Tuple

- A tuple is a collections of items in python that is **ordered**, **unchangeable(immutable)**, and allow **duplicate values**.

ordered

- Tuple items have a defined order, but that order will not change.

- list and tuples are almost same only list is mutable or changeable
- but tuple are immutable unchangeable

- It is denoted as parenthesis ()
- inside parenthise we can stored elements
- we cant give parentheis () directly comma-seprated also use it
  - for ex:- num = 1,2,3,'abc'

- intilaization
- inbuilt functions
- 1. min
- 2. max
- 3. len
- 4. sorted
- 5. reversed
- 6. print
- 7. type
- 8. sum
- 9. index operations (for loop range vs in)
- concatenation
- mutable vs immutable
- slicing

##

Methods of Tuples

- count
- index

In [11]:
```
t1 = ('apple', 'banana', 'cherry', 'orange')
t1
```

Out[11]: ('apple', 'banana', 'cherry', 'orange')

In [24]:
```
type(t1)
```

Out[24]: tuple

In [25]:
```
t2 = ('red', 'blue', 'brown', 'green', 'yellow')

t2, type(t2)
```

Out[25]: (('red', 'blue', 'brown', 'green', 'yellow'), tuple)

In [26]:
```
t3 = (12,23,43,546,32)
t3, type(t3)
```

Out[26]: ((12, 23, 43, 546, 32), tuple)

In [27]:
```
t4 = (12,32,'red', 'orange', 12.32, 32+12j, True, False)
t4, type(t4)
```

Out[27]: ((12, 32, 'red', 'orange', 12.32, (32+12j), True, False), tuple)

In [1]:
```
t5 = (12,23,43,12, (12,34,True), [12,23,43])
t5
```

Out[1]: (12, 23, 43, 12, (12, 34, True), [12, 23, 43])

In [29]:
```
type(t5)
```

Out[29]: tuple

In [30]:
```
t6 = 1,2,3,4,5,'abc'
type(t6)
```

Out[30]: tuple

In [31]:
```
t6
```

Out[31]: (1, 2, 3, 4, 5, 'abc')

In [34]:
```
t7 = tuple(('apple', 'banana'))
t7
```

Out[34]: ('apple', 'banana')

In [1]:
```
l1 = (12,23,34,342,1233)
tuple(l1)
```

Out[1]: (12, 23, 34, 342, 1233)

In [17]:
```
single_tuple = ("hello") # Here we not added , so this is string
type(single_tuple)
```

Out[17]: str

In [18]:
```
single_tuple = ("hello",) # Here we not added , so this is tuple
type(single_tuple)
```

Out[18]: tuple

In [35]:
```
t1
```

Out[35]: ('apple', 'banana', 'cherry', 'orange')

min and max

In [40]:
```
max(t1),max(t2),max(t3),max(t7)
```

Out[40]: ('orange', 'yellow', 546, 'banana')

- not work - t3,t4,t5

In [41]:
```
min(t1),min(t2), min(t3), min(t7)
```

Out[41]: ('apple', 'blue', 12, 'apple')

- not work
- t3,t4,t5

len

In [42]:
```
len(t1), len(t2), len(t3), len(t4), len(t5), len(t6), len(t7)
```

Out[42]: (4, 5, 5, 8, 5, 6, 2)

sorted

In [46]:
```
t2
```

Out[46]: ('red', 'blue', 'brown', 'green', 'yellow')

In [45]:
```
sorted(t2)
```

Out[45]: ['blue', 'brown', 'green', 'red', 'yellow']

Note

- when we apply sorted in tuple
- the output comes in list form

In [49]:
```
sorted(t2,key=len)
```

Out[49]: ['red', 'blue', 'brown', 'green', 'yellow']

In [50]:
```
sorted(t2,key=len, reverse=True)
```

Out[50]:
```
['yellow', 'brown', 'green', 'blue', 'red']
```

reversed

In [52]:
```
t1
```

Out[52]:
```
('apple', 'banana', 'cherry', 'orange')
```

In [13]:
```
list(reversed(t1))
```

Out[13]:
```
['orange', 'cherry', 'banana', 'apple']
```

sum

In [55]:
```
tup = (123,23,435,546,75)
sum(tup)
```

Out[55]:
```
1202
```

In [56]:
```
tup = (123,23,435,546,75)
sum(tup,start=100)
```

Out[56]:
```
1302
```

index operations (for loop range vs in)

In [57]:
```
t1
```

Out[57]:
```
('apple', 'banana', 'cherry', 'orange')
```

In [62]:
```
t3[0], t3[1]
```

Out[62]:
```
(12, 23)
```

In [64]:
```
t1[0],t2[0]
```

Out[64]:
```
('apple', 'red')
```

In [63]:
```
t1[0]
t1[1]
t1[2]
t1[3]

# what is common t1[]
# what is changing i
```

