numpypart2

September 12, 2024

[1]:

```python
import numpy as np

array_2d = np.array([[10, 20, 30], [40, 50, 60]])
element = array_2d[1, 2]  # Access the element in the 2nd row, 3rd column
print("Indexed Element:", element)
```

```
Indexed Element: 60
```

[3]:

```python
import numpy as np

array_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
sliced_array = array_2d[1:, 1:]  # Slice the array from the 2nd row and 2nd column
print("Sliced Array:\n", sliced_array)
```

```
Sliced Array:
 [[5 6]
 [8 9]]
```

[4]:

```python
import numpy as np

array_1 = np.array([1, 2, 3])
array_2 = np.array([4, 5, 6])
concatenated_array = np.concatenate((array_1, array_2))
print("Concatenated Array:", concatenated_array)
```

```
Concatenated Array: [1 2 3 4 5 6]
```

[2]:

```python
import numpy as np

array = np.array([1, 2, 3, 4,5,6])
split_array = np.array_split(array, 3)
print("Split Arrays:", split_array)
```

```
Split Arrays: [array([1, 2]), array([3, 4]), array([5, 6])]
```

[6]:

```python
import numpy as np

```

```
array = np.array([1, 2, 2, 3, 4, 4, 5])
unique_elements = np.unique(array)
print("Unique Elements:", unique_elements)
```

```
Unique Elements: [1 2 3 4 5]
```

[7]:

```
import numpy as np

array = np.array([1, 2, 3, 4, 5])
reversed_array = np.flip(array)
print("Reversed Array:", reversed_array)
```

```
Reversed Array: [5 4 3 2 1]
```

[8]:

```
import numpy as np

array_1 = np.array([1, 2, 3])
array_2 = np.array([4, 5, 6])
stacked_array = np.vstack((array_1, array_2))
print("Vertically Stacked Array:\n", stacked_array)
```

```
Vertically Stacked Array:
 [[1 2 3]
 [4 5 6]]
```

[9]:

```
import numpy as np

array = np.array([1, 2, 3])
repeated_array = np.repeat(array, 3)
print("Repeated Array:", repeated_array)
```

```
Repeated Array: [1 1 1 2 2 2 3 3 3]
```

[10]:

```
import numpy as np

array = np.array([1.1, 2.2, 3.3], dtype=np.float64)
int_array = array.astype(np.int32)
print("Integer Array:", int_array)
```

```
Integer Array: [1 2 3]
```

[11]:

```
import numpy as np

array_2d = np.array([[1, 2, 3], [4, 5, 6]])
transposed_array = np.transpose(array_2d)
print("Transposed Array:\n", transposed_array)
```

```
Transposed Array:
 [[1 4]
 [2 5]
 [3 6]]
```

[12]:

```
import numpy as np

string_array = np.array(['apple', 'banana', 'cherry'])
print("String Array:", string_array)
```

```
String Array: ['apple' 'banana' 'cherry']
```

[13]:

```
import numpy as np

array = np.array([1, 2, 3])
tiled_array = np.tile(array, 3)
print("Tiled Array:", tiled_array)
```

```
Tiled Array: [1 2 3 1 2 3 1 2 3]
```

[14]:

```
import numpy as np

array = np.array([1, 2, 3])
tiled_array = np.tile(array, 3)
print("Tiled Array:", tiled_array)
```

```
Tiled Array: [1 2 3 1 2 3 1 2 3]
```

[15]:

```
import numpy as np

random_array = np.random.randint(1, 100, size=(3, 3))
print("Random Integer Array:\n", random_array)
```

```
Random Integer Array:
 [[39 94 81]
 [42 49 68]
 [82 45 89]]
```

[16]:

```
import numpy as np

array = np.array([1, 0, 3, 0])
boolean_array = array.astype(bool)
print("Boolean Array:", boolean_array)
```

```
Boolean Array: [ True False  True False]
```

[1]:

```
import numpy as np

array = np.array([1, 2, 3, 4, 3])
```

```python
array[array == 3] = 99
print("Array after Replacement:", array)
```

```
Array after Replacement: [ 1  2 99  4 99]
```

