שפת פייתון – סיכום

סוגי משתנים עיקריים

שלם int
שבר float
בוליאני bool
מחרוזת str
רשימה list

פונקצית type
מחזירה את סוג המשתנה:

```
>>> type(9)
<class 'int'>
>>> type ("AAA")
<class 'str'>
>>> type([0,0])
<class 'list'>
```

שלם int

– עבור מספרים שלמים, חיוביים או שליליים, לא מוגבלים בגודל.
ניתן להזין עשרוני, או בבסיס 2,8,16 עם התחיליות 0b,0o,0x

```
>>> 100,0b101,0o347,0x1f2a
(100, 5, 231, 7978)
```

מעבר מבסיס כלשהו לעשרוני:

```
>>> int('100101',2)
37
>>> int('41024',5)
2639
```

מעבר מעשרוני לבסיס 2,8,16:

```
>>> bin(13)
'0b1101'
>>> oct(99)
'0o143'
>>> hex(876)
'0x36c'
```

שבר float

– מספר המיוצג כשבר עשרוני. מוגבל עד 2^{1024}
מספרים גדולים מאוד / קטנים מאוד מוצגים בתצוגה מדעית:

```
>>> 35000000000000000000.0
3.5e+19
>>> 0.00000000000000012345
1.2345e-16
```

עירבוב שלם ושבר מחזיר שבר:

```
>>> 5 + 2.0
7.0
```

פעולות על מספרים

חיבור
חיסור
כפל
חילוק (מחזיר שבר בכל מקרה)
חזקה
חלוקת שלמים (השארית הולכת לאיבוד)
שארית בחלוקת שלמים

```
>>> 17 + 5
22
>>> 17 - 5
12
>>> 17 * 5
85
>>> 17 / 5
3.4
>>> 17 ** 5
1419857
>>> 17 // 5
3
>>> 17 % 5
2
```

מחרוזת str

– אוסף תווים, תחום בין זוג גרשיים (או גרש בודד).

```
>>> x = 'ABC'
>>> y = "qwerty"
```

בוליאני bool

– מקבל ערכים True או False בלבד.

```
>>> 7>5 , 1+1==3
(True, False)
```

בשילוב פעולות חשבון – True / False הופכים ל – 1 / 0

```
>>> 6 + True
7
>>> (8>7) + (6==6) + (1+1<1)
2
```

המרה בין סוגי משתנים

ממחרוזת לשבר

```
>>> float("6.78")
6.78
```

ממחרוזת לשלם

```
>>> int("123")
123
```

משבר לשלם

```
>>> int(5.67)
5
```

משלם לשבר

```
>>> float(8)
8.0
```

ממספר למחרוזת

```
>>> str(100)
'100'
```

ממחרוזת לרשימה

```
>>> list("ABC")
['A', 'B', 'C']
```

ממספר לבוליאני:
כל מספר חוץ מ-0 הופך True

```
>>> bool(3)
True
>>> bool(-1)
True
>>> bool(0)
False
```

ממחרוזת לבוליאני: כל מחרוזת לא ריקה הופכת ל-True

```
>>> bool("abc")
True
>>> bool("")
False
```

מרשימה לבוליאני: כל רשימה לא ריקה הופכת ל-True

```
>>> bool([1,2])
True
>>> bool([])
False
>>>
>>> bool()
False
```

קלט מהמשתמש – input

פקודת input קולטת נתונים מהמשתמש בזמן הרצת התוכנית. התוכן שיזין המשתמש יישמר תמיד בתור str (ולא כמספר, בוליאני, רשימה וכו').
אם הקלט הרצוי מספרי – יש להמיר אותו עם int או float

```
>>> x=input()
43
>>> x
'43'
```

בתוך הסוגריים ניתן לרשום מחרוזת שתוצג למשתמש לפני קליטת הנתונים:

```
>>> x = int(input("enter a number: ")) ; print(2*x)
enter a number: 100
200
```

הדפסה למסך – print

```
>>> print(6+5)
11
```

ניתן להדפיס באותה פקודה מספר פריטים, וביניהם פסיקים

```
>>> print(6,5,"aaa",True)
6 5 aaa True
```

בין הפריטים יפריד רווח בודד אלא אם כן יוזן sep אחר.

```
>>> print("a","b",100, sep="--")
a--b--100
```

בסוף ההדפסה תודפס שורה חדשה, אלא אם כן יוזן end אחר.

