Cheat Sheet

Tuples and Sequences

None

`None` is an object which is a datatype of its own (NoneType).

Used to define no value or nothing.

Code

PYTHON

```
var = None
print(var)
print(type(var))
```

Output

```
None
<class 'NoneType'>
```

Function Without Return

Functions assigned to a variable, when function does not have a `return` statement, the variable will get the value `None`

Code

PYTHON

```python
def increment(a):
    a += 1

a = 55
result = increment(a)
print(result)
```

Output

None

Function That Returns Nothing

When a function returns no value, the default value will be

None

Example - 1

Code

PYTHON

```python
def increment(a):
    a += 1
    return

a = 55
result = increment(a)
print(result)
```

Output

None

Example - 2

Code

PYTHON

```python
def increment(a):
    a += 1
    return None

a = 5
result = increment(a)
print(result)
```

Output

```
None
```

Example - 3

Code

PYTHON

```python
result = print("Hi")
print(result)
```

Output

```
Hi
None
```

Tuple

- Holds an ordered sequence of items.
- Tuple is immutable object, where as list is a mutable object.

Code

PYTHON

```python
a = 2
tuple_a = (5, "Six", a, 8.2)
```

Creating a Tuple

- Created by enclosing elements within (round) brackets.
- Each item is separated by a comma.

Code

PYTHON

```python
a = 2
tuple_a = (5, "Six", a, 8.2)
print(type(tuple_a))
print(tuple_a)
```

Output

```
<class 'tuple'>
(5, 'Six', 2, 8.2)
```

Tuple with a Single Item

Code

PYTHON

```python
a = (1,)
print(type(a))
print(a)
```

Output

```
<class 'tuple'>
(1,)
```

Accessing Tuple Elements

Accessing Tuple elements is also similar to string and list accessing and slicing.

Code

PYTHON

```python
a = 2
tuple_a = (5, "Six", a, 8.2)
print(tuple_a[1])
```

Output

```
Six
```

Tuples are Immutable

Tuples does not support modification.

Code

PYTHON

```python
tuple_a = (1, 2, 3, 5)
tuple_a[3] = 4
print(tuple_a)
```

Output

```
TypeError: 'tuple' object does not support item assignment
```

Operations can be done on Tuples

- len()
- Iterating

- Slicing
- Extended Slicing

Converting to Tuple

tuple(sequence) Takes a sequence and converts it into tuple.

String to Tuple

Code

PYTHON

```python
color = "Red"
tuple_a = tuple(color)
print(tuple_a)
```

Output

```
('R', 'e', 'd')
```

List to Tuple

Code

PYTHON

```python
list_a = [1, 2, 3]
tuple_a = tuple(list_a)
print(tuple_a)
```

Output

```
(1, 2, 3)
```

Sequence to Tuple

Code

PYTHON

```python
tuple_a = tuple(range(4))
print(tuple_a)
```

Output

```
(0, 1, 2, 3)
```

Membership Check

Check if given data element is part of a sequence or not.

Membership Operators

- in
- not in

Example - 1

Code

PYTHON

```python
tuple_a = (1, 2, 3, 4)
is_part = 5 in tuple_a
print(is_part)
```

Output

```
False
```

Example - 2

Code

PYTHON

```python
tuple_a = (1, 2, 3, 4)
is_part = 1 not in tuple_a
print(is_part)
```

Output

```
False
```

List Membership

Code

PYTHON

```python
list_a = [1, 2, 3, 4]
is_part = 1 in list_a
print(is_part)
```

Output

```
True
```

String Membership

Code

PYTHON

```python
word = 'Python'
is_part = 'th' in word
print(is_part)
```

Output

```
True
```

Packing & Unpacking

Unpacking

Values of any sequence can be directly assigned to variables.

Number of variables in the left should match the length of sequence.

Code

PYTHON

```python
tuple_a = ('R', 'e', 'd')
(s_1, s_2, s_3) = tuple_a
print(s_1)
print(s_2)
print(s_3)
```

Output

```
R
e
d
```

Errors in Unpacking

Code

PYTHON

```python
tuple_a = ('R', 'e', 'd')
s_1, s_2 = tuple_a
print(s_1)
```

```
4 print(s_2)
```

Output

```
ValueError: too many values to unpack (expected 2)
```

Code

PYTHON

```
1 tuple_a = ('R', 'e', 'd')
2 s_1, s_2, s_3, s_4 = tuple_a
3 print(s_1)
```

Output

```
ValueError: not enough values to unpack (expected 4, got 3)
```

Tuple Packing

() brackets are optional while creating tuples.

In Tuple Packing, Values separated by commas will be packed into a tuple.

Code

PYTHON

```
1 a = 1, 2, 3
2 print(type(a))
3 print(a)
```

Output

```
<class 'tuple'>
```

```
(1, 2, 3)
```

Code

PYTHON

```python
a = 1,
print(type(a))
print(a)
```

Output

```
<class 'tuple'>
(1,)
```

Code

PYTHON

```python
a, = 1,
print(type(a))
print(a)
```

Output

```
<class 'int'>
1
```

Submit Feedback