Python Tuples

- Tuples are used to store multiple items in a single variable.
- Tuple items can be of any data type (Eg: String, int and Boolean).
- Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage.
- A tuple is a collection which is **ordered (**sequences, just like lists**)** and **unchangeable i.e. immutable**.
- When we say that tuples are ordered, it means that the items have a defined order, and that order will not change

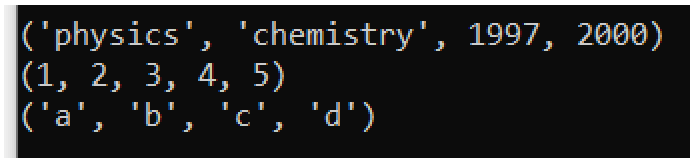
- Tuple allow duplicate values.

- Tuples are written with round brackets.

- The differences between tuples and lists are,
    - the tuples cannot be changed unlike lists and
    - tuples use parentheses, whereas lists use square brackets.
- Creating a tuple is as simple as putting different comma-separated values.
- Optionally you can put these comma-separated values between parentheses also. For example

Example:

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
tup3 = "a", "b", "c", "d";
print(tup1)
print(tup2)
print(tup3)
```



```
('physics', 'chemistry', 1997, 2000)
(1, 2, 3, 4, 5)
('a', 'b', 'c', 'd')
```

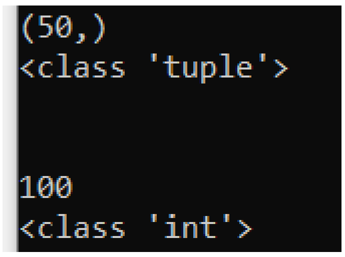
The empty tuple is written as two parentheses containing nothing –

```
tup1 = ();
```

To write a tuple containing a single value (one item), we have to include a comma, even though there is only one value. Otherwise, Python will not recognize it as a tuple.

Example

```
tup1 = (50,);
print(tup1)
print(type(tup1))

print("\n")

tup2 = (100);
print(tup2)
print(type(tup2))
```



```
(50,)
<class 'tuple'>


100
<class 'int'>
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

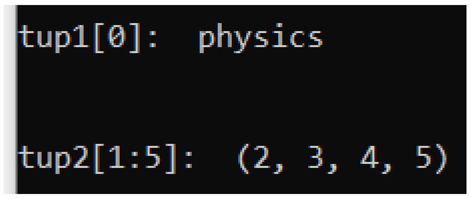
- Tuple items are indexed, the first item has index [0], the second item has index [1] etc.

Accessing Values in Tuples

- To access values in tuple,
    - use the square brackets for slicing along with the index or
    - indices to obtain value available at that index.

Example

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 );

print ("tup1[0]: ", tup1[0]);
print("\n")
print ("tup2[1:5]: ", tup2[1:5]);
```

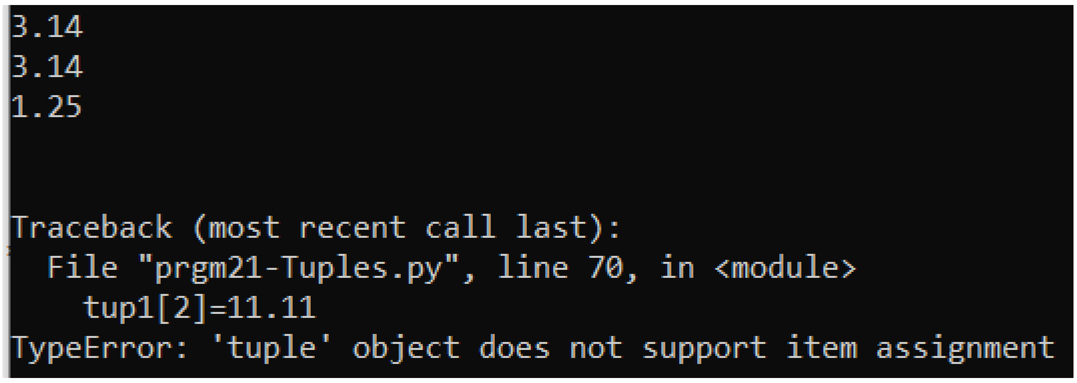


```
tup1[0]:  physics


tup2[1:5]:  (2, 3, 4, 5)
```

Unchangeable

- Tuples are unchangeable (immutable), meaning that
    - we cannot change,
    - add or
    - remove items after the tuple has been created.

```
tup1 = (3.14, 3.14, 1.25);
print(tup1[0])
print(tup1[1])
print(tup1[2])

print("\n")
tup1[2]=11.11
```



```
3.14
3.14
1.25


Traceback (most recent call last):
  File "prgm21-Tuples.py", line 70, in <module>
    tup1[2]=11.11
TypeError: 'tuple' object does not support item assignment
```

Python - Update Tuples

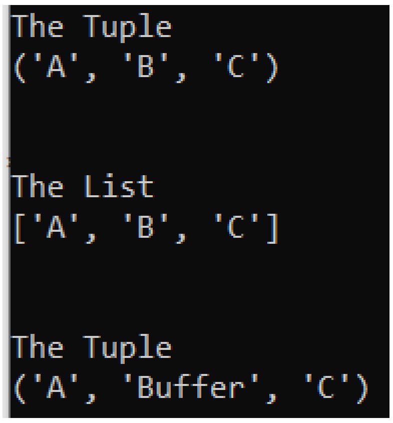
- Tuples are unchangeable, meaning that we cannot change, add, or remove items once the tuple is created.

- But there are some workarounds.

    - You can convert the tuple into a list, change the list, and convert the list back into a tuple.

Example

```
x = ("A", "B", "C")
print("The Tuple")
print(x)

print("\n")
y = list(x)
print("The List")
print(y)

print("\n")

y[1] = "Buffer"


x = tuple(y)
print("The Tuple")
print(x)
```



```
The Tuple
('A', 'B', 'C')


The List
['A', 'B', 'C']


The Tuple
('A', 'Buffer', 'C')
```

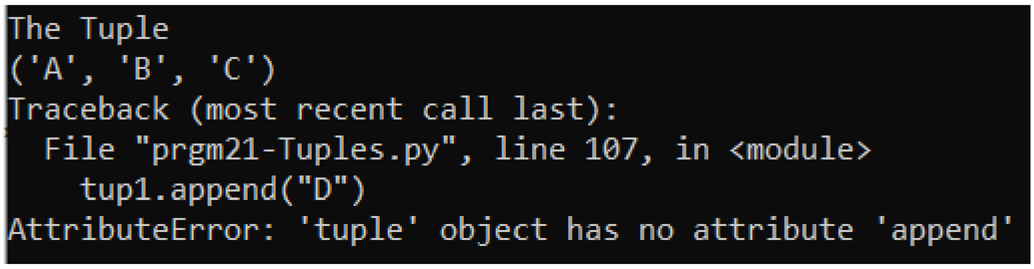
Add Items

- Once a tuple is created, you cannot add items to it.

Example

```
tup1 = ("A", "B", "C")
print("The Tuple")
print(tup1)
```

```
tup1.append("D")
```

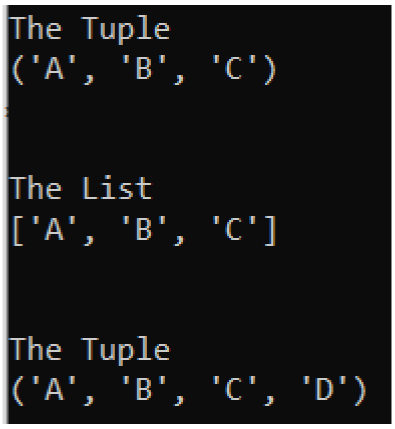


```
The Tuple
('A', 'B', 'C')
Traceback (most recent call last):
  File "prgm21-Tuples.py", line 107, in <module>
    tup1.append("D")
AttributeError: 'tuple' object has no attribute 'append'
```

Just like the workaround for *changing* a tuple, you can convert it into a list, add your item(s), and convert it back into a tuple.

Example

```
x = ("A", "B", "C")
print("The Tuple")
print(x)

print("\n")
y = list(x)
print("The List")
print(y)

print("\n")

y.append ("D")

x = tuple(y)
print("The Tuple")
print(x)
```



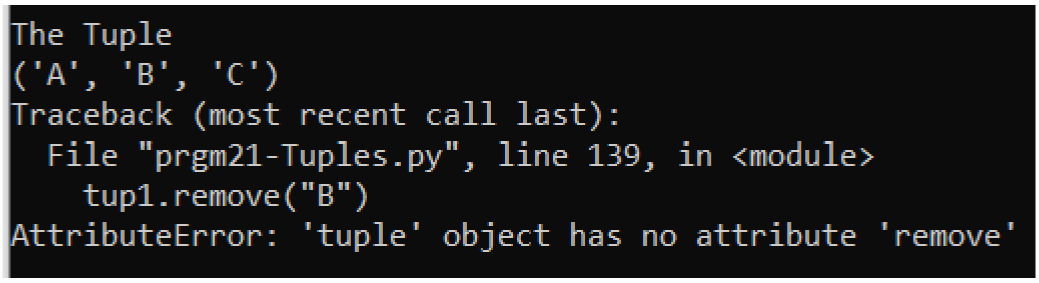
```
The Tuple
('A', 'B', 'C')


The List
['A', 'B', 'C']


The Tuple
('A', 'B', 'C', 'D')
```

Remove Items

- You cannot remove items in a tuple.
- Tuples are **unchangeable**, so we cannot remove items from it.

Example:

```
tup1 = ("A", "B", "C")
print("The Tuple")
print(tup1)

tup1.remove("B")
```

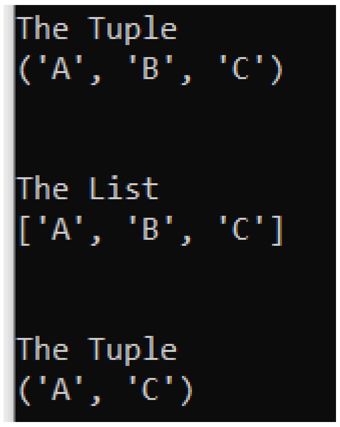


```
The Tuple
('A', 'B', 'C')
Traceback (most recent call last):
  File "prgm21-Tuples.py", line 139, in <module>
    tup1.remove("B")
AttributeError: 'tuple' object has no attribute 'remove'
```

But you can use the same workaround as we used for changing and adding tuple items:

Example:

```
x = ("A", "B", "C")
print("The Tuple")
print(x)
print("\n")
y = list(x)
print("The List")
print(y)
print("\n")
y.remove ("B")
x = tuple(y)
print("The Tuple")
print(x)
```

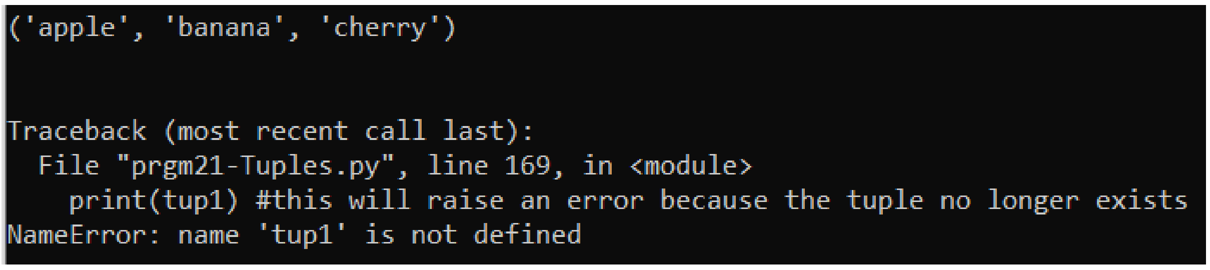


```
The Tuple
('A', 'B', 'C')


The List
['A', 'B', 'C']


The Tuple
('A', 'C')
```

We can also delete the tuple completely using ‘del’ keyword.

Example:

```
tup1 = ("apple", "banana", "cherry")
print(tup1)
print("\n")
del tup1
print(tup1)
```



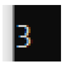
```
('apple', 'banana', 'cherry')


Traceback (most recent call last):
  File "prgm21-Tuples.py", line 169, in <module>
    print(tup1) #this will raise an error because the tuple no longer exists
NameError: name 'tup1' is not defined
```

Tuple Length

- To determine how many items a tuple has, use the len() function:

Example

```
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
```



```
3
```

Looping through Tuples

- We can loop through the tuple items by using a for loop or while loop.

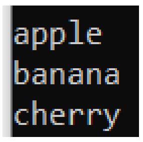
  Example:

  ```
  thistuple = ("apple", "banana", "cherry")
  for x in thistuple:
    print(x)
  ```

- **Loop Through the Index Numbers i.e. we can also loop through the tuple items by referring to their index number (using range() and len() functions).**

  Example:

  ```
  thistuple = ("apple", "banana", "cherry")
  for i in range(len(thistuple)):
    print(thistuple[i])
  ```



```
apple
banana
cherry
```

Using a While Loop

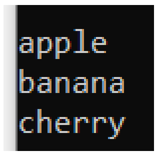
- You can loop through the list items by using a while loop.

- Use the len() function to determine the length of the tuple, then start at 0 and loop through the tuple items by referring to their indexes.

- Remember to increase the index by 1 after each iteration.

Example

```
thistuple = ("apple", "banana", "cherry")
i = 0
while i < len(thistuple):
  print(thistuple[i])
  i = i + 1
```



```
apple
banana
cherry
```