assign1

April 10, 2020

[1]:
```
import numpy as np
```

1 Assignment 1

1.1 Question 1

Use NumPy to create a 10x10 array that contains the multiplication table. Show the code to create the array. Use the `outer` function to create the array.

[4]:
```
x = np.arange(1,11)
y = np.arange(1,11)
ans1 = np.multiply.outer(x,y)
print(ans1)
```

```
[[  1   2   3   4   5   6   7   8   9  10]
 [  2   4   6   8  10  12  14  16  18  20]
 [  3   6   9  12  15  18  21  24  27  30]
 [  4   8  12  16  20  24  28  32  36  40]
 [  5  10  15  20  25  30  35  40  45  50]
 [  6  12  18  24  30  36  42  48  54  60]
 [  7  14  21  28  35  42  49  56  63  70]
 [  8  16  24  32  40  48  56  64  72  80]
 [  9  18  27  36  45  54  63  72  81  90]
 [ 10  20  30  40  50  60  70  80  90 100]]
```

1.2 Question 2

Write code that selects the 4x4 array that's in the middle of the 10x10 array

[12]:
```
ans2 = ans1[3:7,3:7]
print(ans2)
```

```
[[16 20 24 28]
 [20 25 30 35]
 [24 30 36 42]
 [28 35 42 49]]
```

1.3 Question 3

Using `arange` and `reshape`, create a 4x4 array with numbers 1 thru 16

[13]:
```
a = np.arange(1,17)
ans3 = a.reshape((4,4))
print(ans3)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]
```

1.4 Question 4

Consider the arrays from the last two questions. Create a new 4x4 array that has value of true if both values in the two arrays are even and false otherwise. Hint: use a vectorized custom function with two inputs.

[14]:
```
ans4 = np.vectorize(lambda x,y : (x % 2) + (y % 2) == 0)(ans2,ans3)
print(ans4)
```

```
[[False  True False  True]
 [False False False False]
 [False  True False  True]
 [False False False False]]
```

1.5 Question 5

Write code that returns the number of even numbers in the array from Q2.

[15]:
```
ans5 = (ans2 % 2 == 0).sum()
print(ans5)
```

```
12
```

1.6 Question 6

Write code the returns the square root of the numbers in the array from Q3.

[17]:
```
ans6 = ans3 ** .5
print(ans6)
```

```
[[1.         1.41421356 1.73205081 2.        ]
 [2.23606798 2.44948974 2.64575131 2.82842712]
 [3.         3.16227766 3.31662479 3.46410162]
 [3.60555128 3.74165739 3.87298335 4.        ]]
```

1.7 Question 7

Write code that returns the array from Q1, where the numbers on the diagonal are incremented by 1. Do not explicitly enumerate all the elements on the diagonal.

[22]:
```
ans7 = ans1 + np.eye(10, dtype='int32')
print(ans7)
```

```
[[  2   2   3   4   5   6   7   8   9  10]
 [  2   5   6   8  10  12  14  16  18  20]
 [  3   6  10  12  15  18  21  24  27  30]
 [  4   8  12  17  20  24  28  32  36  40]
 [  5  10  15  20  26  30  35  40  45  50]
 [  6  12  18  24  30  37  42  48  54  60]
 [  7  14  21  28  35  42  50  56  63  70]
 [  8  16  24  32  40  48  56  65  72  80]
 [  9  18  27  36  45  54  63  72  82  90]
 [ 10  20  30  40  50  60  70  80  90 101]]
```

1.8 Question 8

Write code that shows the array from Q1 in reverse order. The first row should have the number 100, 90,...10. The second row should be the ninth row in the initial array in reverse order and so on.

[23]:
```
ans8 = np.flip(ans1)
print(ans8)
```

```
[[100  90  80  70  60  50  40  30  20  10]
 [ 90  81  72  63  54  45  36  27  18   9]
 [ 80  72  64  56  48  40  32  24  16   8]
 [ 70  63  56  49  42  35  28  21  14   7]
 [ 60  54  48  42  36  30  24  18  12   6]
 [ 50  45  40  35  30  25  20  15  10   5]
 [ 40  36  32  28  24  20  16  12   8   4]
 [ 30  27  24  21  18  15  12   9   6   3]
 [ 20  18  16  14  12  10   8   6   4   2]
 [ 10   9   8   7   6   5   4   3   2   1]]
```

1.9 Question 9

Write code that creates a 10x10 array of random float numbers between 0.0 and 10.0. The numbers should appear in the array sorted in ascending order.

[29]:
```
a = np.random.uniform(low=0.0, high=10.0, size=(10,10))
ans9 = np.sort(a)
print(ans9)
```

```
[[0.42050101 2.56825974 3.8230365  4.52281389 5.25019624 5.48390379
  6.2242097  8.19581049 8.48678034 8.55118873]
 [0.28383736 0.34456422 0.88817074 1.27667641 2.5321606  4.05935863
  4.91344157 5.3045352  6.50708239 9.31294366]
 [0.13002284 1.10373167 4.59037606 4.66165633 4.76186787 6.58246118
  8.31305678 8.99477969 9.19828154 9.75446024]
 [1.59461565 1.8202103  3.38695911 4.78243193 5.1995943  6.07509638
  6.50926003 6.7466971  7.84225384 7.90786997]
 [2.94406552 3.45053391 3.92416437 4.30306321 5.47800031 5.96717283
  6.46600161 7.05174219 7.47426347 9.09563652]
 [2.91077872 3.40074169 5.11715889 8.19211911 8.23883189 8.35736278
  8.51347688 9.14376271 9.43394854 9.70901912]
 [1.98039691 3.3730209  3.95225209 5.87540987 6.77431188 7.62033841
  7.81136693 8.09240469 9.45599554 9.62414162]
 [1.26571537 2.02287732 2.17360568 2.74707256 4.9271859  5.88020313
  6.1786304  6.63193907 7.85939028 8.52178034]
 [0.69367685 0.72374485 0.9539382  1.21381852 2.26653954 4.58803212
  4.68678487 5.5427788  6.31518867 7.81390956]
 [1.66751847 2.02665535 2.09848324 4.84451624 4.87751216 5.20018013
  6.54588278 7.41927462 8.08081622 9.53433288]]
```

1.10 Question 10

Write code that creates one-dimensional array of 10 elements. The i th element should be the average of the numbers in the i th row of the array from Q1.

[30]:
```
ans10 = ans1.mean(axis=1)
print(ans10)
```

```
[ 5.5 11.  16.5 22.  27.5 33.  38.5 44.  49.5 55. ]
```

1.11 Question 11

Write a method that performs merge sort. Use `partition`, `split`, recursion, and the `hstack` method.

[44]:
```
def np_merge(arr):
    if len(arr) == 1:
        return arr
    mid = len(arr) // 2
    l,r = np.split(arr, [mid])
    l,r = np_merge(l), np_merge(r)
    merge_locs = np.searchsorted(l,r)
    new_arr = np.insert(l, merge_locs, r)
    return new_arr

a = np.array([1,7,2,0,9,8,6,3,4,5])
```

```python
ans11 = np_merge(a)
print(ans11)
```

```
[0 1 2 3 4 5 6 7 8 9]
```