Numpy Practice - Hints - Unibs 2021

Import the numpy package under the name `np` and print version and configuration (base)

`hint: import … as, np.__version__, np.show_config`

```
import numpy as np
print(np.__version__)
print(np.show_config())
```

```
1.25.2
openblas64__info:
    libraries = ['openblas64_', 'openblas64_']
    library_dirs = ['/usr/local/lib']
    language = c
    define_macros = [('HAVE_CBLAS', None), ('BLAS_SYMBOL_SUFFIX', '64_'), ('HAVE_BLAS_ILP64', None)]
    runtime_library_dirs = ['/usr/local/lib']
blas_ilp64_opt_info:
    libraries = ['openblas64_', 'openblas64_']
    library_dirs = ['/usr/local/lib']
    language = c
    define_macros = [('HAVE_CBLAS', None), ('BLAS_SYMBOL_SUFFIX', '64_'), ('HAVE_BLAS_ILP64', None)]
    runtime_library_dirs = ['/usr/local/lib']
openblas64__lapack_info:
    libraries = ['openblas64_', 'openblas64_']
    library_dirs = ['/usr/local/lib']
    language = c
    define_macros = [('HAVE_CBLAS', None), ('BLAS_SYMBOL_SUFFIX', '64_'), ('HAVE_BLAS_ILP64', None), ('HAVE_LAPACKE', None)]
    runtime_library_dirs = ['/usr/local/lib']
lapack_ilp64_opt_info:
    libraries = ['openblas64_', 'openblas64_']
    library_dirs = ['/usr/local/lib']
    language = c
    define_macros = [('HAVE_CBLAS', None), ('BLAS_SYMBOL_SUFFIX', '64_'), ('HAVE_BLAS_ILP64', None), ('HAVE_LAPACKE', None)]
    runtime_library_dirs = ['/usr/local/lib']
Supported SIMD extensions in this NumPy install:
    baseline = SSE,SSE2,SSE3
    found = SSSE3,SSE41,POPCNT,SSE42,AVX,F16C,FMA3,AVX2
    not found = AVX512F,AVX512CD,AVX512_KNL,AVX512_KNM,AVX512_SKX,AVX512_CLX,AVX512_CNL,AVX512_ICL
None
```

Base Practice

1. Create a null vector of size 10 but the fifth value which is 1 (base)

`hint: array[4]`

```
vector = np.zeros(10, dtype=int)
vector[4]=1

print(vector)
```

```
[0 0 0 0 1 0 0 0 0 0]
```

2. Create a 3x3 matrix with values ranging from 0 to 8 with zeros on the diagonal (base)

`hint: np.arange, reshape, np.eye`

```
vector = np.arange(9, dtype=int)
matrix = vector.reshape(3,3)
matrix = matrix - np.eye(3)*matrix
matrix = matrix.astype(int)

print(matrix)
```

```
[[0 1 2]
 [3 0 5]
 [6 7 0]]
```

3. How to add a border (filled with 0's) around an 5x5 matrix of ones? (base)

`hint: np.ones, np.pad`

```
matrix = np.ones((5,5), dtype=int)
matrix = np.pad(matrix, 1, 'constant')

print(matrix)
```

```
[[0 0 0 0 0 0 0]
 [0 1 1 1 1 1 0]
 [0 1 1 1 1 1 0]
 [0 1 1 1 1 1 0]
 [0 1 1 1 1 1 0]
 [0 1 1 1 1 1 0]
 [0 0 0 0 0 0 0]]
```

4. Normalize a 5x5 random matrix (base)

`hint: (x - mean) / std, np.mean, np.std`

```
matrix = np.random.rand(5, 5)
mean = np.mean(matrix)
std = np.std(matrix)

normalized_matrix = (matrix - mean) / std

print("Original matrix is:")
print(matrix)
print("Normalized matrix is:")
print(normalized_matrix)
```

```
Original matrix is:
[[0.62223932 0.74319503 0.9181007  0.31340405 0.04571241]
 [0.69025511 0.10909981 0.28483145 0.35337776 0.09711582]
 [0.24753615 0.36827128 0.78689754 0.14912591 0.63830883]
 [0.84118013 0.07421888 0.98048581 0.65208882 0.40975391]
 [0.68744619 0.07882823 0.10001211 0.86701085 0.65961068]]
Normalized matrix is:
[[ 0.50801513  0.90828421  1.4870856  -0.51398884 -1.39983944]
 [ 0.73309436 -1.19007657 -0.60854186 -0.38170703 -1.22973422]
 [-0.73196021 -0.33242108  1.05290545 -1.05762137  0.56119267]
 [ 1.23253847 -1.30550523  1.69353166  0.60679367 -0.19514593]
 [ 0.723799   -1.29025188 -1.22014976  1.31801803  0.63168518]]
```

