

```
def removeDuplicates(lst: list) -> list:
    no_duplicates = []
    for element in lst:
        if element not in no_duplicates:
            no_duplicates.append(element)

    return no_duplicates
```

```
nums = [1, 2, 3, 4, 1, 2, 3, 4]
print(removeDuplicates(nums))
```

הפונקציה מקבלת רשימה, ומחזירה רשימה ללא כפילויות.

```
[1, 2, 3, 4]
```

Process finished with exit code 0

```
def increase(points: list) -> bool: 2 usages
    for index in range(1, len(points)):
        x1, y1 = points[index - 1]
        x2, y2 = points[index]
        if y2 < y1:
            return False
    return True
```

```
print(increase([(1, 2), (3, 4), (5, 6)]))
print(increase([(1, 2), (3, 4), (5, 3)]))
```

בדיקה האם הקו עולה (בהתאם ל-y)

```
True
False
```

Process finished with exit code 0

```
def zigzag(n: int) -> bool: 2 usages
    if n < 99:
        return True

    n = str(n)
    n1 = n[0]
    n2 = n[1]
    for i in range(len(n)):
        if (i % 2 == 0) and (n[i] != n1):
            return False
        if (i % 2 == 1) and (n[i] != n2):
            return False
    return True
```

```
print(zigzag(313132))
print(zigzag(31313))
```

הפונקציה בודקת האם המספר זיגזג כלומר, האם הוא מורכב מרצף של 2 ספרות.

```
False
True
```

Process finished with exit code 0

```
def most_char(s: str) -> str:
    s = s.lower()
    x = s[0]
    y = s.count(x)
    for c in s:
        if s.count(c) > y:
            x = c
            y = s.count(c)
    return x

print(most_char("hello"))
```

הפונקציה בודקת מה התו שמופיע הכי הרבה פעמים במחרוזת ומחזירה אותו.

```
l
```

Process finished with exit code 0

```
def divide(a: int, b: int) -> int:
    x = 0
    while a >= b:
        x += 1
        a -= b
    return x

print(divide(a: 9, b: 2))
```

הפונקציה מחלקת את a ב-b ללא שארית (a//b)

4

Process finished with exit code 0

```
def complement(numbers: list) -> list: 1 usage
    check_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    new_list = []
    for num in check_list:
        if num not in numbers:
            new_list.append(num)
    return new_list

print(complement([2, 3, 4, 5, 6, 7, 9, 10, 13]))
```

הפונקציה עוברת על הרשימה ומחזירה איזה מספרים בין 1 ל-10 חסרים בה.

[1, 8]

Process finished with exit code 0

```
x = int(input("Enter a number: "))
num = 1
line = 1
while num <= x:
    for j in range(line):
        print(num, end=" ")
        num += 1
        if num > x:
            break
    print()
    line += 1
```

התוכנית קולטת מספר מהמשתמש, ומדפיסה את כל המספרים מ-1 עד המספר שנקלט, תוך חלוקה לשורות (שורה ראשונה תכלול מס' 1, שורה שנייה 2 מס' וכן הלאה)

```
Enter a number: 6
1
2 3
4 5 6
```

הפונקציה מקבלת 2 רשימות של מספרים שלמים. היא מחזירה רשימה של כל במספרים ברשימה הראשונה שמתחלקים בכל המספרים ברשימה השנייה ללא שארית.

```
def divide_by(numbers: list, divisors: list) -> list: 1 usage
    new_list = []
    for number in numbers:
        checker = True
        for divisor in divisors:
            if number % divisor != 0:
                checker = False
                break
        if checker:
            new_list.append(number)
    return new_list

print(divide_by(numbers: [2, 3, 5, 6, 8, 9, 12, 18], divisors: [2, 3]))
```

[6, 12, 18]

Process finished with exit code 0

```
def sum_digits(n: int) -> int:
    sum_a = 0
    while n != 0:
        sum_a += n % 10
        n = n // 10
    return sum_a

print(sum_digits(992))
```

הפונקציה מקבלת מספר שלם וסוכמת את סכום הספרות שמרכיבות אותו.

