**תרגיל בית 1**

מגישים: אביחי חדד (209286665)

אלעד פישר (\*\*\*\*\*\*\*\*\*\*\*\*)

1. קוד:

|  |
| --- |
| # Ex 01 - Calculating the intersection coordinates of two straight lines in plane  # according to two points on each line.  # getting user inputs about the coordinates of the straight lines in plane.  lineA\_x0, lineA\_y0, lineA\_x1, lineA\_y1 = eval(input("Insert line A data: "))  lineB\_x0, lineB\_y0, lineB\_x1, lineB\_y1 = eval(input("Insert line B data: "))  # calculating the slope of a straight line.  lineA\_slope = (lineA\_y0 - lineA\_y1) / (lineA\_x0 - lineA\_x1)  lineB\_slope = (lineB\_y0 - lineB\_y1) / (lineB\_x0 - lineB\_x1)  # checking if the lines are parallel lines.  if lineA\_slope == lineB\_slope:  print("The is no intersection between the two lines.\n"  "try inserting different coordinates, goodbye!")  else:  # calculating the n(unknown) for the equation "y = mx + n".  lineA\_n = lineA\_y0 - lineA\_slope \* lineA\_x0  lineB\_n = lineB\_y0 - lineB\_slope \* lineB\_x0  # calculating the x coordinate of the intersection according to the  # equation x = (Bn - An) / (slopeA - slopeB).  intersection\_x = (lineB\_n - lineA\_n) / (lineA\_slope - lineB\_slope)  # calculating the y coordinate of the intersection according to the equation "y = mx + n."  intersection\_y = lineA\_slope \* intersection\_x + lineA\_n  # printing the coordinates of the intersection.  print("the coordinates of the intersection are: (%.2f, %.2f)" % (intersection\_x, intersection\_y)) |

פלט:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

1. קוד:

|  |
| --- |
| # Ex 02 - getting an integer number and printing a triangle and parallelogram  # with the height of that number.  # getting the size from the user.  size = int(input("Insert size: "))  # running the loop "size" times to print the triangle and parallelogram.  for i in range(1, size + 1):  # declaring the left space(left to the triangle).  left\_space = " " \* (size - i)  # declaring the triangle with base of (size \* 2 - 1).  triangle = "\*" \* (i \* 2 - 1)  # declaring the middle space between the triangle and the parallelogram.  middle\_space = " " \* ((size - i) \* 2)  # declaring the parallelogram.  parallelogram = "\*" \* size  # printing all variables together as a single string.  print(left\_space + triangle + middle\_space + parallelogram) |

פלט:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1. קוד:

|  |
| --- |
| # Ex 03 - Calculating the sum of all the odd numbers  # under 400,000 in the Fibonacci sequence.  # declaring the first two elements in the sequence in a variables.  previous\_element = 1  next\_element = 1  # declaring a variable of the sum of all the odd numbers under 400,000  # declaring him as "previous\_element"(=1) because we want to calculate also the first element  # in the sequence.  sun\_of\_odd\_numbers = previous\_element  # running in a loop until the next index in the Fibonacci sequence is over to 400,000.  while next\_element <= 400000:  # declaring a temp variable in order not to lose "next\_element"'s value later on,  # for assigning the "previous\_element" to be equal to it.  tmp\_next\_element = next\_element  # calculating the next element in the sequence  next\_element = next\_element + previous\_element  # assigning the previous\_element to be equal to the next element in the sequence  previous\_element = tmp\_next\_element  # checking if the "previous\_element" is an odd number, and if so, adding him to the sum.  # checking with "previous\_element" and not with the "next\_element" because the last calculation  # of the next element in the sequence is over 400,000.  if previous\_element % 2 != 0:  sun\_of\_odd\_numbers += previous\_element  # printing the sum of all the odd numbers in the Fibonacci sequence under 400,000.  print("The sum of all odd numbers in Fibonacci sequence is:", sun\_of\_odd\_numbers) |

פלט:

|  |  |
| --- | --- |
|  |  |

1. קוד:

|  |
| --- |
| # Ex 04 - getting a positive integer number from the user, and splitting it  # into group of three digits(if possible) and dividing each digit by 3 and checking  # if all the three digits has different remainders.  # if so the program print's "OK", if not "ERROR"  import math  # getting a positive integer number from the user.  number = input("Insert number: ")  # calculating the number of digits in that number.  digits = int(math.log10(int(number))) + 1  # printing "OK" if there is only 1 digit.  if digits == 1:  print("OK")  # checking the possibility if the number has 2 digits.  elif digits == 2:  # checking if they have different remainders and if so printing "OK".  if int(number[0]) % 3 != int(number[1]) % 3:  print("OK")  # if they have the same remainder printing "ERROR"  else:  print("ERROR")  # if digits > 2.  else:  # running over all the digits in the number,  # and checking each group of three digit separately.  # we used "range(digits - 2)" because it's the last iteration we need to check.  for i in range(digits - 2):  # declaring the remainders of each digit in a variable.  first\_digit\_remainder = int(number[i]) % 3  second\_digit\_remainder = int(number[i + 1]) % 3  third\_digit\_remainder = int(number[i + 2]) % 3  # checking if all the three digits have different remainders.  if (first\_digit\_remainder != second\_digit\_remainder) and \  (first\_digit\_remainder != third\_digit\_remainder) and \  (second\_digit\_remainder != third\_digit\_remainder):  # if true - skipping to the next iteration.  continue  # if there are some digit with the same remainder.  else:  # printing "ERROR" if 2 or more digits have the same remainder.  print("ERROR")  # exiting the loop.  break  # printing "OK" if each digit in group of three in the whole number has different remainder.  else:  print("OK") |

פלטים:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |