Imports

```
from math import sqrt
from collections import Counter
```

Utilities

```
# euclidian distance
def distance(point_1, point_2):
    return sqrt((point_1[0] - point_2[0])**2 + (point_1[1] - point_2[1])**2)
```

```
def knn_classify(k, training_points, training_classifications, point):
    # tuple = (euc_distance, class)
    distances = [(distance(point, training_points[i]), training_classifications[i])
    distances.sort(key=lambda distance: distance[0]) # sort by first value in tuple

    # get the k nearest points by splice
    k_nearest_points = distances[:k]
    k_nearest_labels = [label for points, label in k_nearest_points] # get the labl

    most_common = Counter(k_nearest_labels).most_common(1) # this lists the n(param
    return most_common[0][0]
```

```
# set training points to tuples: x, y
training_points = [(4, 21), (5, 19), (10, 24), (4, 17), (3, 16), (11, 25), (14, 24)

# set class lables to y since used for classification
training_classifications = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]
```

Test

```
point_1 = (8, 21)
point_2 = (14, 25)
point_3 = (11, 22)
point_4 = (5, 20)
```

```
test_1 = knn_classify(1, training_points, training_classifications, point_1)
test_2 = knn_classify(1, training_points, training_classifications, point_2)
test_3 = knn_classify(4, training_points, training_classifications, point_3)
test_4 = knn_classify(4, training_points, training_classifications, point_4)

print(f"point (8, 21) classifification, should be 0: {test_1}")
print(f"point (14, 25) classification, should be 1: {test_2}")
print(f"point (8, 21) classifification, should be 1: {test_3}")
print(f"point (14, 25) classification, should be 0: {test_4}")
```

```
point (8, 21) classifification, should be 0: 0
point (14, 25) classification, should be 1: 1
point (8, 21) classifification, should be 1: 1
point (14, 25) classification, should be 0: 0
```

Refrences:

https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/