פרק 4 אופרטורים וקלט / פלט מהמשתמש

בפרק 4 בתוכנית הלימודים של הנדסאי אלקטרוניקה של משרד החינוך בנושא שפת פייתון רשום :

פרק 4 : אופרטורים

יעדים

היכרות עם מערך האופרטים בשפת Python.

תכנים

1. אופרטורים בסיסים
2. ההבדל בין אופרטור אונארי לבינארי
3. אופרטור קידום/הפחתה עצמיים
4. הכפלה/חלוקה ושארית החלוקה
5. אופרטורי השוואה
6. אופרטורי השמה
7. ערך רצפה, ערך גג ועיגול :
   - floor
   - ceil
   - round
8. האופרטור "is"
9. אופרטורים על סיביות (Bits Wise)

סיכום שעות ההוראה : 8 שעות עיוניות ו- 4 שעות התנסותיות. סה"כ 12 שעות.

4. אופרטורים : פעולות אריתמטיות, לוגיות, השמה והשוואה

בתחילת הפרק נגדיר מושגים שנשתמש בהם בהמשך.

iterable טיפוס של נתונים שניתן לעבור עליו לפי סדר כמו range או while או for . כלומר, ניתן לעבור בלולאה אובייקט אחרי אובייקט . דוגמאות : list, tuple, dict, set, file, range .

איטרטור (iterator) הוא האובייקט איתו עוברים על איברי (הפריטים) של טיפוס איטרלי של נתונים או לחילופין האובייקט שמחזיר ("או ממנו מקבלים") בכל פעם את האיבר הבא בתור.

דוגמה :

```
mylist=[3,-5,6,10]
for number in mylist:
    print(number)
```

mylist היא רשימה והיא טיפוס נתונים iterable . number הוא האיטרטור איתו עוברים על הפריטים ברשימה.

האופרטורים – פעולות - משמשים לביצוע פעולות בין משתנים וערכים.

האופרטורים בפייתון מחולקים לקבוצות הבאות :

- **פעולות אריתמטיות (חשבוניות) .**
- **פעולות לוגיות .**
- **פעולות השוואה.**
- **פעולות זהות.**
- **פעולות השמה.**
- **פעולות על חברויות.**

4.1 אופרטורים אריתמטיים (חשבוניים)

בטבלה הבאה נניח ש x=9 ו y=4

הסבר	דוגמה	השם	האופרטור
חיבור x+y	x+y = 9 + 4 = 13	חיבור	+
חיסור x-y	x-y = 9 - 4 = 5	חיסור	-
כפל x*y	x*y = 9 * 4 = 36	כפל	*
חלוקה x/y	x/y = 9 / 4=2.25	חלוקה	/
שארית לאחר חלוקה של x/y . תוצאת החלוקה השלמה היא 2 ונשאר שארית 1 .	x%y = 9 % 4 = 1	שארית (מודולו)	%
חזקה של x^y	x**y = 9^4 = 6561	חזקה או שורש	**
המספר השלם הקרוב לרצפה של החלוקה x/y . התוצאה "האמיתית" היא 2.25 והמספר השלם הקרוב לרצפה הוא 2. בחלוקה של מספרים שליליים כמו 9/2- התוצאה "האמיתית" היא 2.25- והמספר השלם הקרוב לרצפה היא 3- כי זה המספר שהוא כמו רצפה ל 2.25- .	x//y = 9//4=2 -9//4 = -3	חילוק בתצוגת מנה שלמה - (Floored division)	//

טבלה 1 : פעולות אריתמטיות

הערה לנושא החזקה : ניתן להשתמש באופרטור חזקה ** גם למציאת שורש אם נשתמש בחזקה כשבר.

דוגמה : מצא את השורש של המספר 45 .

```
>>> 45**0.5
6.708203932499369
```

דוגמה נוספת : נמצא את השורש הרביעי של 6561 שהוא תוצאה של 9^4 .

```
>>> 6561**0.25
9.0
```

הערה לנושא האחרון בטבלה שהוא חילוק בתצוגת מנה שלמה // : הרצפה של כל מספר חיובי היא הערך השלם של המספר השלם שמתחתיו שמשמש לו כרצפה, כלומר ללא החלק של השבר. לדוגמה : הריצפה של 1.001 היא 1 וגם של 1.999 היא 1 . עבור מספרים שליליים הריצפה היא המספר השלילי השלם שמתחתיו. לדוגמה : 1.001- היא 2- וגם של 1.999- היא 2- .

