Python Define and Unpack Tuple

Fan Wang

2020-11-30

Contents

1 Tuple

Go to the **RMD**, **PDF**, or **HTML** version of this file. Go back to fan's Python Code Examples Repository (bookdown site).

```
import numpy as np
```

1.1 Tuple Example

A tuple is created with parenthesis on the sides (or no parenthesis), not brackets on the sides. Can access values in a tuple as in list.

```
# Define Tuple, with and without parenthesis
tp_abc = ('a', 'b', 'c')
tp_abc_noparent = 'a', 'b', 'c'
print(f'{tp_abc=}')
```

```
## tp_abc=('a', 'b', 'c')
```

```
print(f'{len(tp_abc)=}')
```

```
## len(tp_abc)=3
```

```
print(f'{tp_abc_noparent=}')
```

```
## tp_abc_noparent=('a', 'b', 'c')
```

```
print(f'{(tp_abc==tp_abc_noparent)=}')
# Check Type
```

```
## (tp_abc==tp_abc_noparent)=True
```

```
print(f'{isinstance(tp_abc, list)=}')
```

```
## isinstance(tp_abc, list)=False
```

```
print(f'{isinstance(tp_abc, tuple)=}')
# select element
```

```
## isinstance(tp_abc, tuple)=True
```

```
print(f'{tp_abc[1]=}')
```

```
## tp_abc[1]='b'
```

Convert tuple to a list:

```
# convert Tuple to list
ls_abc = [i for i in tp_abc]
print(f'{ls_abc=}')
```

```
## ls_abc=['a', 'b', 'c']
```

```
print(f'{isinstance(ls_abc, list)=}')
```

```
## isinstance(ls_abc, list)=True
```

```
print(f'{isinstance(ls_abc, tuple)=}')
```

```
## isinstance(ls_abc, tuple)=False
```

Since the tuple is not mutable, we can not change values inside the tuple:

```
# define the tuple
tp_abc = ('a', 'b', 'c')
# update tuple value
try:
    tp_abc[0] = 'efg'
except TypeError as error:
    print('Caught this error: ' + repr(error))
```

```
## Caught this error: TypeError("'tuple' object does not support item assignment")
```

1.2 Function Returns Tuple and Unpack

When a function returns multiple *items* in a list, that is a tuple. Each element of the list can be accessed. And the tuple can be unpacked:

```
# Results from some function
tp_results = 'a', 123,  [1,2,3]
# Unpack the results
a_st, b_int, c_ls_int = tp_results
# Print
print(f'{tp_results=}')
```

```
## tp_results=('a', 123, [1, 2, 3])
```

```
print(f'{a_st=}')
```

```
## a_st='a'
```

```
print(f'{b_int=}')
```

```
## b_int=123
```

```
print(f'{c_ls_int=}')
```

```
## c_ls_int=[1, 2, 3]
```

Unpack only a subset of the elements in the tuple:

```
# Unpack only one
a_st_self_m1, _ , _ = tp_results
```

```
# Alternative shorter
a_st_self_m2, *_  = tp_results
# Print
print(f'{a_st_self_m1=}')
```

```
## a_st_self_m1='a'
```

```
print(f'{a_st_self_m2=}')
```

```
## a_st_self_m2='a'
```

see unpack the first two elements in list/tuple.