0kjqlotlo

June 8, 2024

Problem :

We have an N-element tuple or sequence that you would like to unpack into a collection of N variables.

data = ['ACME', 50, 91.1, (2012, 12, 21)]

Noob Way

[4]:
```
data = [ 'ACME', 50, 91.1, (2012, 12, 21) ]
product = data[0]
x =  data[1]
y=data[2]
z = data[3]

print(product)
print(x)
print(y)
print(z)
```

```
ACME
50
91.1
(2012, 12, 21)
```

Pro Way

[5]:
```
product_new  , x_new , y_new, z_new = data
```

[6]:
```
print(product_new  , x_new , y_new, z_new )
```

```
ACME 50 91.1 (2012, 12, 21)
```

Works with any object which is iterable e.g., strings, files, and generators.

To discard certain values , we can use throwaway variables

[7]:
```
studentDetails = ['John','Doe',345,(150, 100,95)]

firstname , i_am_of_no_use , total_marks , i_am_of_no_use =studentDetails

print(firstname,total_marks)
```

```
John 345
```

What if we dont know the lenght of the iterable. It may lead to “too many values to unpack” exception.

We can use “star expressions” to address this problem.

Refer the below section

Star unpacking (also known as extended unpacking) in Python allows you to capture an arbitrary number of elements from an iterable into a list while unpacking. This is useful when you don’t know the exact number of elements or when you want to unpack some elements while capturing the rest.

Example 1

[8]:
```
#values = [1, 2, 3, 4, 5]

#a, b, c= values  # ValueError: too many values to unpack (expected 3)

#print(a)      # 1
#print(b)      # 2
#print(c)      # 3
```

Example (Simple Start Unpacking)

[9]:
```
values = [1, 2, 3, 4, 5]

a, b, c, *extra = values

print(a)      # 1
print(b)      # 2
print(c)      # 3
print(extra)  # [4, 5]
```

```
1
2
3
[4, 5]
```

Example (start unpacking in the middle)

[10]:
```
values = [1, 2, 3, 4, 5]
a, *middle, d = values

print(a)      # Output: 1
print(middle) # Output: [2, 3, 4]
print(d)      # Output: 5
```

```
1
[2, 3, 4]
```

```
5
```

Example (Start unpacking with throwaway variables)

[11]:
```
values = [1, 2, 3, 4, 5]
a, b, *_ = values

print(a) # Output: 1
print(b) # Output: 2
```

```
1
2
```

Example (Nested Star Unpacking)

[12]:
```
nested_values = [1, 2, [3, 4, 5], 6, 7]
a, b, (c, *d), e, f = nested_values

print(a) # Output: 1
print(b) # Output: 2
print(c) # Output: 3
print(d) # Output: [4, 5]
print(e) # Output: 6
print(f) # Output: 7
```

```
1
2
3
[4, 5]
6
7
```

Example (Star Unpacking With Strings)

[13]:
```
text = "Python:is:a:powerful:programming:language"
a, *middle, c = text.split(':')

print("First word:", a)
print("Middle words:", middle)
print("Last word:", c)
```

```
First word: Python
Middle words: ['is', 'a', 'powerful', 'programming']
Last word: language
```