המחרוזת '\n' משמשת להדפסת שורה חדשה.

```
print(1,2,3,4, end=" * ")
print(5,6,7, end = " - ")
print(8)

1 2 3 4 * 5 6 7 - 8
```

תנאי if

לאופציה אחת בלבד:
אם התנאי לא מתקיים, לא מתבצע שום דבר

```
>>> if a>b:
        print("wow")
wow
>>> if a==b:
        print("nice")

>>>
```

יותר משתי אופציות: if, elif, elif.... else. אפשר ללא else, ואז לא מובטח כי משהו יתבצע. בכל מקרה לא יותר מתנאי אחד (הראשון שנכון) יבוצע.

```
>>> x=100
>>> if x > 300: print ("a")
elif x > 200: print ("b")
elif x > 50: print ("c")
else: print("d")

c
```

לשתי אופציות: if, else – מובטח כי בדיוק אחת מהאופציות תתבצע.

```
>>> x = 19
>>> if x > 200:
        print(1)
else:
        print(0)

0
```

שילוב תנאים: not, and, or

```
>>> 5==5 and 6==7
False
>>> 5==5 or 6==7
True
>>> 5==5 and not 6==7
True
```

לולאת for

ניתן לרוץ על איברי רשימה:

```
>>> for i in [3,4,5]:
        print(i**2, end = " ")

9 16 25
```

ניתן לעבור על טווח ערכים – range.
הטווח range(start, end,step) מקבל ערך התחלתי, ערך סופי (לא כולל), ודילוג אפשרי.
כל הפרמטרים ב-range חייבים להיות שלמים.

```
>>> for i in range(3,11,2):
        print(i**2, end = " ")

9 25 49 81
```

לולאה על תווים בתוך מחרוזת

```
>>> for i in "ABCD":
        print(4*i, end = ".")

AAAA.BBBB.CCCC.DDDD.
```

לולאה בתוך לולאה:
a בתוך הלולאה החיצונית ולפני הפנימית
b בתוך שתיהן
c בתוך החיצונית ואחרי הפנימית
d מחוץ לשתיהן

```
a,b,c,d = 0,0,0,0
for i in range(4):
    a = a + 1
    for j in range(5):
        b = b + 1
    c = c + 1
d = d + 1
print(a,b,c,d)

4 20 4 1
```

שימוש במשתנה של הלולאה החיצונית בתוך הפנימית:

```
>>> for i in range(1,5):
        for j in range(i):
            print(j, end=" ")
        print()


0
0 1
0 1 2
0 1 2 3
```

יציאה מלולאה לחלוטין – **break**

```
>>> for i in range(10):
        if i==4:
            break
        print(i, end = " ")

0 1 2 3
```

דילוג למקרה הבא בלולאה – **continue**

```
>>> for i in range(5):
        if i==2:
            continue
        print(i, end = " ")

0 1 3 4
```

לולאת while – מתבצעת כל עוד התנאי נכון

```
>>> while x<20:
        print(x)
        x = x + 3

10
13
16
19
```

בשילוב עם break:

```
>>> while True:
        print(x)
        x = x + 3
        if x > 20: break

10
13
16
19
```

כל מיני פונקציות

ערך מוחלט

מינימום

מקסימום

```
>>> abs(5 - 8)
3
>>> min(4,6,3,8,9)
3
>>> max(1,3,8,4)
8
```

בהינתן תו כלשהו, ord מחזיר את מספרו בטבלת ASCII. בהינתן המספר בטבלת ASCII, chr מחזיר את התו עצמו

```
>>> ord("A")
65
>>> chr(66)
'B'
```

eval הופך מחרוזת לביטוי, ומחשב אותו

```
>>> x="5"
>>> y="*"
>>> z="4"
>>> eval(x+y+z)
20
>>>
>>> q="4>1"
>>> print(q)
4>1
>>> print(eval(q))
True
```

ייבוא מודולים (ספריות) – import

ייבוא מאפשר שימוש בפונקציות חיצוניות ובקוד שנכתב בנפרד, בקובץ אחר מזה שעובדים עליו.
אפשר ליצור מודולים לבד, לשמור אותם ולהשתמש בקוד בעתיד, או לייבא מודול (ספריה) מההרחבות של פייתון.
לייבוא 3 אפשרויות עיקריות:

ייבוא המודול בלי לפרט את תוכנו. שימוש בפונקציות בתוכו מחייב לקרוא למודול בנוסף לפונקציה:

```
>>> import math
>>> math.pi
3.141592653589793
>>> math.sqrt(5)
2.23606797749979
```

ייבוא פונקציה מסוימת מתוך המודול. ניתן לקרוא לה ישירות לאחר מכן:

```
>>> from math import log10
>>> log10(1000)
3.0
```

