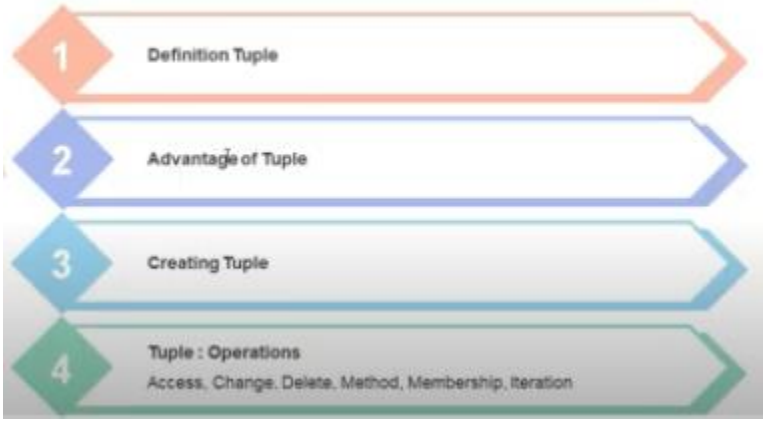
Python Tuple

Agenda



Definition Tuple

- Tuple adalah urutan item, mirip dengan list
- Perbedaan antara list dan tuple adalah:
  Tuple immutable sementara list mutable
- Diapit tanda kurung
- Tuple dengan satu elemen harus memiliki koma di dalam tanda kurung:
  a = (11,)

Advantages of Tuple over List

- Tuple digunakan untuk tipe data heterogen (berbeda) dan list untuk tipe data homogen (serupa).
- Tuple immutable, iterasi melalui tupel lebih cepat daripada dengan list.
- Tupel yang berisi elemen yang tidak dapat diubah dapat digunakan sebagai kunci untuk dictionary sedangkan list tidak bisa.
- Apabila data tidak berubah, mengimplementasikannya sebagai tuple akan menjamin bahwa data tersebut tetap dilindungi dari penulisan

Creating Tuple

- Tuple dibuat dengan menempatkan semua item di dalam tanda kurung, dipisahkan dengan koma
- sebuah tuple dapat berupa integer, float, list, string dll

```
# empty tuple
my_tuple = ()
print(my_tuple)

# tuple having integers
my_tuple = (1, 2, 3)
print(my_tuple)

# tuple with mixed datatypes
my_tuple = (1, "Hello", 3.4)
print(my_tuple)

# nested tuple
my_tuple = ("mouse", [8, 4, 6], (1, 2, 3))
print(my_tuple)

# tuple can be created without parentheses
# also called tuple packing
my_tuple = 3, 4.6, "dog"
print(my_tuple)
```

```
()
(1, 2, 3)
(1, 'Hello', 3.4)
('mouse', [8, 4, 6], (1, 2, 3))
(3, 4.6, 'dog')
```

Why ?

- ❑ Tidak terjadi kebingungan antara [1] dan 1
- ❑ (1) adalah adalah ekspresi yang dapat diterima dengan sempurna.
  - (1) tanpa koma adalah integer 1
  - (1,) dengan koma adalah list yang berisi integer 1

Example of Tuple

Try it out !

```
>>> x = ("1", 2, [3])
>>> print(x)
????
>>> type(x)
????
```

2

```
>>> a = "1"
>>> b = 2
>>> c = [3]
>>> y = a, b, c
>>> print(y)
????
>>> type(y)
????
```

3

```
>>> a = "1"
>>> y = (a)
>>> type(y)
????
>>> a = "1"
>>> y = (a,)
>>> type(y)
????
>>> print(y)
????
>>> print(y[1])
????
>>> len(y)
```

Accessing Element in a Tuple

- Dapat mengakses nilai dalam tuple melalui indeks,
- Nested tuple diakses menggunakan pengindeksan bersarang
- Pengindeksan negatif dapat diterapkan ke tuple yang mirip dengan list
- Dapat mengakses berbagai item dalam tuple dengan menggunakan potongan operator
- Apabila ingin menetapkan kembali nilai dalam tupel menyebabkan TypeError

Accessing Element in a Tuple ...

```
#tuple
simple_tuple=(23, 45, 32)

print(simple_tuple[0])
print(simple_tuple[2])
```

```
23
32
```

```
#nested tuple
nested_tuple = ('Algoritma Pemrograman', [8,4,6], (1,2,3))

print(nested_tuple[0])
print(nested_tuple[1])
print(nested_tuple[0][0])
print(nested_tuple[1][0])
```

```
Algoritma Pemrograman
[8, 4, 6]
A
8
```

Changing a Tuple

- Tidak seperti list, tuple immutable
- Element dalam tuple tidak dapat diubah. Tetapi jika elemen itu sendiri merupakan data tipe mutable seperti list, elemen atau itemnya nested maka dapat diubah.

```
#nested tuple
nested_tuple = ('Algoritma Pemrograman', [8,4,6], (1,2,3))
print(nested_tuple)
nested_tuple[1][1] = 23
print(nested_tuple)
```

```
('Algoritma Pemrograman', [8, 4, 6], (1, 2, 3))
('Algoritma Pemrograman', [8, 23, 6], (1, 2, 3))
```