[2]:

```python
import numpy as np

array = np.array([1, 2, 3, 4, 5])
np.random.shuffle(array)
print("Shuffled Array:", array)
```

```
Shuffled Array: [2 4 5 3 1]
```

[20]:

```python
import numpy as np

array_1 = np.array([1, 2, 3])
array_2 = np.array([4, 5, 6])
stacked_array = np.hstack((array_1, array_2))
print("Horizontally Stacked Array:", stacked_array)
```

```
Horizontally Stacked Array: [1 2 3 4 5 6]
```

[24]:

```python
import numpy as np

array_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
split_rows = np.vsplit(array_2d, 3)
print("Split Rows:", split_rows)
```

```
Split Rows: [array([[1, 2, 3]]), array([[4, 5, 6]]), array([[7, 8, 9]])]
```

[26]:

```python
import numpy as np

array = np.array([10, 20, 30, 40, 50])
bool_index = array > 30
print(bool_index)
filtered_array = array[bool_index]
print("Filtered Array:", filtered_array)
```

```
[False False False  True  True]
Filtered Array: [40 50]
```

[27]:

```python
import numpy as np

list_of_strings = [['cat', 'dog'], ['apple', 'banana']]
string_array = np.array(list_of_strings)
print("String Array:\n", string_array)
```

```
String Array:
 [['cat' 'dog']
 ['apple' 'banana']]
```

[28]:

```
import numpy as np

# Create a 2D array with employee data
employee_data = np.array([
    [101, 'Alice', 28, 'HR'],
    [102, 'Bob', 34, 'Finance'],
    [103, 'Charlie', 25, 'IT'],
    [104, 'David', 30, 'Marketing'],
    [105, 'Eve', 29, 'IT']
])

print("Employee Data Array:\n", employee_data)
```

```
Employee Data Array:
 [['101' 'Alice' '28' 'HR']
 ['102' 'Bob' '34' 'Finance']
 ['103' 'Charlie' '25' 'IT']
 ['104' 'David' '30' 'Marketing']
 ['105' 'Eve' '29' 'IT']]
```

[29]:

```
import numpy as np

date_array = np.arange('2024-01-01', '2024-01-11', dtype='datetime64[D]')
print("Date Array:", date_array)
```

```
Date Array: ['2024-01-01' '2024-01-02' '2024-01-03' '2024-01-04' '2024-01-05'
 '2024-01-06' '2024-01-07' '2024-01-08' '2024-01-09' '2024-01-10']
```

[30]:

```
import numpy as np

data = np.array([(1, 'Alice'), (2, 'Bob')], dtype=[('ID', 'i4'), ('Name',␣
 ↪'U10')])
print("Structured Array:\n", data)
```

```
Structured Array:
 [(1, 'Alice') (2, 'Bob')]
```

[32]:

```
import numpy as np

array_2d = np.array([[1, 2, 3], [4, 5, 6]])
swapped_array = np.swapaxes(array_2d, 0, 1)
print("Swapped Axes Array:\n", swapped_array)
```

```
Swapped Axes Array:
 [[1 4]
 [2 5]
 [3 6]]
```

[36]:

```python
import numpy as np

# Define the structured array with data types
employee_data_type = np.dtype([
    ('EmployeeID', 'i4'),        # 4-byte integer
    ('Name', 'U50'),             # String of max length 50
    ('Department', 'U30'),       # String of max length 30
    ('Salary', 'f8')             # 8-byte float
])

# Create an empty array to store employee data
employees = np.array([
    (101, 'John Doe', 'HR', 55000.50),
    (102, 'Jane Smith', 'IT', 72000.75),
    (103, 'Emily Davis', 'Finance', 61000.00),
    (104, 'Michael Johnson', 'Marketing', 59000.25)
], dtype=employee_data_type)

# Accessing the data
print("Employee Data:")
print(employees)

# Accessing specific fields
print("\nNames of Employees:")
print(employees['Name'])


print("\nSalaries of Employees:")
print(employees['Salary'])
```

```
Employee Data:
[(101, 'John Doe', 'HR', 55000.5 ) (102, 'Jane Smith', 'IT', 72000.75)
 (103, 'Emily Davis', 'Finance', 61000.  )
 (104, 'Michael Johnson', 'Marketing', 59000.25)]

Names of Employees:
['John Doe' 'Jane Smith' 'Emily Davis' 'Michael Johnson']

Salaries of Employees:
[55000.5  72000.75 61000.   59000.25]
```