5. Multiply a 5x3 matrix of ones by a 3x2 matrix of ones (real matrix product) (base)

`hint: np.dot`

```
matrix1 = np.ones((5,3), dtype=int)
matrix2 = np.ones((3,2), dtype=int)

result = np.dot(matrix1,matrix2)

print (result)
```

```
[[3 3]
 [3 3]
 [3 3]
 [3 3]
 [3 3]]
```

Intermediate Practice

6. Create a vector of size 10 with values ranging from 0 to 1, both excluded (intermediate)

`hint: np.linspace`

```
vector = np.linspace(0, 1, 11, False) [1:]

print(vector)
```

```
[0.09090909 0.18181818 0.27272727 0.36363636 0.45454545 0.54545455
 0.63636364 0.72727273 0.81818182 0.90909091]
```

7. Create a random vector of size 10 with values in range (-3, 12) and sort it (intermediate)

`hint: sort`

```python
vector = np.random.randint(-3, 12, 10)
print('Random vector:')
print(vector)

vector.sort()
print('Vector after sorting:')
print(vector)
```

```
Random vector:
[ 6 -2 10  2  2  0 -1  2  8  6]
Vector after sorting:
[-2 -1  0  2  2  2  6  6  8 10]
```

8. Create random vector of size 10 and replace the maximum value by its additive inverse and the minimum value (of the original array) with the median value (intermediate)

`hint: argmax, argmin`

```python
vector = np.random.randint(-3, 12, 10)
print('Random vector:')
print(vector)

new_vector = vector.copy()

max_index = np.argmax(vector)
min_index = np.argmin(vector)

new_vector[max_index] = -vector[max_index]
new_vector[min_index] = np.median(vector)

print('New vector:')
print(new_vector)
```

```
Random vector:
[ 3  7  8  5  6  8  1  3 -1  0]
New vector:
[ 3  7 -8  5  6  8  1  3  4  0]
```

9. Randomly replace p elements in a 2D nxn zero matrix to 1 (intermediate)

`hint: np.put, np.random.choice`

```python
n = 5  # Size of the matrix (nxn)
p = 8  # Number of elements to replace

zero_matrix = np.zeros((n, n), dtype=int)

total_elements = n * n
p = min(p, total_elements)
indices_to_replace = np.random.choice(total_elements, p, replace=False)
np.put(zero_matrix, indices_to_replace, 1)

print("Modified Matrix:")
print(zero_matrix)
```

```
Modified Matrix:
[[1 0 1 1 1]
 [0 0 0 0 0]
 [1 0 0 0 0]
 [0 1 0 0 1]
 [0 1 0 0 0]]
```

10. Subtract the mean of each row of a random 5x10 matrix (intermediate)

`hint: mean(axis=,keepdims=)`

```python
random_matrix = np.random.rand(5, 10)
print('Initial random matrix:')
print(random_matrix)

row_means = np.mean(random_matrix, axis=1, keepdims=True)
new_matrix = random_matrix - row_means

print("New matrix:")
print(new_matrix)
```

```
Initial random matrix:
[[0.85348749 0.08124139 0.4542064  0.48038739 0.58463211 0.47079149
  0.66922937 0.40856873 0.94428138 0.62071633]
 [0.5439322  0.84063487 0.91891885 0.94382265 0.96897799 0.84714604
  0.79835016 0.45163905 0.68767113 0.53908236]
 [0.51765432 0.75709522 0.6971319  0.20849077 0.59077746 0.52564934
  0.88168515 0.27198324 0.03670267 0.07437621]
 [0.62233735 0.52650363 0.43092619 0.65357108 0.8072183  0.05904776
  0.0500315  0.15947916 0.20323658 0.70479437]
 [0.97553516 0.05613453 0.41172607 0.99363773 0.80947119 0.29142773
  0.88495982 0.94623785 0.90348366 0.21373106]]
New matrix:
[[ 0.29673328 -0.47551281 -0.10254781 -0.07636682  0.0278779  -0.08596271
   0.11247516 -0.14818547  0.38752717  0.06396212]
 [-0.21008533  0.08661734  0.16490132  0.18980512  0.21496046  0.09312851
   0.04433263 -0.30237848 -0.0663464  -0.21493517]
 [ 0.06149969  0.30094059  0.24097727 -0.24766386  0.13462283  0.06949471
   0.42553053 -0.18417139 -0.41945196 -0.38177842]
 [ 0.20062276  0.10478904  0.0092116   0.23185649  0.38550371 -0.36266683
  -0.37168309 -0.26223543 -0.21847802  0.28307978]
 [ 0.32690068 -0.59249996 -0.23690841  0.34500325  0.16083671 -0.35720675
   0.23632534  0.29760337  0.25484918 -0.43490342]]
```