4.2 אופרטורים לוגיים

אופרטורים לוגיים משמשים להצהרות עם תנאי . אם התנאי מתקיים מקבלים True אם התנאי לא מתקיים מוחזר False .
בטבלה הבאה נניח ש x=9 ו y=4

הסבר	דוגמה	השם	האופרטור
כאן מתקיימים 2 התנאים.	x < 10 and y >3 נקבל True	טבלת and . וגם אמת F and F=F F and T=F T and F=F T and T=T	and
כאן התנאי הראשון הוא True והשני False והתוצאה True	x < 10 or y >10 נקבל True	או or F or F=F F or T=T T or F=T T or T=T	or
תוצאת הסוגריים היא True ואחרי היפוך היא False	not(x < 10 and y <10) נקבל False	היפוך not	not

טבלה 2 : אופרטורים לוגיים

4.3 אופרטורים להשוואה

אופרטורים להשוואה משמשים להשוואה בין ערכים. התוצאה המוחזרת היא True או False .
בטבלה הבאה נניח ש x=9 ו y=4

הסבר	דוגמה	השם	האופרטור

היות ו x לא שווה ל y נקבל False	print(x==y) בודקים האם x שווה ל y ? . נקבל False .	האם שווה ?	= =
היות ו x לא שווה ל y נקבל True	if (x!=y) : print("x != y") בודקים האם x לא שווה ל y . נקבל True ואת ההדפסה x!=y	האם לא שווה ?	!=
אם x גדול מ y מוחזר True ולכן נקבל את ההדפסה x bigger than y .	if (x>y) : print("x bigger than y")	גדול מ ?	>
מודפס False כי x לא קטן מ y .	print(x<y) נקבל False	קטן מ ?	<
	print(x>y) נקבל True	גדול או שווה ?	>=
	print(x<=y) נקבל False	קטן או שווה ?	<=

טבלה 3 : אופרטורים להשוואה

4.4 פעולות לבדיקת זהות

משתמשים בפעולות לבדיקת זהות כדי לבדוק האם האובייקט הוא אותו אובייקט ונמצא באותה כתובת בזיכרון.

הפעולה לא בודקת האם הערך שיש בשני האובייקטים הוא שווה !

יש 2 אופרטורים : **is** ו **is not**

האופרטור is בודק האם 2 משתנים הם אותו אובייקט . אם כן מוחזר True . אם הם לא אותו אובייקט מוחזר False .

הפעולה is not עובדת הפוך. מוחזר True אם הם לא אותו האובייקט ו False אם הם אותו אובייקט.

דוגמה:

```
x = ["dollar", "euro"]
y = ["dollar", "euro"]
z = x

print(x is z)
```

True # בגלל שהם אותו אובייקט והם באותן כתובות בזיכרון
נקבל :

print(x is y)
False # אפילו שערכם שווה הם לא אותו האובייקט ולא נמצאים באותו כתובות בזיכרון
נקבל :

print(x == y)
True # כי הערכים שלהם שווים
נקבל :

4.5 פעולות על ביטים

פעולות על ביטים משמשות להשוואה בינארית של מספרים. בטבלה הבאה נניח ש x=9 ו y=4

הסבר	דוגמה	השם	האופרטור
x=1001 y=0100 → 0	x & y → 0	AND	&
x=1001 y=0100 →13	x | y → 13	OR	|
x=1001 y=0100 →13	x ^ y → 13	XOR	^
not(x ^ y) → not(13) = False	not(x^y) → False	NOT	~
9<<1 →18 <<1 → 36	x<<2 →36	הזזה שמאלית עם מילוי אפסים	<<
9>>1 →4 >>1 →2	x>>2 → 2	הזזה ימנית עם שמירה על הסימן	>>

טבלה 4 : פעולות על ביטים

4.6 פעולות השמה

הסבר (הערה – במקום שרשום b הכוונה ערך בינארי)	דוגמה	הפעולה
	x=9	=
	x+=4 x=x+4 = 13	+=
	x-=4 x=x-4 = 9-4 = 5	-=
	x*=4 x=x*4=9*4 = 36	*=
	x/=4 x=x/4=9/4 = 2.25	/=
	x%=4 x=x%4 = 9%4=1	%=