[37]:

```python
import numpy as np

# Step 1: Define the structured array with additional fields
```

```
employee_data_type = np.dtype([
    ('EmployeeID', 'i4'),            # 4-byte integer
    ('Name', 'U50'),                 # String of max length 50
    ('Department', 'U30'),           # String of max length 30
    ('Salary', 'f8'),                # 8-byte float
    ('PerformanceRating', 'f4'),     # 4-byte float
    ('Bonus', 'f8')                  # 8-byte float, initially set to 0
])

# Step 2: Create an array with employee data including performance ratings
employees = np.array([
    (101, 'John Doe', 'HR', 55000.50, 4.5, 0.0),
    (102, 'Jane Smith', 'IT', 72000.75, 4.8, 0.0),
    (103, 'Emily Davis', 'Finance', 61000.00, 3.9, 0.0),
    (104, 'Michael Johnson', 'Marketing', 59000.25, 4.2, 0.0),
    (105, 'Anna White', 'Sales', 67000.00, 4.7, 0.0)
], dtype=employee_data_type)

print("Initial Employee Data:")
print(employees)

# Step 3: Calculate bonus based on department and performance rating
for employee in employees:
    if employee['Department'] in ['IT', 'Sales']:
        if employee['PerformanceRating'] > 4.5:
            employee['Bonus'] = employee['Salary'] * 0.10
    else:
        if employee['PerformanceRating'] > 4.0:
            employee['Bonus'] = employee['Salary'] * 0.05

print("\nEmployee Data with Calculated Bonuses:")
print(employees)

# Step 4: Identify the top performer in each department
departments = np.unique(employees['Department'])
top_performers = []

for department in departments:
    dept_employees = employees[employees['Department'] == department]
    top_performer = dept_employees[np.
 ↪argmax(dept_employees['PerformanceRating'])]
    top_performers.append(top_performer)

top_performers = np.array(top_performers, dtype=employee_data_type)

print("\nTop Performers in Each Department:")
print(top_performers)
```

```
Initial Employee Data:
[(101, 'John Doe', 'HR', 55000.5 , 4.5, 0.)
 (102, 'Jane Smith', 'IT', 72000.75, 4.8, 0.)
 (103, 'Emily Davis', 'Finance', 61000.  , 3.9, 0.)
 (104, 'Michael Johnson', 'Marketing', 59000.25, 4.2, 0.)
 (105, 'Anna White', 'Sales', 67000.  , 4.7, 0.)]

Employee Data with Calculated Bonuses:
[(101, 'John Doe', 'HR', 55000.5 , 4.5, 2750.025 )
 (102, 'Jane Smith', 'IT', 72000.75, 4.8, 7200.075 )
 (103, 'Emily Davis', 'Finance', 61000.  , 3.9,    0.    )
 (104, 'Michael Johnson', 'Marketing', 59000.25, 4.2, 2950.0125)
 (105, 'Anna White', 'Sales', 67000.  , 4.7, 6700.    )]

Top Performers in Each Department:
[(103, 'Emily Davis', 'Finance', 61000.  , 3.9,    0.    )
 (101, 'John Doe', 'HR', 55000.5 , 4.5, 2750.025 )
 (102, 'Jane Smith', 'IT', 72000.75, 4.8, 7200.075 )
 (104, 'Michael Johnson', 'Marketing', 59000.25, 4.2, 2950.0125)
 (105, 'Anna White', 'Sales', 67000.  , 4.7, 6700.    )]
```

[ ]: