Q1: (5 points) Why the data type in NumPy is floating point in many cases?

By default dtype uses a floating in numpy which is (np.float64)

Example:

Arr = np.array([2, 5.0, 2])

Arr.dtype

# dtype('float64')

In the example one of the elements is 5.0 is a floating-point number. Floats are a more complex data type in Python, which means that all other data types have to match this. Therefore, all elements of the array are converted to floats and are stored with the dtype float64.To change this we can explicitly change the dtype to Int or float.

Q2: (40 points) Write a statement that performs the desired action. Assume the list housePrices = ['$140,000', '$550,000', '$480,000'] exists. (10 point for each subquestion)

* 1. Update the price of the second item in housePrices to '$175,000'.

A: housePrices[1]= '$175,000'

* 1. Add a price to the end of the list with a value of '$1,000,000', use .append().

A: housePrices[len(housePrice):] = ['$1,000,000']

* 1. Remove the first element from housePrices, using the pop() method.

A: housePrices.pop(0)

* 1. Sort the list in-place (without creating a new object) by calling sort function:

A: housePrices.sort(key=lambda x: x.split('$'))

print(housePrices)

Q3: (10 points) Which statement adds 'data' to the following dictionary? (5 points) Why? (5 points)

prices = {'apples': 1.99, 'oranges': 1.49, 'kiwi': 0.79}

prices['pears'] = 1.79

prices['pears': 1.79]

Answer: prices['pears'] = 1.79

Here a new key-value pair is added to an existing dictionary using the assignment operator.

Q4: (10 points) What's the result of set([100, 200, 100, 200, 300])? Select the correct answer (5 points) and explain Why (5 points):

A list with the following elements: [100, 200, 100, 200, 300].

A set that contains 100, 200, and 300.

A set that contains 100, 200, 300, another 100, and another 200.

Answer: A set that contains 100, 200, and 300

The built-in set type in Python includes the following features:

* Sets have no order.
* Set components are special and their elements cannot have duplicates.
* Although the elements that make up a set may be changed, the set itself must be immutable.

Q5: (25 points) At the end of Chapter4, section 4.5, you can find an example of utilizing array operations. Clearly explain why you need the following method in such code: (5 points for each)

* + 1. .cumsum(): cumsum() method is used to get cumulative sum of the random steps in the walk code.

* + 1. .min(): After calculating the cumulative sum we can get the min value along the walk trajectory.

* + 1. .max(): After calculating the cumulative sum we can get the max value along the walk trajectory.

* + 1. .abs(walk): It is used to get how long it took the random walk to get at least n steps away from the origin 0 in either direction. .abs(walk) gives us a boolean array indicating where it has reached or exceeded n.

* + 1. .argmax(): To find out the index of the first n or -n we can use argmax to compute this, which returns the first index of the maximum value in the boolean array.

Q6: (10 points) The python code (given in p123) simulates 5,000 random walks, change this code to: (5 points for each)

6.1 Simulate 7500 walks, a single random walk with 2,000 steps.

import numpy as np

nwalks = 7500

nsteps = 2000

draws = np.random.randint(0, 2, size=(nwalks, nsteps)) # 0 or 1

steps = np.where(draws > 0, 1, -1)

walks = steps.cumsum(1)

Walks

Output: array([[ -1, -2, -3, ..., 84, 85, 84],

[ -1, 0, -1, ..., 70, 69, 70],

[ -1, 0, 1, ..., -6, -5, -6],

...,

[ -1, -2, -3, ..., -12, -11, -12],

[ 1, 2, 1, ..., -2, -1, 0],

[ -1, 0, 1, ..., -8, -7, -8]])

6.2. Find how long it took the random walk to get at least 50 steps away from the origin 0 in either direction

hits50 = (np.abs(walks) >= 50).any(1)

print(hits50)

print("Number that hit 50 or -50 is: ", hits50.sum())

crossing\_times = (np.abs(walks[hits50]) >= 50).argmax(1)

print("crossing time is: ", crossing\_times.mean())

Output:

[False False True ... True False False]

Number that hit 50 or -50 is: 3941

crossing time is: 1134.7746764780513