//=	x//4 x=x//4 = 9//4 = 2	
**=	x**=4 x=x**4 = 9^4= 6561	
&=	x&=y x=x&y = 9&4=0	9 & 4= b1001 & b0100 = b0000 =0
|=	x|=y x==x|y = 9|4=13	9 | 4= b1001 | b0100 = b1101 =13
^=	x^=3 x=x^3=9^3=10	9 ^ 3= b1001 ^ b0011 = b10010 =10
>>=	x>>=3 x>>3=9>>3=1	הזזה ימנית מחלקת את הערך ב 2 תוך שמירת הסימן ומזניחה שארית.
<<=	x<<=3 x=x<<3=9<<3=72	הזזה שמאלית מכפילה את המספר ב 2 .

אופרטורים של השמה משמש להעביר ערכים למשתנים. בטבלה הבאה נניח ש x=9 ו y=4

טבלה 5 : פעולות השמה

4.7 פעולות חברות - Membership Operators

משתמשים בפעולות חברות כדי לבדוק האם רצף מסוים נמצא באובייקט.

יש שני אופרטורים. א. **in** ב. **not in**

לדוגמה:

```
colors = ["Red", "Green", "Blue"]
print("Green" in colors)
```

התשובה שנקבל היא True .

4.8 המתודות floor( ) Ceil( ) ו round( )

שלוש המתודות נמצאות במודול math של פייתון ולפני השימוש בהן יש לרשום **import math** .

4.8.א המתודה floor( ) מקבלת מספר כלשהו ומחזירה את המספר השלם הקרוב כלפי מטה למספר ששלחנו אליה.

דוגמה :

```
print(math.floor(0.123))        #   0
print(math.floor(1.3456))       #   1
print(math.floor(15.021))       #   15
print(math.floor(-5.3))         #  -6
print(math.floor(-20.654))      #  -21
print(math.floor(10.0))         #   10
print(math.floor(1.9999))       #   1
```

4.8.ב המתודה ceil( ) מקבלת מספר כלשהו ומחזירה את המספר השלם הקרוב כלפי מעלה למספר ששלחנו אליה.

```
print(math.ceil(0.123))          #   1
print(math.ceil(1.3456))         #   2
print(math.ceil(15.021))         #   16
print(math.ceil(-5.3))           #   -5
print(math.ceil(-20.654)) #   -20
print(math.ceil(10.0))           #   10
print(math.ceil(1.9999))         #   2
```

4.8.ג המתודה round( ) מקבלת 2 ערכים. הראשון הוא מספר והשני הוא כמה ספרות אחרי הנקודה רוצים. היאמחזירה את המספר הממשי המעוגל (נקודה צפה – floating point) למספר ששלחנו אליה עם מספר הספרות הרצוי אחרי הנקודה. ברירת המחדל למספר הספרות הוא 0 (אם לא שלחנו אליה את מספר הספרות אחרי הנקודה שרוצים) ואז מוחזר המספר השלם .

דוגמאות :

```
print(round(5.76543, 2))          #  5.77
print(round(3.12345,3))           # 3.123
print(round(-12.654321,4))        # -12.6543
print(round(-123,456))            # -123
```

4.9 המרות – casting (ניתן גם לקרוא ליהוק)

ניתן להמיר משתנה מטיפוס כלשהו לטיפוס אחר . מושג כזה נקרא המרה או ליהוק - casting . המושג ליהוק בא מהכוונה שנותנים תפקיד חדש למשתנה (כמו בהצגה או סרט שבהם מלהקים אנשים לתפקיד כלשהו). המרה בפייתון נעשית בעזרת פונקציות בנאי.

4.9.1 המרה ל int

בנייה של מספר שלם ממספר שלם ליטרלי (מילולי) , ממשתנה ממשי float או ממחרוזת.