20

Process finished with exit code 0

```
def ten_power(n: int) -> int:
    r = 1
    for i in range(n, 0, -1):
        r *= 10
    return r

print(ten_power(3))
```

הפונקציה מקבלת מספר מעריך חזקה לבסיס 10 ומחזירה את התוצאה.

1000

Process finished with exit code 0

```
def doubles_check(lst: list) -> bool:
    t = []
    for x in lst:
        if x in t:
            return False
        t.append(x)
    return True

print(doubles_check([1, 2, 3, 4]))
print(doubles_check([1, 2, 3, 1]))
```

הפונקציה בודקת אם קיימות כפילויות ברשימה ומחזירה ערך בוליאני.

True
False

Process finished with exit code 0

```
def panagram(sentence: str) -> bool:
    sentence = sentence.lower()
    abc = "abcdefghijklmnopqrstuvwxyz"
    for char in abc:
        if char not in sentence:
            return False
    return True

print(panagram("abcdefghijklmnopqrstuvwxyz"))
print(panagram("afas"))
```

הפונקציה בודקת אם הטקסט מכיל את כל אותיות ה-abc, ומחזירה ערך בוליאני בהתאם.

True
False

Process finished with exit code 0

```
def triangle_mul(n: int) -> str:
    for row in range(n + 1):
        for column in range(row):
            print(f"{row * (column + 1)}", end=" ")
        print()
    for row in range(n - 1, 0, -1):
        for column in range(row):
            print(f"{row * (column + 1)}", end=" ")
        print()

triangle_mul(5)
```

הפונקציה מדפיסה משולש
לוח כפל

```
1
2 4
3 6 9
4 8 12 16
5 10 15 20 25
4 8 12 16
3 6 9
2 4
1
```

```
def mul_rishoni(n: int) -> int:
    mul = 1
    for i in range(2, n):
        checker = True
        for x in range(2, i):
            if i % x == 0:
                checker = False
                break
        if checker:
            mul *= i
    return mul

print(mul_rishoni(11))
```

הפונקציה מכפילה מספרים ראשוניים
עד הערך שהוכנס עליה (ולא כולל).
כלומר עבור הערך 11, היא תחשב:
 $2 \cdot 3 \cdot 5 \cdot 7 = 210$ (כי 11 לא נכלל)

210

Process finished with exit code 0

```
def different_nums(n: int) -> bool:
    checker = ""
    for char in str(n):
        if char not in checker:
            checker += char
        else:
            return False
    return True

print(different_nums(1234))
print(different_nums(1231))
```

הפונקציה בודקת אם הספרות של הערך
שונות אחת מהשנייה או לא.

True
False

Process finished with exit code 0

```
def power(a: int, b: int) -> int:
    x = a
    for i in range(b - 1):
        x *= a
    return x

print(power(2, 4))
print(2 * 2 * 2 * 2)
```

הפונקציה מחשבת את הבסיס (a) בחזקת המעריך (b) ומחזירה תוצאה

```
16
16
Process finished with exit code 0
```

```
def letters_in_str(s: str) -> list:
    letters_list = []
    for char in s.lower():
        if 'a' <= char <='z' and char not in letters_list:
            letters_list.append(char)
    return letters_list

print(letters_in_str("This is NoamNew12345"))
```

הפונקציה בודקת איזה אותיות יש בסטרינג שהתקבל ומחזירה אותן כרשימה.

```
['t', 'h', 'i', 's', 'n', 'o', 'a', 'm', 'e', 'w']
Process finished with exit code 0
```

```
def larger_than(lst: list, num: int) -> int:
    val = None
    for n in lst:
        if n > num:
            if val is None:
                val = n
            elif n < val:
                val = n
    return val

print(larger_than([1,2,3,4,5], 5))
print(larger_than([1,2,3,4,5], 3))
```

