```
# --- 1.1 Create an array of zeros ---
print("----- 1.1 Create an array of zeros -----")
zeros_array = np.zeros(10)
print("Array of zeros:", zeros_array)
print()

# --- 1.2 Create an array of ones ---
print("----- 1.2 Create an array of ones -----")
ones_array = np.ones(10)
print("Array of ones:", ones_array)
print()

# --- 1.3 Create an array of fives ---
print("----- 1.3 Create an array of fives -----")
fives_array = np.full(10, 5)
print("Array of fives using np.full:", fives_array)

# Alternatively, without using np.full
fives_array_alt = np.ones(10) * 5
print("Array of fives without using np.full:", fives_array_alt)
print()

# --- 2.1 Create an array of integers from 10 to 50 ---
print("----- 2.1 Create an array of integers from 10 to 50 -----")
int_array = np.arange(10, 50)
print("Array of integers from 10 to 49:", int_array)
print()

# --- 2.2 Generate even integers with a step size of 2 ---
print("----- 2.2 Generate even integers with a step size of 2 -----")
```

```
even_array = np.arange(0, 21, 2)

print("Even integers from 0 to 20:", even_array)

print()


# --- 2.3 Create a 3x3 matrix with values from 0 to 8 using reshape ---

print("----- 2.3 Create a 3x3 matrix with values from 0 to 8 -----")

array_1d = np.arange(9)

matrix_3x3 = array_1d.reshape(3, 3)

print("3x3 Matrix:\n", matrix_3x3)

print()


# --- 3.1 Create an identity matrix ---

print("----- 3.1 Create an identity matrix -----")

identity_matrix = np.eye(3)

print("3x3 Identity Matrix:\n", identity_matrix)

print()


# --- 3.2 Generate random numbers between 0 and 1 ---

print("----- 3.2 Generate random numbers between 0 and 1 -----")

random_matrix = np.random.rand(2, 5)

print("2x5 Matrix of random numbers between 0 and 1:\n", random_matrix)

print()


# --- 3.3 Create an array from a standard normal distribution ---

print("----- 3.3 Create an array from a standard normal distribution -----")

standard_normal_array = np.random.randn(10)

print("Array from standard normal distribution:", standard_normal_array)

print()


# --- 4.1 Create a 10x10 matrix with values from 0.01 to 1 using np.arange ---

print("----- 4.1 Create a 10x10 matrix with values from 0.01 to 1 using np.arange -----")
```

```
values_arange = np.arange(0.01, 1.01, 0.01)  # 100 elements
matrix_arange = values_arange.reshape(10, 10)
print("10x10 Matrix using np.arange:\n", matrix_arange)
print()

# --- 4.2 Create a 10x10 matrix with values from 0.01 to 1 using np.linspace ---
print("----- 4.2 Create a 10x10 matrix with values from 0.01 to 1 using np.linspace -----")
values_linspace = np.linspace(0.01, 1.00, 100)
matrix_linspace = values_linspace.reshape(10, 10)
print("10x10 Matrix using np.linspace:\n", matrix_linspace)
print()

# ===============================
# Intermediate Questions
# ===============================

# --- 5.1 Slice matrices to extract specific rows and columns ---
print("----- 5.1 Slice matrices to extract specific rows and columns -----")
# Create a 5x5 matrix for demonstration
matrix_5x5 = np.arange(25).reshape(5, 5)
print("Original 5x5 Matrix:\n", matrix_5x5)

# Extract the third row (index 2)
third_row = matrix_5x5[2, :]
print("Third Row:", third_row)

# Extract the second column (index 1)
second_column = matrix_5x5[:, 1]
print("Second Column:", second_column)
print()
```

```
# --- 5.2 Extract a submatrix ---
print("----- 5.2 Extract a submatrix -----")
# Create a 6x6 matrix for demonstration
matrix_6x6 = np.arange(36).reshape(6, 6)
print("Original 6x6 Matrix:\n", matrix_6x6)

# Extract rows 2 to 4 (indices 1 to 4) and columns 3 to 5 (indices 2 to 5)
submatrix = matrix_6x6[1:4, 2:5]
print("Submatrix (rows 2-4, columns 3-5):\n", submatrix)
print()

# ================================
# Complex Questions
# ================================

# --- 6.1 Calculate the sum of all elements in a matrix ---
print("----- 6.1 Calculate the sum of all elements in a matrix -----")
# Create a 4x4 matrix
matrix_4x4 = np.arange(16).reshape(4, 4)
print("4x4 Matrix:\n", matrix_4x4)

# Using universal function
total_sum = np.sum(matrix_4x4)
print("Total Sum using np.sum:", total_sum)

# Using method chaining
total_sum_method = matrix_4x4.sum()
print("Total Sum using method chaining:", total_sum_method)
print()

# --- 6.2 Calculate the standard deviation along an axis ---
```

```
print("----- 6.2 Calculate the standard deviation along an axis -----")
# Create a 3x7 matrix with random integers
matrix_3x7 = np.random.randint(0, 10, size=(3, 7))
print("3x7 Matrix:\n", matrix_3x7)

# Compute standard deviation along columns using universal function
std_dev_cols = np.std(matrix_3x7, axis=0)
print("Standard Deviation along columns using np.std:", std_dev_cols)

# Compute standard deviation along columns using method chaining
std_dev_cols_method = matrix_3x7.std(axis=0)
print("Standard Deviation along columns using method chaining:", std_dev_cols_method)
print()

# --- 6.3 Perform axis-specific aggregations ---
print("----- 6.3 Perform axis-specific aggregations -----")
# Create a 5x5 matrix with random integers
matrix_5x5_random = np.random.randint(0, 20, size=(5, 5))
print("5x5 Matrix:\n", matrix_5x5_random)

# Calculate the mean of each row
mean_rows = matrix_5x5_random.mean(axis=1)
print("Mean of each row:", mean_rows)

# Calculate the maximum of each column
max_columns = matrix_5x5_random.max(axis=0)
print("Maximum of each column:", max_columns)
print()

# --- 6.4 Combine multiple statistical operations ---
print("----- 6.4 Combine multiple statistical operations -----")
```

```
# Create a 2x8 matrix with random floats

matrix_2x8 = np.random.rand(2, 8)

print("2x8 Matrix:\n", matrix_2x8)


# Compute sum, mean, and standard deviation along the rows

sum_rows = np.sum(matrix_2x8, axis=1)

mean_rows = np.mean(matrix_2x8, axis=1)

std_dev_rows = np.std(matrix_2x8, axis=1)


print("Sum along rows:", sum_rows)

print("Mean along rows:", mean_rows)

print("Standard Deviation along rows:", std_dev_rows)

print()


# (Optional) Add more slicing examples here if required.
```