

```

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))

```

הfonקציה מקבלת רשימה, ומחזירה רשימה ללא כפליות.

```
[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

```

בדיקה האם היקו עולה (בהתאם ל-y)

```
True  
False
```

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

```

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

```

```

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))

```

הfonקציה בודקת האם המספר זיגזג
כלומר, האם הוא מרכיב 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"))

```

הfonקציה בודקת מה התו שופיע 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))

```

הfonקציה 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]))

```

הfonקציה עוברת על הרשימה ומוחזירה 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 מ' 1, שורה שנייה מ' 2 וכו' וכן הלאה)

```

Enter a number: 6
1
2 3
4 5 6

```

הfonקציה מקבלת 2 רשימות של מספרים שלמים. היא מוחזירה רשימה של כל המספרים 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))
```

הfonקציה מקבלת מספרשלם וווכמת 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))
```

הfonקציה מקבלת מספרעריך 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]))
```

הfonקציה בודקת אם קיימות כפליות בראשימה ומחזירה ערךבוליאני.

```
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"))
```

הfonקציה בודקת אם הטקסט מכיל את כל אותיות הא-abc, ומחזירה ערךבוליאני בההתאם.

```
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)

```

הfonקציה מדפסת 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))

```

הfonקציה מכפילה מספרים ראשוניים
עד הערך שהוכנס עלייה (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))

```

הfonקציה 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)

```

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

```

16
16
Process finished with exit code 0

```

הfonקציה בודקת איזה אותיות יש בסטרינג
שהתקבל ומחזירה אותן כרשימה.

```

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"))

```

```

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

```

הfonקציה עוברת על רשימה ומחזירה
None אם אין מספר גדול מהמספר
הנוסף, או מחזירה את המספר הקטן
bijור שגדל מהמספר הנוסף.

```

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))

```

```

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 מדפסה 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))

```

הfonקציה בודקת כמה פעמים לוקח למחשב
לנחש את המספר המדוייק שהמשתמש 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]))

```

הfonקציה מקבלת רשימה ומוחזירה את
2 המספרים (הראשונים) שההפרש ביניהם
הוא הקטן ביותר (לא שימוש ב-

```

(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)

```

הfonקציה מדפיסה משולש כוכבות
עליה 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))

```

הfonקציה בודקת אם המספר שהתקבל
נמצא על הרצף הטריבונאצ'

```

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]))

```

הfonקציה מוצאת את הערך המקסימלי
באיברים בראשימה (מי שగודל מהשכנים)
ומוחזירה 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))

```

הfonקציה 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"))

```

הfonקציה מוצאת את המילה הארוכה והקצרה ביותר במחוץ, ומחזירה את ההפרש באורך שלהן

```

10
Process finished with exit code 0

```

```

def multiply(n: int, m: int) -> int:
    result = 0
    counter = 0
    while counter < m:
        result += n
        counter += 1
    return result

print(multiply(3, 4))

```

הfonקציה מחשבת מכפלה של 2 אינטים שהתקבלו

```

12
Process finished with exit code 0

```

```

def discount(sum: int, isStudent: bool) -> float:
    if sum <= 250:
        return (sum * 0.1)
    if isStudent:
        return (sum * 0.25)
    return (sum * 0.15)

print(discount(250, True))
print(discount(250, False))
print(discount(300, True))
print(discount(300, False))

```

הfonקציה מקבלת סכום קנייה וערך בולי אני אם הלקוחה סטודנט.

אם קנה ממתח ל-250 (כולל – הfonקציה תחזיר כמה תהיה 10% הנחה).

אם קנה מעל 250, אם סטודנט – 25% – הנחה, אם לא סטודנט – 15% – הנחה.

* הfonקציה מחזירה את סכום ההנחה ולא סכום הקנייה לאחר הנחה!