הפונקציה עוברת על רשימה ומחזירה None אם אין מספר גדול מהמספר הנוסף, או מחזירה את המספר הקטן ביותר שגדול מהמספר הנוסף.

```
None
4
Process finished with exit code 0
```

```
word = input("Enter a word: ")
letters = ""
for char in word:
    letters += char
    print(letters, end="\n")
```

התוכנית מדפיסה מילה עם תוספת אות בכל ירידת שורה.

```
Enter a word: Noam
N
No
Noa
Noam
```

```
import random

# usage

def guess(num: int) -> int:
    count = 0
    while True:
        x = random.randint(0, num)
        if x == num:
            return (count + 1)
        count += 1

n = int(input("Enter a number: "))
print(guess(n))
```

הפונקציה בודקת כמה פעמים לוקח למחשב
לנחש את המספר המדויק שהשתמש הכניס.

```
Enter a number: 200
56
Process finished with exit code 0
```

```
def miniff(numbers: list) -> (int, int):
    diff = max(numbers) - min(numbers)
    n1 = None
    n2 = None
    for num in numbers:
        for val in numbers:
            if num != val:
                if abs(num - val) < diff:
                    diff = abs(num - val)
                    n1 = num
                    n2 = val
            else:
                continue
    return (n1, n2)

print(miniff([1,3,4,10,11]))
```

הפונקציה מקבלת רשימה ומחזירה את
2 המספרים (הראשונים) שהפרש ביניהם
הוא הקטן ביותר (ללא שימוש ב-sort)

```
(3, 4)
Process finished with exit code 0
```

```
def aestrics_triangle(n: int) -> str:
    for row in range(n + 1):
        for column in range(row):
            print("*", end=" ")
        print()
    for row in range(n - 1, 0, -1):
        for column in range(row):
            print("*", end=" ")
        print()

aestrics_triangle(5)
```

הפונקציה מדפיסה משולש כוכביות
עולה ויורד

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

```
def TribonacciChecker(n: int) -> bool:
    a = 0
    b = 1
    c = 1
    while a <= n:
        if a == n:
            return True
        next_term = a + b + c
        a = b
        b = c
        c = next_term
    return False

print(TribonacciChecker(13))
print(TribonacciChecker(11))
```

הפונקציה בודקת אם המספר שהתקבל
נמצא על הרצף הטריבונאצי'

```
True
False

Process finished with exit code 0
```

```
def find_local_maxima(nums: list[int]) -> list[int]:
    local_max = []
    index = 0
    while index < len(nums):
        # First index
        if index == 0:
            if nums[index] > nums[index + 1]:
                local_max.append(nums[index])
        # Last Index
        elif index == len(nums) - 1:
            if nums[index] > nums[index - 1]:
                local_max.append(nums[index])
        # Middle Index
        else:
            if nums[index + 1] < nums[index] > nums[index - 1]:
                local_max.append(nums[index])
        index += 1

    return local_max

print(find_local_maxima([1,2,3,2,5,6,7,8,5]))
```

הפונקציה מוצאת את הערך המקסימלי
באיברים ברשימה (מי שגדול מהשכנים)
ומחזירה אותו

```
[3, 8]

Process finished with exit code 0
```

```
def is_perfect(n: int) -> bool:
    if n <= 1:
        return False
    sum_divisors = 0
    for i in range(1, n):
        if n % i == 0:
            sum_divisors += i
    return sum_divisors == n

print(is_perfect(6))
print(is_perfect(10))
```

הפונקציה בודקת אם מספר הוא
מספר מושלם (שווה לסכום כל
המחלקים שלו למעט עצמו)

```
True
False

Process finished with exit code 0
```

```
def word_len_diff(sentence: str) -> int:
    words = sentence.split(' ')
    min = len(words[0])
    max = len(words[0])
    for word in words:
        if len(word) > max:
            max = len(word)
        else:
            if len(word) < min:
                min = len(word)
    return max - min

print(word_len_diff("I love python programming"))
```

הפונקציה מוצאת את המילה הארוכה
והקצרה ביותר במחרוזת, ומחזירה
את ההפרש באורך שלהן

10

Process finished with exit code 0