Basic Syntax in Python

Chih-Keng Hung

February 17, 2025

1 Basic arithmetic operations

Code:

```python
a = 10 + 5  # addition
b = 10 - 5  # subtraction
c = 10 * 5  # multiplication
d = 10 / 5  # division

# Exponentiation and Modulus
e = 2 ** 3  # exponentiation
f = 10 % 3  # modulus
print('Results:\n', 'Addition:', a, 'Subtraction:', b, 'Multiplication:', c, 'Division:', d, 'Exponentiation:', e, 'Modulus:', f)
```

Output:

```
Results:
 Addition: 15 Subtraction: 5 Multiplication: 50 Division: 2.0 Exponentiation: 8 Modulus: 1
```

2 String operations

Code:

```python

# Strings
g = 'Hello, World!'  # String
h = 'Hello, ' + 'World!'  # String concatenation
i = 'Hello, World!' * 2  # String multiplication
j = 'Hello, World!'[0]  # String indexing
k = 'Hello, World!'[0:5]  # String slicing

print('Strings:', '\nString:', g, '\nString concatenation:', h, '\nString multiplication:', i, '\nString indexing:', j, '\nString slicing:', k)
```

Output:

```
Strings:
String: Hello, World!
String concatenation: Hello, World!
String multiplication: Hello, World!Hello, World!
String indexing: H
String slicing: Hello
```

3 List operations

Code:

```
# Lists
l = [1, 2, 3, 4, 5]  # List
m = [1, 2, 3, 4, 5] + [6, 7, 8, 9, 10]  # List concatenation
n = [1, 2, 3, 4, 5] * 2  # List multiplication
o = [1, 2, 3, 4, 5][0]  # List indexing
p = [1, 2, 3, 4, 5][0:3]  # List slicing

print('Lists:', '\nList:', l, '\nList concatenation:', m, '\nList multiplication:', n, '\nList indexing:'
                                          , o, '\nList slicing:', p)
```

Output:

```
Lists:
List: [1, 2, 3, 4, 5]
List concatenation: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
List multiplication: [1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
List indexing: 1
List slicing: [1, 2, 3]
```

4 Tuple operations

Code:

```
# Tuples
q = (1, 2, 3, 4, 5)  # Tuple
r = (1, 2, 3, 4, 5) + (6, 7, 8, 9, 10)  # Tuple concatenation
s = (1, 2, 3, 4, 5) * 2  # Tuple multiplication
t = (1, 2, 3, 4, 5)[0]  # Tuple indexing
u = (1, 2, 3, 4, 5)[0:3]  # Tuple slicing

print('Tuples:', '\nTuple:', q, '\nTuple concatenation:', r, '\nTuple multiplication:', s, '\nTuple
                                          indexing:', t, '\nTuple slicing:', u)
```

Output:

```
Tuples:
Tuple: (1, 2, 3, 4, 5)
Tuple concatenation: (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
Tuple multiplication: (1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
Tuple indexing: 1
Tuple slicing: (1, 2, 3)
```

5 Dictionary operations

Code:

```
# Dictionaries
v = {'a': 1, 'b': 2, 'c': 3}  # Dictionary
w = {'a': 1, 'b': 2, 'c': 3}['a']  # Dictionary indexing

print('Dictionaries:', '\nDictionary:', v, '\nDictionary indexing:', w)
```

Output:

```
Dictionaries:
Dictionary: {'a': 1, 'b': 2, 'c': 3}
Dictionary indexing: 1
```

6 Conditional statements

Code:

```python
# if statement
x = 10
print('if statement:')
if x > 5:
    print('x > 5')
elif x < 5:
    print('x < 5')
else:
    print('x = 5')
```

Output:

```
if statement:
x > 5
```

7 Loops

Code:

```python
# for loop
y = [1, 2, 3, 4, 5]
print('for loop:')
for i in y:
    print(i)

# while loop
z = 0
print('while loop:')
while z < 5:
    print(z)
    z += 1
```

Output:

```
for loop:
1
2
3
4
5
while loop:
0
1
2
```

```
3
4
```

8 Function definition and usage.

Code:

```python
# Functions
print('Functions:')
def add(x, y):
    return x + y

print('add(10, 5): 10 + 5 =', add(10, 5))
```