```

25.0
25.0
75.0
45.0

```

```
def is_unique(num: int) -> bool:
    num = abs(num)
    while num > 0:
        current_digit = num % 10
        remaining_part = num // 10
        temp_num = remaining_part
        while temp_num > 0:
            if temp_num % 10 == current_digit:
                return False
            temp_num //= 10
        num //= 10
    return True

print(is_unique(1234506))
print(is_unique(12345060))
```

הfonקציה עוברת על מספר שלם (בלי להפקיד סטרינג, רשימה, סט וכו') ובודקת אם יש בו כפיליות במספרים.

```
True  
False  
  
Process finished with exit code 0
```

```
def divisible(nums: list, divisor: int) -> list:
    divisible = []
    for num in nums:
        if num % divisor == 0:
            divisible.append(num)
    if divisible:
        return divisible
    return None

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

פונקציה שמקבלת רשימה מספרים ומספר.
מחזירה את כל המספרים בראשיה שמתחלקים
ללא שארית במספר הבודד.
אם אין כאלה – תחזיר None

```
[2, 4]
None

Process finished with exit code 0
```

```
def aztec_pyramid(n: int):
    width = 2
    for row in range(n):
        max_width = 2 + (n - 1) * 4
        spaces = (max_width - width) // 2
        for _ in range(2):
            print(" " * spaces + "*" * width)
        width += 4

aztec_pyramid(5)
```

פירמידה אצטנית (כל קומבה היא 2 קומות וקומה ראשונה היא 2 כוכבויות וכל קומה אחרת גדרה ב-4 כוכבויות)


```

def pyramid(char: str, n: int) -> str:
    for row in range(n, 1, -1):
        print(char * row)
        print(char * row)
    print(char)
    for row in range(2, n + 1):
        print(char * row)
        print(char * row)

pyramid("$", 3)

```

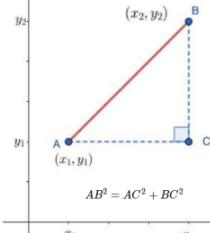
פירמידה שאין לי מושג איך לקרוא לה

\$\$\$
\$\$\$
\$\$
\$\$
\$
\$\$
\$\$
\$\$\$
\$\$\$

כתבו פונקציה המתקבלת כפרמטר רשותה של tuples המכילה קואורדינטות (y,x) של נקודות שונות במשורט.
לحل דוגמה לשימוש כזו [[1,1),(2,2),(5,5),(6,6)]. עליכם למצוא את המרחק הקצר ביותר בין שתי הנקודות
הקרובות ביותר ובסופה ללחוץ או וו מילנווקיה.

רמז: בכיר מתרגיל בית 2 ניתן לחשב מרחק בין שני נקודות באמצעות משפט פיתגוראס.

הערות:
נניח לנו שהרשימה כוללת לפחות 2 נקודות
ניתן לחסוך פונקציות עוז למימוש



```

import math
2 usages
def calc_distance(point1, point2):
    x1, y1 = point1
    x2, y2 = point2
    return math.sqrt(math.pow(x1 - x2, 2) + math.pow(y1 - y2, 2))

1 usage
def shortest_distance(coords: list):
    min_distance = calc_distance(coords[0], coords[1])
    for index1 in range(len(coords)):
        for index2 in range(index1 + 1, len(coords)):
            curr_dist = calc_distance(coords[index1], coords[index2])
            if curr_dist < min_distance:
                min_distance = curr_dist
    return min_distance

print(shortest_distance([(6,6), (5,5), (2,3), (1,1)]))

```

1.4142135623730951

Process finished with exit code 0

```

def FractionalDecrease(nominator, denominator):
    minimum = min(nominator, denominator)
    for i in range(minimum, 0, -1):
        if (nominator % i) == 0 and (denominator % i) == 0:
            return nominator // i, denominator // i

print(FractionalDecrease(123, 492))
print(FractionalDecrease(99, 33))

```

מצומם מונה ומכנה של שבר

(1, 4)
(3, 1)

Process finished with exit code 0