דוגמאות:

```
x1 = 12
x = int(x1)
y1 = 3.14
y = int(y1)
z1 = "123"
z = int(z1)
```

```
w1 = "2A"
w = int (w1,16)  # העברה מבסיס הקסה לעשרוני
v1=True
v = int(v1)
print("type of v1 is : ", type(w1), "and w = ",w)
print("type of x1 is : ", type(x1), "and x = ",x)
print("type of y1 is : ", type(y1), "and y = ",y)
print("type of z1 is : ", type(z1), "and z = ",z)
print("type of w1 is : ", type(w1), "and w = ",w)
print("type of v1 is : ", type(v1), "and v = ",v)
```

ההדפסות שנקבל :

type of v1 is : <class 'str'> and w = 42
type of x1 is : <class 'int'> and x = 12
type of y1 is : <class 'float'> and y = 3
type of z1 is : <class 'str'> and z = 123
type of w1 is : <class 'str'> and w = 42
type of v1 is : <class 'bool'> and v = 1

4.9.2 המרה ל float

בנייה של מספר ממשי float ממספר שלם ליטרלי או ממשי ליטרלי או מחרוזת .

דוגמאות:

```
x = float(1)
y = float(1.234)
z = float("1234")
w = float("1.234")
v = float (True)
print(x)
print(y)
print(z)
print(w)
print(v)
```

ההדפסות שנקבל :

1.0

1.234

1234.0

1.234

1.0

4.9.3 המרה ל str

בנייה של מחרוזת ממגוון נתונים רחב כולל מחרוזות, ליטרליים שלמים וליטרליים ממשיים.

דוגמאות :

```
x = str("123")
y = str(123)
z = str(3.456)
print("x = ",x , "and type of x : ", type(x))
print("y = ",y , "and type of y : ", type(y))
print("z = ",z , "and type of z : ", type(z))
```

ההדפסות שנקבל :

```
x =  123 and type of x :  <class 'str'>
y =  123 and type of y :  <class 'str'>
z =  3.456 and type of z :  <class 'str'>
```

4.9.4 המרה ל bool

דוגמאות :

```
print(bool(0))
print(bool("0"))    #  "0" = 0x30 ASCII
print(bool (1.234))
```

ההדפסות שנקבל :

```
False
True
True
```

4.9.5 המרה לתו chr

המרה של ערך מספרי לקוד ה unicode שלו.

```
print(chr(0x30))
print(chr(48))
print(chr(65))
print(chr(1488))
print(chr(1489))
```

```
print(chr(1490))
```

ההדפסות שנקבל :

```
0
0
A
א
ב
ג
```

4.9.6 המרה מקוד אסקי לערך מספרי - ( )ord

הפונקציה מחזירה את הערך השלם לתו ה unicode שהכנסנו

דוגמאות:

```
print(ord("A"))
print(ord('0'))
print(ord("א"))
```

ההדפסות שנקבל :

```
65
48
1488
```

4.9.7 המרה להקסה ( )hex

הפונקציה ממירה את המספר שצוין לערך הקסה דצימלי שלו. הערך המוחזר הוא מחרוזת .

דוגמאות:

```
x = hex(255)
print("x = ", x, "and type of x : ", type(x))
print(hex(40+8))
a,b = 60,5
print(hex(a+b))
```

ההדפסות שנקבל :

```
x =  0xff and type of x :  <class 'str'>
0x30
0x41
```

4.9.8 המרה לאוקטלי ( )oct

הפונקציה ממירה את המספר שצוין לערך אוקטלי (בסיס 8) שלו. הערך המוחזר הוא מחרוזת .

דוגמאות:

```
x = oct(255)
print("x = ", x, "and type of x : ", type(x))
print(oct(40+8))
a,b = 60,5
print(oct(a+b))
```

ההדפסות שנקבל :

```
x =  0o377 and type of x :  <class 'str'>
0o60
0o101
```

4.9.9 המרה לבינארי bin( )

הפונקציה ממירה את המספר שצוין לערך בינארי שלו. הערך המוחזר הוא מחרוזת .

דוגמאות:

```
x = bin(255)
print("x = ", x, "and type of x : ", type(x))
print(bin(40+8))
a,b = 60,5
print(bin(a+b))
```

ההדפסות שנקבל :

```
x =  0b11111111 and type of x :  <class 'str'>
0b110000
0b1000001
```

4.10 קלט מהמשתמש input ופלט print

פעולה של קלט מהמשתמש מתבצעת בעזרת הפונקציה **input** ופעולה של פלט למסך מתבצעת בעזרת הפונקציה **print** .