Changing a Tuple …

Mirip seperti List,

- Penggunaan operator + untuk mengkombinasikan 2 tuple.
- Dapat juga menggunakan elemen **repeat** dalam tuple untuk memberikan kelipatan angka dari penggunaan operator *
- + dan * menghasilkan tuple baru

```
# Concatenation
print((1,2,3)+(4,5,))

#Repeat
print(('Repeat',)*3)

(1, 2, 3, 4, 5)
('Repeat', 'Repeat', 'Repeat')
```

Deleting a Tuple

- Elemen Tuple immutable maka elemen tersebut tidak dapat dihapus pula.
- Tetapi menghapus Tuple secara keseluruhan dapat dilakukan dengan menggunakan kata kunci del

```
#tuple
simple_tuple=(23, 45, 32)

del simple_tuple[3]
#TypeError: 'tuple' object doesn't support item deletion

del simple_tuple
#can delete entire tuple

simple_tuple
#NameError: name 'simple_tuple' is not defined
```

Python Tuple Methods

Insert the title of your subtitle Here

Metode untuk menambahkan item atau menghapus item tidak tersedia dengan Tuple. hanya dua metode berikut yang tersedia

Method	Description
count(x)	Mengembalikan jumlah item yang sama dengan x
index(x)	Mengembalikan index item pertama yang sama dengan x

```
#tuple
simple_tuple=('a', 'l','g','o','r','i','t','m','a',)

print('total jumlah dari element a adalah' , simple_tuple.count('a'))
print('index dari t adalah ', simple_tuple.index('t'))

total jumlah dari element a adalah 2
index dari t adalah  6
```

Python Tuple Methods

- Menguji keberadaan elemen / item dalam Tuple

```
#tuple
simple_tuple=('a', 'l','g','o','r','i','t','m','a',)

print('a' in simple_tuple)
print('l' in simple_tuple)
print('b' in simple_tuple)
print('o' in simple_tuple)
print('s' in simple_tuple)
```

```
True
True
False
True
False
```

Iterating Through a Tuple

- Penggunaan pengulangan for dapat melakukan iterasi melalui setiap elemen dalam Tuple.

```
nama2 = ('Bapak Sigit','Bapak Khozaimi','Bapak Doni', 'Bapak Yudha','Ibu Imamah','Ibu Fifin')
for nama in nama2:
  print('Assalamualaikum wr wb ',nama)
```

```
Assalamualaikum wr wb  Bapak Sigit
Assalamualaikum wr wb  Bapak Khozaimi
Assalamualaikum wr wb  Bapak Doni
Assalamualaikum wr wb  Bapak Yudha
Assalamualaikum wr wb  Ibu Imamah
Assalamualaikum wr wb  Ibu Fifin
```

Built-in Functions with Tuple

Function	Description
all()	Return True if all elements of the tuple are true (or if the tuple is empty).
any()	Return True if any element of the tuple is true. If the tuple is empty, return False.
enumerate()	Return an enumerate object. It contains the index and value of all the items of tuple as pairs.
len()	Return the length (the number of items) in the tuple.
max()	Return the largest item in the tuple.
min()	Return the smallest item in the tuple
sorted()	Take elements in the tuple and return a new sorted list (does not sort the tuple itself).
sum()	Retrun the sum of all elements in the tuple.
tuple()	Convert an iterable (list, string, set, dictionary) to a tuple.

```
# all() = retuen true if all element of the tuple are true
t1 = (2, 1, 3)
print(t1)
```

```
(2, 1, 3)
```

```
# any() = return true if any element of the tuple is true . if tuple empty return false
t2 = ("algoritma pemrograman", 2, 3, [3,5])
t3 = ()
print(any(t1))
print(any(t2))
print(any(t3))
print(5 in t2)
print(5 not in t2)
```

```
True
True
False
False
True
```

```
# Enumerate()= return a enumarate object
t4=("mawar", "melati", "anggrek","matahari", "lily")
for i in enumerate(t4):
  print(i)
```

```
(0, 'mawar')
(1, 'melati')
(2, 'anggrek')
(3, 'matahari')
(4, 'lily')
```

```
# len()=return the length
print(len(t1))
print(len(t2))
print(len(t3))
print(len(t4))
```

```
3
4
0
5
```

```
# max(), min()
t5 = (3,14,5,23,9,17,14,57,29,55)
print(max(t5))
print(min(t5))
```

```
57
3
```

```
# sorted() = take elements in tuple an return  a new sorted list (does not sort the tuple itself)
t1 = (2, 1, 3)
print(sorted(t1))
print(sorted(t5))
```

```
[1, 2, 3]
[3, 5, 9, 14, 14, 17, 23, 29, 55, 57]
```

```
# sum() = return the sum of all elements in the tuple
print(sum(t1))
print(sum(t5))
```

```
6
226
```

```
# tuple

mylist=[1, 5, "alpro"]
print(mylist)
print(tuple(mylist))
```

```
[1, 5, 'alpro']
(1, 5, 'alpro')
```