NumPy Practice Tasks – Solutions

Simple, Intermediate, and Hard Solutions

Simple Task Solutions

S1. Array from 1 to 20

```
np.arange(1, 21)
```

S2. 3x3 Zero Matrix

```
np.zeros((3,3))
```

S3. 5x5 Identity Matrix

```
np.eye(5)
```

S4. 10 Random Integers

```
np.random.randint(1, 101, 10)
```

S5. Max, Min, Mean

```
arr.max(), arr.min(), arr.mean()
```

S6. Reshape into 3x4

```
arr.reshape(3, 4)
```

S7. 50 Even Numbers

```
np.arange(0, 100, 2)
```

S8. Reverse Array

```
arr[::-1]
```

S9. Extract values ¿ 50

```
arr[arr > 50]
```

S10. 100 Random Floats

```
np.random.random(100)
```

Intermediate Task Solutions

I1. 5x5 Border of 1, Inside 0

```
m = np.ones((5,5))
m[1:-1, 1:-1] = 0
```

I2. Normalize Array

```
(arr - arr.min()) / (arr.max() - arr.min())
```

I3. Stack Vertically + Horizontally

```
np.vstack((a, b))
np.hstack((a, b))
```

I4. Row/Column Sums

```
mat.sum(axis=1), mat.sum(axis=0)
```

I5. Replace Negatives with 0

```
arr[arr < 0] = 0
```

I6. Extract Diagonal

```
np.diag(mat)
```

I7. Split Array into 3 Parts

```
np.split(arr, 3)
```

I8. Dot Product

```
np.dot(a, b)
```

I9. Boolean Mask Filter

```
arr[mask]
```

I10. Euclidean Distance

```
np.linalg.norm(a - b)
```

Hard Task Solutions

H1. Sort 10x10 Matrix

```
np.sort(mat, axis=1)   # row-wise
np.sort(mat, axis=0)   # column-wise
```

H2. Manual Matrix Multiplication

```
result = np.zeros((A.shape[0], B.shape[1]))
for i in range(A.shape[0]):
    for j in range(B.shape[1]):
        for k in range(A.shape[1]):
            result[i, j] += A[i, k] * B[k, j]
```

H3. Variance, SD, Percentiles

```
arr.var(), arr.std(), np.percentile(arr, [25, 50, 75])
```

H4. 1000 Coin Flips

```
flips = np.random.randint(0, 2, 1000)
flips.mean()  # probability of heads
```

H5. Moving Average

```
def moving_avg(arr, k):
    return np.convolve(arr, np.ones(k)/k, mode='valid')
```

H6. 2D Gaussian Matrix

```
x, y = np.meshgrid(np.linspace(-1,1,50), np.linspace(-1,1,50))
g = np.exp(-(x**2 + y**2))
```

H7. One-Hot Encoding

```
def one_hot(arr):
    n = arr.max() + 1
    return np.eye(n)[arr]
```

H8. Division Handling Zero

```
np.divide(a, b, out=np.zeros_like(a), where=b!=0)
```

H9. Extract Slices from 3D

```
arr[0]      # first layer
arr[:,0]    # first row across layers
arr[:,:,0]  # first column across layers
```

H10. Flatten N-D Array Manually

```
def flatten(arr):
    return arr.reshape(-1)
```