(TP1 - 1)

↳
tp1 = ( )
tp2 = (2, 4, 6, 8, 10)
bool (tp1) → False
bool (tp2) → True

↳ tp1 = [2, 3, [4, 5, 6], 10)
tp1 [2] [2] = 6

↳ tp1 = (2, 3, 4, 5)
tp1 [1] = 50 ✗ Type Error

↳ tp1 = (2, 3, [4, 5, 6], 7, 8)
tp1 [2] [2] = 22
(2, 3, [4, 5, 22], 7, 8)

↳ Tuples are immutable, so less operation
index, count

↳ tp1 = (2, 4, 6, 8, 10, "python", 8)
tp1.count(5) #0
tp1.count(8) #2
tp1.index(10) # 4
tp1.index(2,3) # ValueError
index(8,4) #6

X = [4, 5, 6, ...

len(X) #5

↳ x = [2, 3, 6]
len(x) #3

// string, list, tuple the len function is same

Index based

tp1 = (2, 3, 5, 6, 7, 8)
tp1[:] #(2, 3, 5, 6, 7, 8)
tp1[0:] # (2, 3, 5, 6, 7, 8)
tp1[3:] #(6, 7, 8)

x = 5,
type(x) → Tuple

↳ x = 5, 8, 9
type(x) → class 'tuple'

Data chansing take list, data chansing not reswi
→ tuple

ABdW)

t1 = (1, 2, 3, 4, 5, 6)
type(t1) → tuple

t2 = tuple([1, 2, 3, 4, 5])
type(t2)
tuple

t3 = tuple('python')
# ('p', 'y', 't', 'h', 'o', 'n')

t4 = tuple(range(1, 5))
(1, 2, 3, 4)

t5 = ()
type(t5)
tuple

t6 = (5)
type(t6)
<int>

t7 = (5,)
type(t7)
tuple

t8 = 10,11,12,13,14
tuple

tp1-2

Traverse

t1 = (6,5,4,3,2,1)
t1[0] # 6
t1[-1] # 1
t1[-3] # 3
t1[2:5] # (4,3,2)

t1[2] = 10
// TypeError

for i in t1:
    pt(i, end=" ")
6,5,4,3,2,1

Comprehension

```
# T = (iterable)
# for x in range(1,5):
#     pt(x)
# T = (x for x in range(1,5))
type(T)
generator
T = (* (x for x in range(1,5)),)
Tuple
(1,2,3,4)
T2 = tuple(x for x in range(1,5))
(1,2,3,4)
T3 = tuple(x ** 2 for x in range(1,5))
(1,4,9,16)
T4 = tuple(x.lower() for x in "pytON")
'p','y','t','h','o','n'
T5 = tuple(int(x) for x in "12345")
(1,2,3,4,5)
T6 = tuple(x for x in "ab*#%6"
    if x.is
('a','b')
```

Tpl-3

Read

↳ Indexing
↳ slicing

t1 = (3, 6, 9, 12, 15, 18, 21)

t1[4]
#15
t1[-3]
#15

↳ t1[::]
t1[2:]
(9, 12, 15, 18, 21)
t1[2:5]
(9, 12, 15)
t1[-5:-2])
(9, 12, 15)

↳ t1[::2]
↳ (3, 9, 15, 21)
t1[::-1]
↳ t1[4::-1]
(15, 12, 9, 6, 3)
↳ t1[4:0:-1])
(15, 12, 9, 6)
↳ t1[-3:-7:-1])
(15, 12, 9, 6)

Packing / unpacking

Concatination
+
Repetation
*
packing / unpacking
*
Membership
in # not in

T1 = (1, 2, 3)
T2 = (8, 9, 10)
T1 + T2
(1, 2, 3, 8, 9, 10)

T1 = (1, 2, 3)
T1 * 5
(1, 2, 3 ... 1, 2, 3)

→ Packing / unpacking

T1 = 1, 2, 3, 4, 5
type(T1) :- class tuple
a, b, c, d, e = T1
multiple variables for single variable of tuple
1, 2, 3, 4, 5
a b c d e

↳ a, b, c = T1 // value error
a, b, *c = T1
a = 1
b = 2
c = [3, 4, 5]

a, *b, c = T1
a = 1
b = [2, 3, 4]
c = 5

*a, b, c = T1
a = [1, 2, 3]
b = 4
c = 5

↳ Membership

t1 = [1, 2, 3, 4, 5]
1 in t1 → True
3 not in t1 → False