4.10.1 קלט בעזרת פונקציית input

הפונקציה input מאפשרת לבצע קלט מהמשתמש .

התחביר של הפונקציה :

משתנה = input(" מחרוזת שתודפס למסך ")

לדוגמה :

```
>>> color = input("which is the color you like the most ?  : ")
```

נקבל את ההודעה ונכניס red .

```
which is the color you like the most ?  : red
```

נבקש הדפסה של color

```
>>> print(color)
red
```

נבדוק האם המשתמש הקיש red ? אם כן נדפיס nice . אם לא נדפיס ok

```
>>> if color == "red":
 print("nice")
else:
 print("ok")
```

ההדפסה שנקבל כאשר הכנסנו red : nice

הטיפוס שהמשתמש מקיש הוא תמיד str (מחרוזתי) .

דוגמה:

```
a= input("Please enter a number : ")
Please enter a number : 123
>>> type(a)
<class 'str'>
```

אם נרצה קלט של משתנה מטיפוס אחר נעשה casting .

דוגמה :

```
>>> b = int(input("Please enter a number : "))
Please enter a number : 123
>>> type(b)
<class 'int'>
```

דוגמה : נרשום תוכנית שיכולה לגרום לבלבול .

```
>>> a= input("Please enter a number : ")
```

נקבל הדפסה להכניס מספר ונקיש 123 :

```
Please enter a number : 123
```

נרשום את הפקודה :

```
>>> print (5*(a))
```

נקבל :

```
123123123123123
```

הדבר יכול לבלבל ואם חשבנו שנקבל את המכפלה 123 * 5 שהיא 615 לא כך הוא. קיבלנו 5 הדפסות של המחרוזת 123 .

אם בכל זאת נרצה לקבל את המכפלה 5*123 נצטרך לבצע casting לקליטת המשתנה b ולרשום את הפקודות הבאות :

```
>>> b = int(input("Please enter a number : "))
Please enter a number : 123
>>> print(5*b)
615
```

4.10.2 פלט בעזרת הפונקציה print

הפונקציה print מדפיסה למסך מספרים , מחרוזות , משתנים, ביטויים ועוד. בהדפסה ניתן להיעזר בתו הבקרה \ (back slash – אלכסון הפוך) עם אפשרות של \n (New line) להורדת הסמן לתחילת השורה הבאה או \t (Tab) להעברת הסמן לשדה ההדפסה הבא או בתו הבקרה % כפי שנראה בדוגמאות הבאות . הפונקציה print( ) איננה מחזירה ערך.

דוגמאות :

1. כדי לרדת שורה נרשום print( ) .
2. הדפסת המחרוזת המופיעה בין גרשים בודדים :

```
>>> print ('Hello, yood dalet!')
Hello, yood dalet!
```

3. הדפסת המחרוזת המופיעה בין גרשיים (גרשים כפולים):

```
>>> print ("Hello, yood dalet!")
Hello, yood dalet!
```

4. הדפסת המחרוזת גם אם שמים 3 פעמים גרשיים :

```
print ("""Hello, yood dalet!""")
Hello, yood dalet!
```

5. שימוש ב \n כדי לרדת לתחילת השורה הבאה (New line) .

```
>>>print ("Hello\nYood dalet")
Hello
Yood dalet
```

6. שימוש ב \t כדי לעבור לשדה ההדפסה הבא (tab) .

```
>>> print ("Hello\tYood dalet")
Hello  Yood dalet
```

7. ניתן לבטל את ירידת השורה או העברת הסמן לשורה הבאה על ידי הוספת התו r לפני המחרוזת :

```
>>> print (r"Hello\nYood dalet")
Hello\nYood dalet
```

```
>>> print (r"Hello\tYood dalet")
Hello\tYood dalet
```

8. הדפסת \n ו \t ללא ירידת שורה (על ידי הוספת אלכסון הפוך) :

```
>>> print ("hello \\n Yood \\t dalet")
hello \n Yood \t dalet
```

9. הדפסה של גרש בתוך מחרוזת : נרשום את המחרוזת בין גרשיים ואז נוכל לרשום גרש :

```
>>> print("I don't like homework")
I don't like homework
```

