Tuple

- A tuple is an ordered, immutable collection in Python.
- You can't change, add, or remove elements once created.
- Useful for storing related information (like coordinates, colors, database records).
- Tuples are defined using round brackets ( )

In [1]:
```
t = ()
print(t)
```

```
()
```

In [2]:
```
type (t)
```

Out[2]:
```
tuple
```

Tuple Functions:

- count()
- index()

In [6]:
```
t = (10,20,30)
print(t)
```

```
(10, 20, 30)
```

In [7]:
```
t.count(20)
```

Out[7]:
```
1
```

In [8]:
```
t = (1,2,5,8,1,9,3,2,8)
print(t)
```

```
(1, 2, 5, 8, 1, 9, 3, 2, 8)
```

In [9]:
```
t.count(2)
```

Out[9]:
```
2
```

In [10]:
```
t.index(1)
```

Out[10]:
```
0
```

In [11]:
```
t.index(1)
```

Out[11]:
```
0
```

In [13]:
```
t.index(8)
```

Out[13]:
```
3
```

In [14]:
```
print(len(t))
```

```
9
```

In [15]:
```
bank_details = (1234, 'jhd679j',20000)
print(bank_details)
```

```
(1234, 'jhd679j', 20000)
```

In [16]:
```
bank_details.index(20000)
```

Out[16]: 2

In [17]:
```
bank_details.count(1234)
```

Out[17]: 1

In [18]:
```
print(len(bank_details))
```

```
3
```

In [19]:
```
t = (10,20,30)
t
```

Out[19]: (10, 20, 30)

In [20]:
```
t2 = t*2
t2
```

Out[20]: (10, 20, 30, 10, 20, 30)

In [21]:
```
for i in (t):
    print(i)
```

```
10
20
30
```

In [22]:
```
for i in enumerate(t):
    print(i)
```

```
(0, 10)
(1, 20)
(2, 30)
```

Set

- Set is a collection of unique elements.
- it doesn't allow duplicates and is unordered.
- Set is mutable - you can add or remove items, but you can't change items themselves.
- Set = Unique + Unordered + Mutable
- Sets are defined using flower brackets { }

In [23]:
```
s = {}
s
```

Out[23]: {}

In [24]:
```
type(s)
```

Out[24]: dict

In [25]:
```
s = {1,2,3,4,5,6}
print(s)
```

```
{1, 2, 3, 4, 5, 6}
```

In [26]:
```
s1 = {1,2,3,2,5,3,6,5,7,6,8,9,8}
print(s1)
```

```
{1, 2, 3, 5, 6, 7, 8, 9}
```

Adding elements

- add() : Adds a single element
- update() : Adds multiple elements

In [32]:
```
s = {1,2,3}
s.add(4)
s.update([5,6])
print(s)
```

```
{1, 2, 3, 4, 5, 6}
```

Removing elements

- remove() : removes element if present (throws error if element not found).
- discard() : removes element if present or else ignores (it do not throw error if element not found).
- - "remove complains, discard forgives!"
- pop() : Removes random element.
- clear() : Removes all elements.

In [33]:
```
s = {1,2,3,4,5,6,7,8,9}
print(s)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [34]:
```
s.remove(1)
print(s)
```

```
{2, 3, 4, 5, 6, 7, 8, 9}
```

In [35]:
```
s.remove(10)
print(s)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[35], line 1
----> 1 s.remove(10)
      2 print(s)

KeyError: 10
```

In [36]:
```
s.discard(2)
s
```

Out[36]:
```
{3, 4, 5, 6, 7, 8, 9}
```

In [37]:
```
s.discard(10)
s
```

Out[37]:
```
{3, 4, 5, 6, 7, 8, 9}
```

In [38]:
```
s.pop()
s
```

Out[38]:
```
{4, 5, 6, 7, 8, 9}
```

In [39]:
```
s.pop()
s
```

Out[39]:
```
{5, 6, 7, 8, 9}
```

In [44]:
```
s.pop(7)
s
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[44], line 1
----> 1 s.pop(7)
      2 s

TypeError: set.pop() takes no arguments (1 given)
```

In [45]:
```
s.pop()
s
```

Out[45]:
```
{6, 7, 8, 9}
```

In [46]:
```
s.clear()
s
```

Out[46]:
```
set()
```

In [47]:
```
s
```

Out[47]:
```
set()
```

union()

- Combine elements from two sets but keep only unique elements.

- Union gives everyone who used apps(mobile or web!)

In [48]:
```
a = {1,2,3,4,5}
b = {4,5,6,7,8}
c = {8,9,10}
```

In [49]:
```
type(c)
```

Out[49]: set

In [51]:
```
a.union(b)
```

Out[51]: {1, 2, 3, 4, 5, 6, 7, 8}

In [52]:
```
a.union(b,c)
```

Out[52]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In [53]:
```
b.union(c)
```

Out[53]: {4, 5, 6, 7, 8, 9, 10}

In [54]:
```
a.union(c)
```

Out[54]: {1, 2, 3, 4, 5, 8, 9, 10}

short cut way union operator using |

In [55]:
```
a | b
```

Out[55]: {1, 2, 3, 4, 5, 6, 7, 8}

In [56]:
```
a | b | c
```

Out[56]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In [57]:
```
b | c
```

Out[57]: {4, 5, 6, 7, 8, 9, 10}

intersection()

- prints a new set with common elements present in both sets.

In [58]:
```
a = {1,2,3,4,5}
b = {4,5,6,7,8}
c = {8,9,10}
```

In [59]:
```
a.intersection(b)
```

Out[59]: {4, 5}

In [60]:
```
b.intersection(c)
```

Out[60]:
```
{8}
```

In [61]:
```
a.intersection(b,c)
```

Out[61]:
```
set()
```

In [62]:
```
a.intersection(c)
```

Out[62]:
```
set()
```

short cut way intersection operator using &

In [63]:
```
a = {1,2,3,4,5}
b = {4,5,6,7,8}
c = {8,9,10}
```

In [64]:
```
a & b
```

Out[64]:
```
{4, 5}
```

In [65]:
```
b & c
```

Out[65]:
```
{8}
```

difference()

- prints the elements that are present in first set but not in the second set.

In [66]:
```
a = {1,2,3,4,5}
b = {4,5,6,7,8}
c = {8,9,10}
```

In [67]:
```
a.difference(b)
```

Out[67]:
```
{1, 2, 3}
```

In [68]:
```
b.difference(c)
```

Out[68]:
```
{4, 5, 6, 7}
```

In [69]:
```
b.difference(a)
```

Out[69]:
```
{6, 7, 8}
```

In [70]:
```
c.difference(b)
```

Out[70]:
```
{9, 10}
```

In [71]:
```
c.difference(a)
```

Out[71]:
```
{8, 9, 10}
```

In [72]:
```
a.difference(c)
```

Out[72]:
```
{1, 2, 3, 4, 5}
```

short cut way difference operator using -

In [73]:
```
a - b
```

Out[73]:
```
{1, 2, 3}
```

In [74]:
```
b - a
```

Out[74]:
```
{6, 7, 8}
```

In [ ]: