Practical 3

Devansh Tyagi
551
E-3 Batch

Perform all matrix operations

```python
import numpy as np

#dataset
names = np.array(['John', 'Emma', 'Michael', 'Sophia'])
ages = np.array([20, 22, 19, 21])
genders = np.array(['M', 'F', 'M', 'F'])
marks = np.array([85, 90, 78, 92])

#All Matrix operations
matrix = np.array([ages, marks])
transposed_matrix = np.transpose(matrix)
matrix_sum = np.sum(matrix)
matrix_mean = np.mean(matrix)
matrix_product = np.prod(matrix)

print("These are all the matrix operations:")
print("========================================")
print("Transpose:\n", transposed_matrix)
print("========================================")
print("Sum:", matrix_sum)
print("========================================")
print("Mean:", matrix_mean)
print("========================================")
print("Product:", matrix_product)
print("========================================")
```

```
These are all the matrix operations:
========================================
Transpose:
 [[20 85]
 [22 90]
 [19 78]
 [21 92]]
========================================
Sum: 427
========================================
Mean: 53.375
========================================
Product: 9637611984000
========================================
```

Horizontal and Vertical stacking of numpy array

```python
import numpy as np

#dataset
names = np.array(['John', 'Emma', 'Michael', 'Sophia'])
ages = np.array([20, 22, 19, 21])
genders = np.array(['M', 'F', 'M', 'F'])
marks = np.array([85, 90, 78, 92])

# Horizontal
horizontal_stack = np.hstack((names,ages,genders,marks))
# Vertical
vertical_stack = np.vstack((names,ages,genders,marks))

print("Horizontal stack:")
print(horizontal_stack)
print("================================")
print("Vertical stack:")
print(vertical_stack)
print("================================")
```

```
Horizontal stack:
['John' 'Emma' 'Michael' 'Sophia' '20' '22' '19' '21' 'M' 'F' 'M' 'F' '85'
 '90' '78' '92']
================================
Vertical stack:
[['John' 'Emma' 'Michael' 'Sophia']
 ['20' '22' '19' '21']
 ['M' 'F' 'M' 'F']
 ['85' '90' '78' '92']]
================================
```

Custom sequence generation

```
[7]  import numpy as np

     #dataset
     names = np.array(['John', 'Emma', 'Michael', 'Sophia'])
     ages = np.array([20, 22, 19, 21])
     genders = np.array(['M', 'F', 'M', 'F'])
     marks = np.array([85, 90, 78, 92])

     # Custom sequence generation
     sequence = np.arange(0, 10, 2)

     print("Custom sequence generation:")
     print("Sequence:", sequence)
```

```
Custom sequence generation:
Sequence: [0 2 4 6 8]
```

Arithmetic and Statistical Operations

```python
import numpy as np

#dataset
names = np.array(['John', 'Emma', 'Michael', 'Sophia'])
ages = np.array([20, 22, 19, 21])
genders = np.array(['M', 'F', 'M', 'F'])
marks = np.array([85, 90, 78, 92])

# Arithmetic and statistical operations
addition = np.add(ages, marks)
subtraction = np.subtract(ages, marks)
mean = np.mean(marks)
std_dev = np.std(marks)
variance = np.var(marks)

print("Arithmetic and statistical operations:")
print("==================================")
print("Addition:", addition)
print("==================================")
print("Subtraction:", subtraction)
print("==================================")
print("Mean:", mean)
print("==================================")
print("Standard Deviation:", std_dev)
print("==================================")
print("Variance:", variance)
print("==================================")
```

```
Arithmetic and statistical operations:
==================================
Addition: [105 112  97 113]
==================================
Subtraction: [-65 -68 -59 -71]
==================================
Mean: 86.25
==================================
Standard Deviation: 5.402545696243577
==================================
Variance: 29.1875
==================================
```