Out[63]:
```
'orange'
```

In [60]:
```
for i in t1:
    print(i)
```

```
apple
banana
cherry
orange
```

In [61]:
```
for i in range(len(t1)):
    print(t1[i], i)
```

```
apple 0
banana 1
cherry 2
orange 3
```

concatenation

In [65]:
```
t1 = (100,200,300)
t2 = ("A", "B", "C")

t1+t2
```

Out[65]: (100, 200, 300, 'A', 'B', 'C')

In [70]:
```
t1 = (100,200,300)
t2 = ("A", "B", "C")

t2*2
```

Out[70]: ('A', 'B', 'C', 'A', 'B', 'C')

mutable vs immutable

In [83]:
```
t1 = (12,23,43,54,65)
t1
```

Out[83]: (12, 23, 43, 54, 65)

In [78]:
```
str1 = 'w,e,r,t,ty,er'
str1
```

Out[78]: 'w,e,r,t,ty,er'

In [79]:
```
str1[0]= 'er'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[79], line 1
----> 1 str1[0]= 'er'

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

In [84]:
```
t1[3] = 'a'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[84], line 1
----> 1 t1[3] = 'a'

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

- In tuple we can't change the value of using index
- so it is immutable

slicing

In [91]:
```
t1 = ('A', 'B', 'C',12, 23, 43, 546, 32,12, 32, 'red', 'orange', 12.32, (32+12j))
```

In [95]:
```
t1[2:-10], t1[2:]
```

Out[95]:
```
(('C', 12, 23, 43),
 ('C',
  12,
  23,
  43,
  546,
  32,
  12,
  32,
  'red',
  'orange',
  12.32,
  (32+12j),
  True,
  False))
```

In [99]:
```
t1[:] # start to end
t1[::] # start to end
t1[::-1] # reverse the tuple elements
```

Out[99]:
```
(False,
 True,
 (32+12j),
 12.32,
 'orange',
 'red',
 32,
 12,
 32,
 546,
 43,
 23,
 12,
 'C',
 'B',
 'A')
```

In [100…
```
t8 = ()
type(t8)
```

Out[100…
```
tuple
```

In [101…
```
len(t8)
```

Out[101…
```
0
```

Methods

- count
- index

count

In [104…
```
t1 = (12,12,23,34,345,5,65,12,12,12,342,23,23,12)
t1
```

Out[104…
```
(12, 12, 23, 34, 345, 5, 65, 12, 12, 12, 342, 23, 23, 12)
```

In [105…
```
t1.count(12)
```

Out[105…
```
6
```

In [109…
```
t1.count(34)
```

Out[109…
```
1
```

In [110…
```
t1.count(100)
```

Out[110…
```
0
```

index

In [111…
```
t1.index(12)
```

Out[111…
```
0
```

In [123…
```
i1 = t1.index(12)
i2 = t1.index(12,i1+1)
i3 = t1.index(12,i2+1)
i4 = t1.index(12,i3+1)
i5 = t1.index(12,i4+1)
i6 = t1.index(12,i5+1)
#i7 = t1.index(12,i6+1) # here error will comes
i1,i2,i3,i4,i5,i6
```

Out[123…
```
(0, 1, 7, 8, 9, 13)
```

Packing and Unpacking

- packing is the process of putting multiple values into a single tuple.
- Unpacking is extracting the values from a tuple into separate variables

In [126…
```
a = 'python'
b = 29
c = 'Doctor'

tuple_pack = a,b,c
print(tuple_pack)
```

```
('python', 29, 'Doctor')
```

In [125…
```
type(tuple_pack)
```

Out[125…
```
tuple
```

In [129…
```
name, age, profesion = tuple_pack
print(name)
print(age)
print(profesion)
```

```
python
29
Doctor
```

In [130…
```
type(name)
```

Out[130…
```
str
```

In [131…
```
type(age)
```

Out[131…
```
int
```

In [132…
```
type(profesion)
```

Out[132…
```
str
```

In [ ]:

In [5]:
```
t8 = ("h",)
min(t8)
```

Out[5]:
```
'h'
```

In [6]:
```
t2 = ("apple", "mango", "orange", "banana")
sorted(t2,key=len)
```

Out[6]:
```
['apple', 'mango', 'orange', 'banana']
```

In [7]:
```
t2 = ("apple", "mango", "orange", "banana")
sorted(t2)
```

Out[7]:
```
['apple', 'banana', 'mango', 'orange']
```

In [8]:
```
t2 = ("apple", "mango", "orange", "banana")
sorted(t2,key=len, reverse=True)
```

Out[8]:
```
['orange', 'banana', 'apple', 'mango']
```

In [9]:
```
t2 = ("apple", "mango", "orange", "banana")
sorted(t2,reverse=True)
```

Out[9]:
```
['orange', 'mango', 'banana', 'apple']
```

In [ ]: