לשרשר כמה מחרוזות באמצעות האופרטור +

```
1   var1="Hello World "
2   var2="I am programming in Python"
3   print (var1+var2)
4
```

```
Python 3.7.4 (default, Jul  9 2019, 00:
[GCC 6.3.0 20170516] on linux
Hello World I am programming in Python
```

מכפלת מחרוזת באמצעות האופרטור *

- ניתן להכפיל את המחרוזת באמצעות האופרטור *

```
1   var1="Hello World "
2   print (var1*2)
3
```

```
Python 3.7.4 (default, Jul  9
[GCC 6.3.0 20170516] on linux
Hello World Hello World
>
```

הפעולה ord

- החזרת ערך ASCII הדצימלי של התו שבסוגריים

```
1  print (ord('a'))
2
3
```

```
Python 3.7.4 (default, Jul  9
[GCC 6.3.0 20170516] on linux
97
```

- קישור לטבלת ASCII

הפעולה chr

- החזרת התו בעל ערך ה- ASCII הדצימלי שבסוגריים

```
1   print (chr(97))
2
3
```

```
Python 3.7.4
[GCC 6.3.0 20
a
```

- קישור לטבלת ASCII

היפוך מחרוזת באמצעות לולאה ומשתנה עזר מחרוזת ריקה

```
str1=""     #empty string
for letter in "hello":
    str1=letter+str1
print (str1)
```

- פלט:

```
olleh
```

פעולות על מחרוזות ופונקציות מובנות

ניתן להתייחס לקטע (מספר תווים או מילים) בתוך המחרוזת על ידי ציון אינדקס מ- ואינדקס עד (לא כולל), המופרדים בנקודתיים, על פי המבנה הבא:

st [x : y]

שם המחרוזת מ.... עד, לא כולל

לדוגמה:

הסבר	הפלט שיוצג	הוראת הפלט
0 1 2 3 4 W a t e r	at	print (v[1:3])
0 1 2 3 4 W a t e r + f	Waf	s = v[0:2] + 'f'
-5 -4 -3 -2 -1 W a t e r	ate	print (v[-4:-1])
	Wafer	s = v[0:2] + 'f' + v[3:5] print (s)
	WaWa	s = v[0:2] * 2 print (s)

פעולות על מחרוזות

קיימת אפשרות להתייחס לתווים הנמצאים במרחקים קבועים במחרוזת. זאת על ידי הוספת אינדקס דילוג.

st [x : y : z]

שם המחרוזת מ.... קפיצה במרווחים של עד לא כולל

לדוגמה:

הפלט שיוצג	קוד התכנית
	digits = '0123456789'
	even = digits[0:10:2]
	trio = digits[3:10:3]
02468	print (even)
369	print (trio)

היפוך מחרוזת

```
s="HOUSE"
print(s[::-1])
```

- פלט: ESUOH

פעולות על מחרוזות

בטבלה הבאה מופיעה רשימה חלקית של פעולות אפשרויות לביצוע על מחרוזות (חלק מהפעולות הזכרנו בתרגול קודם).

המשתנים s ו-t מייצגים תווים או מחרוזות.

x הוא תו או רצף תווים.

n מייצג מספר חיובי שלם.

k, j ו-i מייצגים מספרים שלמים

הפעולה	תיאור הפעולה
x in s	מחזירה True אם האיבר x (תו או רצף תווים) מוכל בתוך מחרוזת s, בכל מקרה אחר תחזיר False
x not in s	מחזירה False אם האיבר x(תו או רצף תווים) נמצא בתוך מחרוזת s, בכל מקרה אחרת יוחזר True (פעולה הופכית לפעולה הקודמת)
s + t	מחזירה מחרוזת שהיא שרשור של שתי המחרוזות s ו-t
s*n, n*s	לכל n טבעי (שלם גדול מאפס) תוחזר מחרוזת שהיא שרשור של s לעצמו כפול n פעמים. לכל n קטן או שווה לאפס תוחזר מחרוזת ריקה
s[i]	החזרת התו ה-i במחרוזת s (מתחיל מ-0).
s[i:j]	החזרת תת מחרוזת של מחרוזת s ממקום i ועד j לא כולל.
s[i:j:k]	החזרת תת מחרוזת של s ממקום i ועד j לא כולל, בקפיצות של k
s.index(x)	מחזירה מספר שלם שהוא אינדקס המיקום של x (המכיל תו או רצף תווים) במחרוזת s
s.count(x)	מחזירה שלם, שהוא מספר החזרות של x (המכיל תו או רצף תווים) במחרוזת s

בדיקה אם תו או מחרוזת נמצאת/לא נמצאת במחרוזת אחרת

```
if "y" in "python":
  print("y in python")
```

```
if "s" not in "python":
  print ("s is not in python")
```

הפעולה	תיאור הפעולה
x in s	מחזירה True אם האיבר x (תו או רצף תווים) מוכל בתוך מחרוזת s, בכל מקרה אחר תחזיר False
x not in s	מחזירה False אם האיבר x(תו או רצף תווים) נמצא בתוך מחרוזת s, בכל מקרה אחרת יוחזר True (פעולה הופכית לפעולה הקודמת)

str1=input(“enter a string “))
str2= input(“enter another string “))

```
if str2 in str1 or str1 in str2:
  print ("YES")
else:
  print ("NO" )
```

פונקציות מובנות לטיפול במחרוזות

בטבלה הבאה חלק מהפונקציות המובנות ב-python לטיפול בנתונים סדרתיים :

הפונקציה	תיאור הפונקציה
len(s)	מחזירה מספר שלם המייצג את מספר התווים במחרוזת
min(s)	מחזירה את התו בעל ערך ה-ASCII הנמוך ביותר במחרוזת s
max(s)	מחזירה את התו בעל ערך ה-ASCII הגבוה ביותר במחרוזת s
cmp(s,t)	משווה שתי מחרוזות, s ו-t באופן מילוני, ומחזירה : -1 עבור s < t 0 עבור s = t 1 עבור s > t

בגירסה ב בלבד

הפונקציה len מקבלת מחרוזת ומחזירה את אורכה כולל רווחים

```
print (len("hello"))
```

פלט: 5

```
print (len ("hello world"))
```

פלט: 11

הפונקציה join

מכניסה מחרוזת נתונה

בין כל תו של מחרוזת אחרת,

לדוגמא:

```
s = "-"
string = "ilanit"
print(s.join(string))
```

```
i-l-a-n-i-t
```

פלט:

פיצול מחרוזת לרשימה באמצעות הפונקציה split (עפ"י מה שנרשום בסוגריים)

```
string = 'Hello. My name is Ilanit kay.I live in Kfar Sava'
print (string.split('.'))
```

- פלט:

```
['Hello', ' My name is Ilanit kay', 'I live in Kfar Sava']
```

החלפת תת מחרוזת בתת מחרוזת אחרת במחרוזת באמצעות הפונקציה replace

- הפונקציה replace מקבלת 2 מחרוזות הראשונה מה אנחנו רוצים להחליף והשניה במה אנו רוצים להחליף. הפונקציה תחליף את כל המופיעים שיהיו.
- לדוגמא: החלפת המחרוזת day ב night

```
1   string = 'After all, tomorrow is another day!'
2   print (string.replace('day','night') )
```

```
1   After all, tomorrow is another night!
```

דוגמא נוספת: הפונקציה replace

ומה אם רוצים להסיר את הרווחים הפנימיים בין המילים? בשביל זה יש לנו מתודה בשם replace שמקבלת שני פרמטרים – מה אנחנו רוצים להחליף, ובמה אנחנו רוצים להחליף את זה.

```
1   string = 'I think this is the beginning of a beautiful friendship.'
2   print(string.replace(' ',''))
```

```
1   IthinkthisisthebeginningofabeautifulfriendshIp.
```

בדוגמה הזאת ביקשנו מהמתודה **replace** לרוץ על המחרוזת במשתנה string ובכל פעם שהיא נתקלת בתו רווח (' ') להחליף אותו במחרוזת ריקה ('')‎. המשמעות היא בפועל הסרה של כל הרווחים לאלתר. כמובן שנוכל להחליף בצורה הזאת כל תו ואפילו מחרוזות שלמות בתוך המחרוזת המקורית.

שינוי גודל האותיות
מאותיות קטנות לגדולות באמצעות
הפונקציה upper

המתודה **upper** מחזירה כל מחרוזת שתקבל ב-upper case – אותיות גדולות בלבד. אותיות שכבר היו גדולות לא יושפעו.

```
1   string = 'You talking to me?'
2   print(string.upper())
3
```

```
1   YOU TALKING TO ME?
```

שינוי גודל האותיות
מאותיות גדולות לקטנות
באמצעות הפונקציה lower

```
s="HOUSE"
print (s.lower() )
```

- פלט: house

הפונקציה startswith
הפונקציה endswith

- הפונקציות startswith ו- endswith מקבלות מחרוזת כפרמטר ובודקות האם המחרוזת שעליה הופעלו מתחילה או מסתיימת במחרוזת שקיבלו.

```
string = "You can handle the job!"
print (string.startswith('You can'))
print (string.endswith('people'))
```

פלט:
True
False

הפונקציה find

- בודקת אם המחרוזת נמצאת במחרוזת אחרת מחזירה את אינדקס המיקום הראשון שבו הופיעה ובמידה והמחרוזת לא נמצאת יוחזר 1-.

```
str1 = "this is string example....wow!!!"
str2 = "example"

print (str1.find(str2))
```

- פלט: 15

תרגול

לפניך שורת הקוד הבאה:

st = ' StudyingPython '

לכל שורת קוד יש להגדיר את הפלט המתקבל:

	שורות הקוד	פלט
1.	`s = st[::]` `print(s)`	
2.	`s = st[::-1]` `print(s)`	
3.	`s = st[::2]` `print(s)`	
4.	`s = st[::-2]` `print(s)`	
5.	`s = st[:5]` `print(s)`	
6.	`s = st[:-6]` `print(s)`	
7.	`s = st[8:]` `print(s)`	
8.	`s = st[8::2]` `print(s)`	
9.	`s = st[8::-2]` `print(s)`	
10.	`s = st[:9-1]` `print(s)`	

תרגול

לפניך מספר קטעי קוד. לכל קטע יש להגדיר את הפלט:

	שורות הקוד	פלט

1.	`st = 'abcde'`
	`for x in st:`
	`  print(st.index(x))`	

2.	`st = 'abcde'`
	`for x in st:`
	`  print(x, x.upper())`	

3.	`st = 'aBcdE'`
	`for x in st:`
	`  print(x.islower())`	

4.	`st = 'abcde'`
	`for x in st[2:]:`
	`  print(x)`	

5.	`st = 'abcde'`
	`for x in st[::-1]:`
	`  print(x)`	