Tuple Introduction:

1. What is tuple?
2. Creation
3. Representation
4. Immutable
5. Traverse
6. What is tuple?

t1=(1,2,3,4,5,6)

t2=(1.8,1.8,1.8,3.5)

t3=(“Python”,”john”,”anil”)

t4=(7,3.2,”John”,True,5+6j)

* Tuple is same as List but immutable
* Ordered elements
* Duplicates
* Heterogenous

No change in tuple length (no element addition), Immutable (no change of data)

1. Creation

T1=(1,2,3,4,5,6)

T2=tuple(iterable)

Tuple([1,2,3,4,5,6])

T3=()

T4=(3) 🡪 int

(3,)🡪 tuple

T5=10,11,12,13,15

Prog1:

t1=(1,2,3,4,5,6)

print(type(t1))

# <class 'tuple'>

print(t1)

#(1, 2, 3, 4, 5, 6)

t2=tuple([1,2,3,4,5,6])

print(type(t2))

#<class 'tuple'>

print(t2)

#(1, 2, 3, 4, 5, 6)

t3=tuple("python")

print(type(t3))

# ('p', 'y', 't', 'h', 'o', 'n')

print(t3)

# <class 'tuple'>

t4=tuple(range(1,5))

print(type(t4))

#<class 'tuple'>

print(t4)

# (1, 2, 3, 4)

t5=()

print(type(t5))

#<class 'tuple'>

print(t5)

# ()

t6=(5)

print(type(t6))

# <class 'int'>

t7=(5,)

print(type(t7))

# <class 'tuple'>

print(t7)

# (5,)

t8=10,11,12,13,14

print(type(t8))

# <class 'tuple'>

print(t8)

# (10, 11, 12, 13, 14)

TRAVERSE:

############# TRAVERSE

t1=(6,5,4,3,2,1)

print(t1[0])

#6

print(t1[-1])

#1

print(t1[-3])

#3

print(t1[2:5])

#(4, 3, 2)

# t1[2]=10

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

for i in t1:

    print(i,end=" ")

#6 5 4 3 2 1

#

print()

Tuple comprehension and methods:

#T=(iterable)

#for x in range(1,5):

#     print(x)

#1,2,3,4,5

T=(x for x in range(1,5))

print(type(T))

# <class 'generator'>

T=(\*(x for x in range(1,5)),)

print(type(T))

#<class 'tuple'>

print(T)

#(1, 2, 3, 4)

T2=tuple(x for x in range(1,5))

print(type(T2))

#class 'tuple'>

print(T2)

#(1,2,3,4)

T3=tuple(x\*\*2 for x in range(1,5))

print(type(T3))

#<class 'tuple'>

print(T3)

#(1, 4, 9, 16)

T4=tuple(  x.lower() for x in "PyThoN")

print(type(T4))

#<class 'tuple'>

print(T4)

#('p', 'y', 't', 'h', 'o', 'n')

T5=tuple(int(x) for x in "12345")

print(type(T5))

#<class 'tuple'>

print(T5)

#(1, 2, 3, 4, 5)

#

T6=tuple(x for x in "ab\*#%6" if x.isalpha())

print(type(T6))

#<class 'tuple'>

print(T6)

#('a', 'b')

Slicing and Indexing

#Read

    #indexing

    #slicing

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

print(t1[4])

#15

print(t1[-3])

#15

print(t1[::])

#(3, 6, 9, 12, 15, 18, 21)

print(t1[2:])

#(9, 12, 15, 18, 21)

print(t1[2:5])

#(9, 12, 15)

print(t1[-5:-2])

#(9, 12, 15)

print(t1[::2])

#(3, 9, 15, 21)

print(t1[::-1])

#

print(t1[4::-1])

#(15, 12, 9, 6, 3)

print(t1[4:0:-1])

#(15, 12, 9, 6)

print(t1[-3:-7:-1])

#(15, 12, 9, 6)

Packing and Unpacking:

#Concatination

            # +

# Repetation:

             # \*

# Packing and unpacking:

         # \*

#membership

    # in , not in

T1=(1,2,3)

T2=(8,9,10)

print(T1+T2)

#  (1, 2, 3, 8, 9, 10)

#Repetation

T1=(1,2,3)

print(T1\*5)

# (1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3)

#packing and unpacking

T1=1,2,3,4,5

print(type(T1))

#single variable for multiple variables as tuple

# <class 'tuple'>

a,b,c,d,e=T1

#multiple variables for single variable as tuple

print(a,b,c,d,e)

# (1 2 3 4 5)

############

# a,b,c=T1 #ValueError: too many values to unpack (expected 3)

a,b,\*c=T1

print(a,b,c)

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

a,\*b,c=T1

print(a,b,c)

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

\*a,b,c=T1

print(a,b,c)

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

#membership

t1=[1,2,3,4,5]

print(1 in t1)

#True

print(3 not in t1)

#False