10. הדפסה של גרש במחרוזת שנמצאת בין גרשים על ידי הוספת אלכסון הפוך :

```
>>> print("I don\'t like homework")
I don't like homework
```

11 . הזכרנו שהפונקציה print איננה מחזירה ערך . בדוגמה כאן ננסה להעביר למשתנה x את הערך החוזר מהפונקציה ונראה שנקבל None .

```
>>> x= print ("Hello Yood Dalet")
Hello Yood Dalet
>>> print(x)
None
```

12. הדפסה של משתנה :

```
>>> x=10
>>> print (x)
10
```

13. הדפסה של מספר משתנים :

```
>>> x,y,z = 10,20,30
>>> print ( x , y ,z)
10 20 30
```

```
>>> print("x+y = ", x ,'+', y,'=',x+y)
x+y =  10 + 20 = 30
```

14. הדפסה של מחרוזות ומשתנים : ניתן לשלב מחרוזות ומשתנים עם הפרדה של פסיק ביניהם.

```
>>> x,y = 10,20
>>> print("x+y = ", x , y,x+y)
```

```
x+y =  10 20 30
```

15. דרך נוספת להדפסת משתנים ומחרוזות (די דומה לשפת C) בעזרת התו **%** :

```
>>> myName = "ArPo"
>>> print("my name is %s and I am %d years old" % (myName,20))
my name is ArPo and I am 20 years old
```

16. הדפסה עם קביעה מה יופיע בין ההדפסות בעזרת המילה sep :

```
>>> print (1,2,3,4,sep='+')
1+2+3+4
```

17. הדפסה בעזרת המילה end : לפעמים רוצים לבצע הדפסה בלולאה אבל לא לרדת שורה . כאן נעזרים במילה end . בהתחלה נראה דוגמה להדפסה בלולאה אבל בכל הדפסה יורדים שורה:

```
>>> for x in range(3):
        print(x)

0
1
2
```

נראה כיצד ניתן לרשום את כל ההדפסות בשורה אחת:

```
>>> for x in range(3):
        print(x, end='    ')

0    1    2
```

4.11 פונקציות/מתודות לפעולות מתמטיות

קיימות מספר פונקציות בפייתון עבור פעולות מתמטיות. נסביר את העיקריות.

4.11.1 הפונקציה abs( )

הפונקציה מחזירה את הערך המוחלט של מספר ששולחים אליה.

אם עובדים עם מספרים קומפלקסים כמו לדוגמה abs(3+5j) אז נצטרך למצוא את השורש של $\sqrt{(3^2 + 5^2)}$ שהוא 5.8309518948453 .

דוגמה :

```
r = abs(-3.14)
print(r)
x = abs(1j)
print(x)
y=abs(1+1j)
print(y)
z = abs(3+5j)
print(z)
```

ההדפסות שנקבל :

```
3.14
1.0
1.4142135623730951
5.830951894845301
```

4.11.2 הפונקציה round( )

הפונקציה round( ) מחזירה מספר עם נקודה צפה שהוא הערך המעוגל של המספר שצוין, עם מספר המספרים העשרוניים שצוין. התחביר : **round*(number, digits)*** .

number הוא המספר שאותו רוצים לעגל .

digits – כמה ספרות לעגל . אם לא רושמים כלום אז ברירת המחדל היא 0 .

דוגמאות :

```
print(round(12))   # עבור מספרים שלמים
print(round(3.14)) # עבור מספרים ממשיים
print(round(10.5))  # עבור מספר באמצע יש עיגול למעלה
print(round(2.6653, 2)) # עבור ממשיים עם דיוק של 2 ספרות
print(round(2.6743, 3))  # עבור ממשיים עם דיוק של 3 ספרות
print(round(2.6753, 3))  # עבור ממשיים עם דיוק של 3 ספרות
print(round(2.6753, 4))  # עבור ממשיים עם דיוק של 3 ספרות
```

ההדפסות שנקבל:

```
12
3
10
2.67
```

2.674

2.675

2.6753

4.11.3 הפונקציה pow( )

הפונקציה pow( ) מחזירה את הערך של x בחזקת y (x^y) . אם קיים פרמטר שלישי, הוא מחזיר x בחזקה של y, מודולוס – שארית - z . (z הוא אופציונלי ולא חייבים לשים אותו).

התחביר : **pow(*x, y, z*)** . x המספר שיש להעלות בחזקת y . z הוא המודולוס – שארית כלומר (x^y) %z

דוגמאות (ללא z):

```
print(pow(3, 4))    # 3**4
print(pow(-2,5 ))   # (-2)**5
print(pow(2,-5))    # 2**(-5)
print(pow(-2,-5))   # (-2)**(-5)
```

ההדפסות שנקבל :

81

-32

0.03125

-0.03125

דוגמאות לשימוש במודולוס z .

```
print(pow(3, 4,10))     # (3**4) % 10 = 81%10 = 1
print(pow(7,2,5))       # (7**2) % 5  = 49 % 6 = 4
```

וההדפסות :

1

4

4.11.4 הפונקציה sum ( )

הפונקציה sum( ) מחברת את כל הפריטים של iterable ומחזירה את סכומם. ה iterable יכול להיות רשימה , tuple , dict (מילון) וכו. הפריטים חייבים להיות מספרים.

התחביר : sum(iterable, start) .

ה iterable כפי שציינו, יכול להיות רשימה , tuple , dict (מילון) וכו.

start – הערך שיש לחבר אל ה iterable . הוא לא חייב להופיע (אופציונלי).

דוגמאות :

```
numbers = [2.5, -3, 4, -5]          # הגדרה של רשימה עם אתחול נתונים
sum_of_numbers = sum(numbers)       # הפונקציה sum המסכמת את כל הפריטים
print(sum_of_numbers)
```

```
sum_of_numbers = sum(numbers, 10)  # (start) 10 + סיכום המספרים שברשימה
print(sum_of_numbers)
```

ההדפסות שנקבל :

```
-1.5
8.5
```

4.11.5 המתודה divmod( )

המתודה divmod ( ) מקבלת שני מספרים (ב tuple) ומחזירה זוג מספרים (ב tuple) המורכבים מהמנה (תוצאת החלוקה) והשארית שלהם.

התחביר: divmod(x,y)

אם x ו y מספרים שלמים יוחזרו הערכים : x // y ו x%y .

אם אחד מהם , או שניהם, הם ממשיים (float) יוחזר מספר ממשי שהוא x / y ו x % y .

דוגמאות :

```
# עבודה עם מספרים שלמים
print('divmod(9, 4) = ', divmod(9, 4))
print('divmod(4, 9) = ', divmod(4, 9))
print('divmod(5, 5) = ', divmod(5, 5))

# עבודה עם מספרים ממשיים
print('divmod(9.0, 4) = ', divmod(9.0, 4))
print('divmod(4, 9.0) = ', divmod(4, 9.0))
print('divmod(7.55, 2.5) = ', divmod(7.55, 2.5))
print('divmod(2.66, 0.5) = ', divmod(2.66, 0.5))
```

ההדפסות שנקבל :

```
divmod(9, 4) =  (2, 1)
divmod(4, 9) =  (0, 4)
divmod(5, 5) =  (1, 0)
divmod(9.0, 4) =  (2.0, 1.0)
divmod(4, 9.0) =  (0.0, 4.0)
divmod(7.55, 2.5) =  (3.0, 0.04999999999999982)
divmod(2.66, 0.5) =  (5.0, 0.16000000000000014)
```

4.11.6 הפונקציה ( )max והפונקציה ( )min

הפונקציה ( )max מחזירה את הפריט הגדול ביותר ב iterable . ניתן להשתמש בפונקציה גם כדי למצוא את הפריט הגדול ביותר בין שני פרמטרים או יותר.

התחביר : max(iterable, *iterables, key, default)

iterable - כל טיפוס נתונים שהוא iterable כמו list, tuple. set, dictionary .

***iterables** – אופציונאלי . כל מספר של iterables . יכול להיות יותר מ 1 .

key – אופציונאלי . פונקציית key (נלמד בפרקים הבאים) שבה מועברים ה iterables (טיפוסי הנתונים) ומתבצעת השוואה בהתבסס על הערך המוחזר (יהיה ברור יותר בדוגמה שבהמשך).

default – אופציונאלי. ערך ברירת המחדל אם ה iterable ריק.

