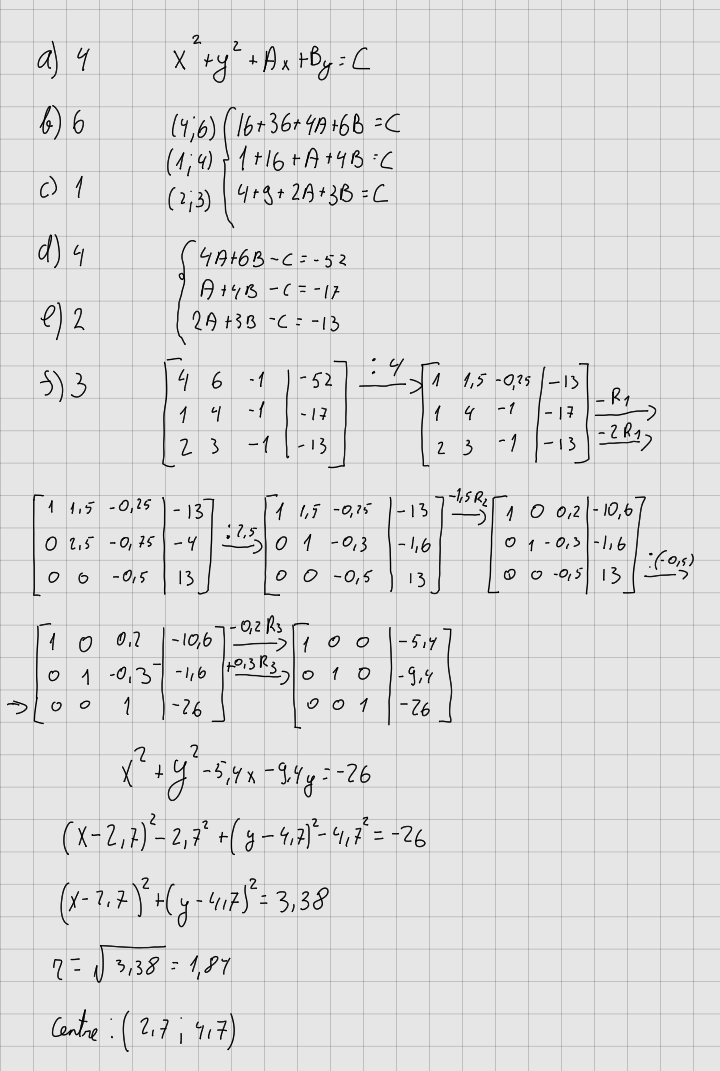
# Calculation by hand



# Python

1. import csv  # importing CSV library to use csv functions
2. from typing import List
3. import numpy as np  # importing CSV library to use matrix-calculations
4. # The class Point initializes an object with x and y coordinates as float values.
5. class Point:
6. def \_\_init\_\_(self, x, y):
7. self.x = float(x)
8. self.y = float(y)
9. # [\]
10. def print(self):
11. """
12. prints the values of the attributes "x" and "y".
13. """
14. print("{0} {1}".format(self.x, self.y))
15. # [\]
16. def circeq(self):
17. return [self.x, self.y, -1]
18. def circsq(self):
19. return -self.x\*\*2 - self.y\*\*2
20. x = 0
21. y = 0
22. def dictionary(self):
23. """
24. The function returns the dictionary representation of an object's attributes.
25. :return: The method `dictionary` is returning the dictionary representation of the object's
26. attributes using the `\_\_dict\_\_` attribute.
27. """
28. return self.\_\_dict\_\_
29. # [\]
30. # `inputarray = []` initializes an empty list called `inputarray`. This list will be used to store
31. # `Point` objects created from the data in the input file.
32. inputarray = []
33. # [\]
34. # This code is opening a CSV file named "inputfile AS1.csv" in read mode using the `open()` function
35. # and assigning it to the variable `input`. Then, it is using the `csv.DictReader()` function to read
36. # the contents of the CSV file and convert them into a dictionary format. The resulting dictionary is
37. # assigned to the variable `locations`. The `with` statement is used to ensure that the file is
38. # properly closed after it has been read.
39. with open(r"inputfile AS1.csv", "r") as input:
40. locations = csv.DictReader(input)
41. # [\]
42. # Essentially, this code is converting the data from the input CSV file into a list of `Point` objects
43. # that can be used for further calculations.
44. for pointentry in locations:
45. point = Point(pointentry["x"], pointentry["y"])
46. inputarray.append(point)
47. # [\]
48. # `calcarray = np.array(inputarray)` is converting the list of `Point` objects stored in `inputarray`
49. # into a numpy array called `calcarray`. This allows for easier manipulation and calculation of the
50. # data using numpy functions.
51. calcarray = np.array(inputarray)
52. # [\]
53. # `numOfRows = calcarray.shape[0]` is calculating the number of rows in the numpy array `calcarray`
54. # and assigning it to the variable `numOfRows`. This value is used later in the code to determine if
55. # there are exactly three points in the input file, in which case a circle will be calculated.
56. numOfRows = calcarray.shape[0]
57. # [\]
58. sol = []
59. eq = []
60. # This code block checks if the number of rows in the input CSV file is equal to 3. If it is, then it
61. # assumes that the input file contains three points and calculates the equation of the circle that
62. # passes through those three points.
63. if numOfRows == 3:
64. print("Given three points a circle will be calculated")
65. for n in range(3):
66. eq.append(inputarray[n].circeq())
67. sol.append(inputarray[n].circsq())
68. eqarray = np.array(eq)
69. solarray = np.array(sol)
70. solution = np.linalg.solve(eqarray, solarray)
71. # [\]
72. # additional +/- formatting  (You do not need to implement this but it is recommended to do)
73. if solution[0] < 0:
74. pr\_sol0 = str(solution[0])
75. else:
76. pr\_sol0 = "+" + str(solution[0])
77. if solution[1] < 0:
78. pr\_sol1 = str(solution[1])
79. else:
80. pr\_sol1 = "+" + str(solution[1])
81. if solution[2] < 0:
82. pr\_sol2 = str(solution[2])
83. else:
84. pr\_sol2 = abs(solution[2])
85. Equation = ["Equation:", "x^2+y^2{0}x{1}y = {2}".format(pr\_sol0, pr\_sol1, pr\_sol2)]
86. print(Equation[0], Equation[1])
87. #This code is writing the equation of the circle calculated
88. # earlier to a new CSV file named "outputfile AS1.csv".
89. f = open("outputfile AS1.csv", "w")
90. writer = csv.writer(f)
91. writer.writerow(Equation)
92. f.close()
93. # [\]