Output:

```
Functions:
add(10, 5): 10 + 5 = 15
```

9 Class definition and usage

```python
# Classes
class MyClass:
    def __init__(self, x):
        self.x = x

    def add(self, y):
        return self.x + y

my_class = MyClass(10)
print('Classes:')
print(my_class.add(5))
```

Output:

```
Classes:
15
```

10 Numpy practice

Code:

```python
# numpy practice
print('numpy practice:')
# 1. Create a 1D array with 10 elements, values from 0 to 9
a = np.arange(10)
print('np.arange(10):', a)

# 2. Create a 1D array with 10 elements, values from 0 to 9, interval 1
b = np.linspace(0, 9, 10)
print('np.linspace(0, 9, 10):', b)

# 3. Create a 2D array with 8 elements, all values are 0
```

```python
c = np.zeros((2, 4))
print('np.zeros((2, 4)):', c)

# 4. Create a 2D array with 8 elements, all values are 0, then use a for loop to assign values
d = np.zeros((2, 4))
for i in range(2):
    for j in range(4):
        d[i, j] = i + j
print('2D array: d =', d)
print('d shape:', d.shape)

# 5. Create an empty array, use a for loop to append elements to form a 1D array
e = []
for i in range(10):
    e.append(i)
print('e:', e)

# 6. Create an empty array, use a for loop to append elements to form a 2D array
f = []
for i in range(2):
    f.append([])
    for j in range(4):
        f[i].append(i + j)
print('f:', f)
print('f shape:', np.array(f).shape)

# 7. Create a 1D array with 5 elements, values are 1, 2, 3, 4, 5 and use the enumerate function to list the index and value
g = [1, 2, 3, 4, 5]
print('g:', g)
for i, j in enumerate(g):
    print('index:', i, 'value:', j)
```

Output:

```
numpy practice:
np.arange(10): [0 1 2 3 4 5 6 7 8 9]
np.linspace(0, 9, 10): [0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
np.zeros((2, 4)): [[0. 0. 0. 0.]
 [0. 0. 0. 0.]]
2D array: d = [[0. 1. 2. 3.]
 [1. 2. 3. 4.]]
d shape: (2, 4)
e: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
f: [[0, 1, 2, 3], [1, 2, 3, 4]]
f shape: (2, 4)
g: [1, 2, 3, 4, 5]
index: 0 value: 1
index: 1 value: 2
index: 2 value: 3
index: 3 value: 4
index: 4 value: 5
```

11 Practice 1

Purpose:

1. To practice the basic usage of numpy and matplotlib.
2. To understand the concept of mean and standard deviation.

Task:

1. Generate a 1-d array with several elements, each element is sampled from a statistical population with a normal distribution with mean = 0 and standard deviation = 3.
2. Plot the data points and mark the mean and standard deviation calculated from the data.
3. Will the mean and standard deviation be the same as the population?
4. State the reason.

Hint:

1. Use `np.random.normal` to generate the data.
2. Visualize the data points with `plt.scatter`, `plt.axvline`, `plt.hist`, `plt.plot`, etc.
3. Calculate the mean and standard deviation with `np.mean` and `np.std` or function defined by yourself. Be aware of the n or n-1 in the denominator used to calculate the standard deviation.

Write your code:

```python
import numpy as np
import matplotlib.pyplot as plt

# Generate data

# Calculate mean and standard deviation

# Plot data

plt.show()

# Print mean and standard deviation
print('......')
```

12 Practice 2

Purpose:

1. To be familiar with the usage of conditional statements, for loop, and indexing.
2. Be able to check the correctness of the coding process by visualization.

Task:

1. Generate a 1-d array with 50 elements, each element is sampled from a statistical population with a normal distribution with mean = 0 and standard deviation = 3.
2. Pick up the data points that are within 1 standard deviation from the mean ($mean - std < data < mean + std$).
3. Plot the picked data points and original data points to check if the filtering process is correct.
4. Mark the mean-stdev and mean+stdev on the plot might be helpful.
5. Print all the index of the data points that are picked (These index might range from 0 to 49).

Hint:

1. Use a for loop to iterate through the data points.
2. Use an if statement to filter out the data points.
3. Add additional lists to store the information (index, value, etc.) needed in the filtering process if needed (`list.append`, `list.index`, `list.pop`, etc.).

Write your code:

```python
import numpy as np
import matplotlib.pyplot as plt

# Generate data

# Calculate mean and standard deviation

# Filter data points within 1 standard deviation from the mean

# Plot data

plt.show()

# Print indices of filtered data points
print('......')
```