ייבוא כל תוכן המודול (לרוב לא מומלץ) בעזרת כוכבית. כל התוכן ניתן לגישה ישירה:

```
>>> from math import *
>>> e
2.718281828459045
>>> pi
3.141592653589793
>>> cos(0)
1.0
```

פעולות על מחרוזות

חיבור שתי מחרוזות ייצור מחרוזת ארוכה מהצמדת שתיהן.

```
>>> "abc" + "123"
'abc123'
```

כפל מחרוזת במספר שלם ישכפל את המחרוזת.

```
>>> "ab " * 4
'ab ab ab ab '
```

in – יחזיר בוליאני: האם מחרוזת אחת נמצאת בתוך מחרוזת אחרת

```
>>> "5" in "4567"
True
```

len – אורך המחרוזת

```
>>> len("abcde")
5
```

for – יעבור על כל אחד מהתווים במחרוזת

```
>>> for char in "abc":
        print(5*char, end=" ")

aaaaa bbbbb ccccc
```

אינדקס [ ] – תו במיקום מסוים בתוך מחרוזת.
מיקומים מתחילים מאפס.
מיקומים שליליים נספרים מהסוף, כאשר מיקום 1- הוא התו האחרון.

```
>>> x = 'AaBbCcDdEe'
>>> x[0]
'A'
>>> x[3]
'b'
>>> x[-1]
'e'
>>> x[-10]
'A'
```

פריסה [start :end :step] – החל מתו במיקום start, עד ולא כולל מיקום end, בדילוג step.

```
>>> x="abcdefgh"
>>> x[2:4]
'cd'
>>> x[3:]
'defgh'
>>> x[:3]
'abc'
>>> x[::2]
'aceg'
>>> x[5:1:-1]
'fedc'
>>> x[::-1]
'hgfedcba'
```

אותיות גדולות / קטנות

```
>>> x = "Aaa bbb cCCc defgh  "
>>> x.upper()
'AAA BBB CCCC DEFGH  '
>>> x.lower()
'aaa bbb cccc defgh  '
>>> x.title()
'Aaa Bbb Cccc Defgh  '
>>> x.capitalize()
'Aaa bbb cccc defgh  '
>>> x.swapcase()
'aAA BBB CccC DEFGH  '
```

ניקוי רווחים משני הצדדים

```
>>> x.strip()
'Aaa bbb cCCc defgh'
```

פיצול לרשימת מילים (לפי רווחים)

```
>>> x.split()
['Aaa', 'bbb', 'cCCc', 'defgh']
```

פיצול לפי כל תו אחר מלבד רווח

```
>>> "aaabccbdbe".split("b")
['aaa', 'cc', 'd', 'e']
```

ארגון מחרוזת לפי מקום מדוד (מרכוז / יישור לשמאל / יישור לימין / מילוי באפסים)

```
>>> "abcd".center(10)
'   abcd   '
>>> "abcd".ljust(10)
'abcd      '
>>> "abcd".rjust(10)
'      abcd'
>>> "abcd".zfill(10)
'000000abcd'
```

הצמדת מחרוזות למחרוזת אחת: פונקציית join מקבלת רשימה של מחרוזות, או מחרוזת בודדת.

```
>>> "##".join(["1","2","3"])
'1##2##3'
>>> "-".join("abcde")
'a-b-c-d-e'
```

find – מחזיר את המיקום של המופע הראשון של תת-מחרוזת במחרוזת כלשהי. מחזיר 1- אם היא לא נמצאה.

```
>>> "abcdabcdefefef".find('e')
8
>>> "abcdabcdefefef".find('X')
-1
```

replace – החלפת כל המופעים של תת-מחרוזת באחרת

```
>>> "abcabdabe".replace("ab","1")
'1c1d1e'
```

count – סופר כמה פעמים מופיעה תת-מחרוזת בתוך מחרוזת כלשהי:

```
>>> "Good Morning!".count("o")
3
```

פעולות על רשימות

רשימות יכולות לכלול איברים מאותו סוג, מסוגים שונים, ואפילו רשימות מותרות בתור איברים.

```
L1 = [2, 3, 5, 7, 11]
L2 = [6, 7.5, "A", True]
L3 = [[4,6],[3,1,0],[11]]
```

חיבור שתי רשימות ייצור רשימה ארוכה מהצמדת שתיהן

```
>>> [1,2,3]+["a","b"]
[1, 2, 3, 'a', 'b']
```

כפל רשימה במספר שלם ישכפל את הרשימה

```
>>> [0]*8
[0, 0, 0, 0, 0, 0, 0, 0]
```

for – יעבור על כל אחד מהאיברים ברשימה

```
>>> for i in ["good","day"]:
        print(i[0])

g
d
```

in – יחזיר בוליאני: האם איבר כלשהו נמצא בתוך הרשימה

```
>>> 6 in [3,5,7]
False
>>> "a" in ["ab","cd"]
False
>>> 2 in [1,2,3]
True
>>> "5" in [4,5,6]
False
```

len – אורך הרשימה

```
>>> L=[1,2,3,"a","b"]
>>> len(L)
5
```

הוספת איבר לרשימה: על-ידי חיבור, או append

```
>>> L=[1,6,2]
>>> L = L + [100]
>>> L.append(9)
>>> L
[1, 6, 2, 100, 9]
```

אינדקס [ ] –איבר במיקום מסויים בתוך רשימה. מיקומים מתחילים מאפס. מיקומים שליליים נספרים מהסוף, כאשר מיקום 1- הוא האיבר האחרון. ברשימה בתוך רשימה, או מחרוזת בתוך רשימה, משרשרים את האינדקסים.

```
>>> L = [23,500,[5,7,9],"ABCD"]
>>> L[1]
500
>>> L[2][0]
5
>>> L[-1][-2]
'C'
```

הסרת איבר – remove

```
>>> L = [4,7,10,13]
>>> L.remove(10)
>>> L
[4, 7, 13]
```

הסרת איבר על-פי מיקומו – del

```
>>> L = [4,7,10,13]
>>> del L[1]
>>> L
[4, 10, 13]
```

פריסה [start :end :step] – החל מאיבר במיקום start, עד ולא כולל מיקום end, בדילוג step.

```
>>> L = ["abcd", 5, 6, 7, [8,9,10,11]]
>>> L[1:3]
[5, 6]
>>> L[1::2]
[5, 7]
>>> L[0][1:3]
'bc'
>>> L[-1][::-1]
[11, 10, 9, 8]
```

מיון הפריטים ברשימה –

sorted מחזיר רשימה ממוינת (אבל משאיר את המקורית כפי שהייתה)

```
>>> L = [6,2,9,3,88,1,44]
>>> print(sorted(L))
[1, 2, 3, 6, 9, 44, 88]
>>> print(L)
[6, 2, 9, 3, 88, 1, 44]
```

sort הופך את הרשימה המקורית לממוינת

```
>>> L = [6,2,9,3,88,1,44]
>>> L.sort()
>>> L
[1, 2, 3, 6, 9, 44, 88]
```

ספירה – count מחזיר כמה פעמים איבר מופיע ברשימה

```
>>> L=[1,2,1,3,1,4,1,5]
>>> L.count(1)
4
```

חיפוש – index מחזיר מהו המיקום הראשון בו מופיע איבר כלשהו

```
>>> L=[4,4,5,5,2,4,5]
>>> L.index(2)
4
```

הגדרת פונקציות

def להגדרת פונקציה

```
>>> def f():
        for i in range(3):
                print(5*"X")


>>> f()
XXXXX
XXXXX
XXXXX
```

פונקציה קוראת לפונקציה אחרת

```
>>> def surround(text):
        l=len(text)
        print((l+2)*"*")
        print("*"+text+"*")
        print((l+2)*"*")


>>> def abb(sentence):
        s=""
        words=sentence.split()
        for word in words:
                s = s + word[0]
        return s

>>> surround(abb("Abc def ghij klm"))
******
*Adgk*
******
```

פרמטרים שרוצים שהפונקציה תקבל, רושמים בסוגריים בעת הגדרתה. כאשר קוראים לה, יש לספק את הפרמטרים הדרושים

```
>>> def f(a,b):
        sum2 = a*a + b*b
        print(sum2)


>>> f(4,5)
41
```

לפרמטרים ניתן לספק ערכי ברירת מחדל. במקרה זה לא חייבים לספק אותם כאשר קוראים לפונקציה.

```
>>> def rectangle(a=2,b=4):
        for i in range(a):
                print(b*"X")


>>> rectangle()
XXXX
XXXX
>>> rectangle(3,7)
XXXXXXX
XXXXXXX
XXXXXXX
```

return - עבור ערך שהפונקציה מחזירה

```
>>> def sumSfarot(n):
        s = 0
        for i in str(n):
                s = s + int(i)
        return s

>>> sumSfarot(2345)
14
>>> sumSfarot(8923)+sumSfarot(1112)
27
```

ניתן להחזיר יותר מערך אחד – ביניהם פסיקים

```
>>> def parabolaMinMax(a,b,c):
        x=-b/(2*a)
        y=a*x*x+b*x+c
        return x,y

>>> parabolaMinMax(1,4,11)
(-2.0, 7.0)
```