הפונקציה ( )min מחזירה את הפריט הקטן ביותר ב iterable . ניתן להשתמש בפונקציה גם כדי למצוא את הפריט הקטנביותר בין שני פרמטרים או יותר.

דוגמאות :

```
# מציאת המקסימום מבין ערכים ששולחים לפונקציה
max_number = max(5, -10, 8, 5)
print("The maximum number is:", max_number)

# מציאת הערך המקסימלי ברשימה
number = [5, -10, 8, 5, 10, 6, 10]
biggest_number = max(number);
print("The biggest number is:", biggest_number)

# מציאת המחרוזת עם עם ערך יוניקוד הגדול ביותר
languages = ["Python", "C Programming", "Java", "Visual Studio"]  # V is the bigger unicode in
the list
largest_string = max(languages);
print("The largest string is:", largest_string)
```

ההדפסות שנקבל:

```
The maximum number is: 8
The biggest number is: 10
```

The largest string is: Visual Studio

דוגמאות לשימוש ב ( )min

```
# מציאת המינימום מבין הערכים ששולחים לפונקציה
min_number = min(5, -10, 8, 5)
print("The minimumm number is:", min_number)

# מציאת הערך המינימלי ברשימה
number = [5, -10, 8, 5, 10, 6, 10]
smallest_number = min(number);
print("The smallest number is:", smallest_number)

# מציאת המחרוזת עם עם ערך יוניקוד הקטן ביותר
languages = ["Python", "C Programming", "Java", "Visual Studio"]  # C is the smallest unicode in the list
smallest_string = min(languages);
print("The smallest string is:", smallest_string)
```

ההדפסות שנקבל :

```
The maximum number is: 8
The biggest number is: 10
The largest string is: Visual Studio
The minimumm number is: -10
The smallest number is: -10
The smallest string is: C Programming
```

4.12 תרגילים

לכל התרגילים נניח ש x=9 ו y=4 והוגדר import math .

1. יש לרשום פקודה שתדפיס את מכפלת הערכים x ו y .
2. יש לרשום פקודה שתדפיס את החלוקה בין x ל y .
3. יש לרשום 2 שורות שייבדקו האם x לא שווה ל y ותוציא הדפסה מתאימה .
4. מה נקבל מקטע התוכנית הבאה ?

```
if(x==y or x>y) :
  print("YES")
```

5. מה נקבל מקטע התוכנית הבאה ?

```
if(x==y and x>y) :

    print("YES")
```

6. מה נקבל מקטע התוכנית הבאה ?

```
if(x%y == 1) :
   print("yes")
```

7. מה נקבל בקטע התוכנית הבא ?

```
x = 9
y=11
x&=y
print(x)
```

8. מה נקבל מקטע התוכנית הבא ?

```
x = 9
y=4
y|=x
print(y)
```

9. מה נקבל עבור x^=y ?
10. מה נקבל עבור x>>=2 ?
11. מה נקבל עבור x<<=y ?
12. מה נקבל עבור z>>2 אם ידוע ש z=-40 ?
13. מה נקבל עבור w>>3 אם ידוע ש w=-40 ?
14. מדוע מקבלים False בתוכנית הבאה ?

```
colors = ["Red", "Green", "Blue"]
print("Green " in colors)
```

15. יש להדפיס את תוצאות התרגילים הבאים:

```
print(math.floor(0.123456))
print(math.floor(-15.0213))
print(math.floor(-25.3))
print(math.floor(20.654))

print(math.ceil(-15.32))
print(math.ceil(-120.654))
print(math.ceil(40.0))
print(math.ceil(-1.9999))
print(round(15.1276543, 2))
print(round(-13.12345,3))
print(round(-12.65432101,6))
print(round(-123,456))
```

16. נתונה התוכנית הבאה (השאלה שייכת לרשימות ואין הכרח לפתור אותה כאן):

```
list1=['A','B','C','a','b','c','א','ב','ג']
    for i in list1:
        print(i,"= ",ord(i),"unicode", " = ", hex(ord(i)))
```

מה ההדפסות בסיום התוכנית ?

17. יש לקלוט שתי רשימות של חמישה איברים ולמצוא מי המספר הגדול ב 2 הרשימות.

18. יש לקלוט רשימה של 5 שמות ולהדפיס מהי הרשימה "הגדולה" ביותר.