Mathematical Operations

```python
import numpy as np
names = np.array(['John', 'Emma', 'Michael', 'Sophia'])
ages = np.array([20, 22, 19, 21])
genders = np.array(['M', 'F', 'M', 'F'])
marks = np.array([85, 90, 78, 92])

# Mathematical operations
squared = np.square(marks)
square_root = np.sqrt(marks)
logarithm = np.log(marks)

print("Mathematical operations:")
print("================================")
print("Squared:", squared)
print("================================")
print("Square Root:", square_root)
print("================================")
print("Logarithm:", logarithm)
print("================================")
```

```
Mathematical operations:
================================
Squared: [7225 8100 6084 8464]
================================
Square Root: [9.21954446 9.48683298 8.83176087 9.59166305]
================================
Logarithm: [4.44265126 4.49980967 4.35670883 4.52178858]
================================
```

Bitwise Operators

```python
import numpy as np

#dataset
names = np.array(['John', 'Emma', 'Michael', 'Sophia'])
ages = np.array([20, 22, 19, 21])
genders = np.array(['M', 'F', 'M', 'F'])
marks = np.array([85, 90, 78, 92])

# Bitwise operators
bitwise_and = np.bitwise_and(ages, marks)
bitwise_or = np.bitwise_or(ages, marks)
bitwise_xor = np.bitwise_xor(ages, marks)
bitwise_not = np.bitwise_not(ages, marks)

print("Bitwise operators:")
print("==================================")
print("Bitwise AND:", bitwise_and)
print("==================================")
print("Bitwise OR:", bitwise_or)
print("==================================")
print("Bitwise XOR:", bitwise_xor)
print("==================================")
print("Bitwise NOT:", bitwise_not)
print("==================================")
```

```
Bitwise operators:
==================================
Bitwise AND: [20 18  2 20]
==================================
Bitwise OR: [85 94 95 93]
==================================
Bitwise XOR: [65 76 93 73]
==================================
Bitwise NOT: [-21 -23 -20 -22]
==================================
```

Copying and viewing arrays

```python
import numpy as np

#dataset
names = np.array(['John', 'Emma', 'Michael', 'Sophia'])
ages = np.array([20, 22, 19, 21])
genders = np.array(['M', 'F', 'M', 'F'])
marks = np.array([85, 90, 78, 92])

# Copying and viewing arrays
marks_copy = marks.copy()
marks_view = marks.view()

print("Copying and viewing arrays:")
print("==================================")
print("Copy:", marks_copy)
print("==================================")
print("View:", marks_view)
print("==================================")
```

```
Copying and viewing arrays:
==================================
Copy: [85 90 78 92]
==================================
View: [85 90 78 92]
==================================
```

Data stacking, searching, sorting, counting, broadcasting

```python
import numpy as np

#dataset
names = np.array(['John', 'Emma', 'Michael', 'Sophia'])
ages = np.array([20, 22, 19, 21])
genders = np.array(['M', 'F', 'M', 'F'])
marks = np.array([85, 90, 78, 92])

# Data stacking, searching, sorting, counting, broadcasting
stacked_data = np.column_stack((names, ages, genders, marks))
search_index = np.where(names == 'Emma')[0]
sorted_names = np.sort(names)
unique_names, unique_counts = np.unique(names, return_counts=True)
broadcasted_array = np.broadcast_to(marks, (len(names), len(names)))

print("Data stacking, searching, sorting, counting, broadcasting:")
print("================================")
print("Stacked Data:", stacked_data)
print("================================")
print("Search Index:", search_index)
print("================================")
print("Sorted Names:", sorted_names)
print("================================")
print("Unique Names:", unique_names)
print("================================")
print("Unique Counts:", unique_counts)
print("================================")
print("Broadcasted Array:", broadcasted_array)
print("================================")
```

```
Data stacking, searching, sorting, counting, broadcas
================================
Stacked Data: [['John' '20' 'M' '85']
 ['Emma' '22' 'F' '90']
 ['Michael' '19' 'M' '78']
 ['Sophia' '21' 'F' '92']]
================================
Search Index: [1]
================================
Sorted Names: ['Emma' 'John' 'Michael' 'Sophia']
================================
Unique Names: ['Emma' 'John' 'Michael' 'Sophia']
================================
Unique Counts: [1 1 1 1]
================================
Broadcasted Array: [[85 90 78 92]
 [85 90 78 92]
 [85 90 78 92]
 [85 90 78 92]]
================================
```

Thankyou !!