LAPORAN

LAPORAN PRATIKUM PERTEMUAN 2 STRRUKTUR DATA

Dibuat oleh :

Fatih Hamzah Zulkarnain

25071306040

TI-B

Dosen Pengampu :

Reny Fitri Yani, S.T, M.T

PRODI S1 TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS RIAU

2025

Kata Pengantar

Puji Syukur telah saya panjatkan kepada tuhan yang maha esa yang telah memberikan saya kesempatan untuk menyelesaikan tugas laporan pratikum pertemuan 2 ini.
Tujuan saya membuat laporan ini Adalah untuk mendalami ilmu yang telah saya dapatkan pada saat pratikum.
Saya ucapkan terimakasih kepada ibu Reny Fitri Yani,S.T,M.T selaku dosen mata kuliah struktur data dan asisten dosen yang telah memberikan tugas laporan sehingga dapat memperdalam pengetahuan saya.
Saya menyadari bahwa lapporan ini masih sangat jauh dari sempurna,maka dari itu saya selaku penulis laporan meminta maaf atas ketidak sempurnaan lsporsn ini.akhir kata semoga laporan ini memberikan manfaat bagi penulis maupun pembaca.

Sabtu,7 februari 2026

Fatih Hamzah Zulkarnain

Tugas Pratikum 2 : Menjelaskan Tiap-tiap kode yang telah diuji saat pratikum

LIST

Python List

List adalah struktur data pada python yang mampu menyimpan lebih dari satu data, seperti array.

Contoh :

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

Maka akan menghasilkan output :

```
['apple', 'banana', 'cherry']
```

Walaupun dalam list tersebut ada nama item yang sama,maka nama item yang sama tetap muncul.

Contoh :

```
thislist = ["apple", "banana", "cherry", "apple",
"cherry"]

print(thislist)
```

Output :

```
['apple', 'banana', 'cherry', 'apple', 'cherry']
```

Untuk menentukan berapa banyak item yang dimiliki sebuah daftar, gunakan len().

Contoh :

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

Output :

```
3
```

Lalu item dalam daftar list dapat berupa tipe data apa pun

Contoh :

```
list1 = ["apple", "banana", "cherry"]
list2 = [1, 5, 7, 9, 3]
list3 = [True, False, False]

print(list1)
print(list2)
print(list3)
```

Output :

```
['apple', 'banana', 'cherry']
[1, 5, 7, 9, 3]
[True, False, False]
```

Dalam satu daftar list bisa berupa berbagai tipe data :

```
list1 = ["abc", 34, True, 40, "male"]

print(list1)
```

Output :

```
['abc', 34, True, 40, 'male']
```

Access items

Item dalam list bis akita munculkan dengan cara memasukkan nomor indeksnya,dimulai dari 0:

```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

Output :

```
banana
```

Selanjutnya kalua angka indeksnya negative maka dihitung dari belakang,seperti -1 maka satu dari belakang, -2 dua dari belakang begitu seterusnya:

```
thislist = ["apple", "banana", "cherry"]
print(thislist[-1])
```

Output:

```
cherry
```

Lalu dapat menentukan rentang indeks dengan menentukan di mana rentang tersebut dimulai dan di mana rentang tersebut berakhir.

```
thislist = ["apple", "banana", "cherry", "orange",
"kiwi", "melon", "mango"]
print(thislist[2:5])
```

Output :

```
['cherry', 'orange', 'kiwi']
```

Begitu juga dengan angka indeksnya negative seperti :

```
thislist = ["apple", "banana", "cherry", "orange",
"kiwi", "melon", "mango"]
print(thislist[-4:-1])
```

Output :

```
['orange', 'kiwi', 'melon']
```

Untuk menentukan apakah suatu item tertentu ada dalam sebuah daftar, gunakan inkata kunci:

```
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
  print("iya, 'apple' termasuk dalam list")
```

Output :

```
iya, 'apple' termasuk dalam list
```

Change List items:

Untuk mengubah nilai item tertentu, lihat nomor indeksnya:

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"

print(thislist)
```

Output :

```
['apple', 'blackcurrant', 'cherry']
```

Untuk mengubah nilai item dalam rentang tertentu, definisikan daftar dengan nilai baru, dan rujuk ke rentang nomor indeks tempat Anda ingin memasukkan nilai baru:

```
thislist = ["apple", "banana", "cherry", "orange",
"kiwi", "mango"]

thislist[1:3] = ["blackcurrant", "watermelon"]

print(thislist)
```

```
['apple', 'blackcurrant', 'watermelon', 'orange', 'kiwi', 'mango']
```

Untuk menyisipkan item daftar baru, tanpa mengganti nilai yang sudah ada, kita dapat menggunakan insert().Untuk menyisipkan item daftar baru, tanpa mengganti nilai yang sudah ada, kita dapat menggunakan insert():

```
thislist = ["apple", "banana", "cherry"]

thislist.insert(2, "watermelon")

print(thislist)
```

Output :

```
['apple', 'banana', 'watermelon', 'cherry']
```

Add List Items

Untuk menambahkan item ke akhir daftar, gunakan append() metode berikut:

```
thislist = ["apple", "banana", "cherry"]

thislist.append("orange")

print(thislist)
```

Output :

```
['apple', 'banana', 'cherry', 'orange']
```

Untuk menyisipkan item daftar pada indeks tertentu, gunakan insert()metode tersebut. Metode ini insert()menyisipkan item pada indeks yang ditentukan

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)
```

Output :

```
['apple', 'orange', 'banana', 'cherry']
```

Untuk menambahkan elemen dari *daftar lain* ke daftar saat ini, gunakan extend() metode tersebut

```
thislist = ["apple", "banana", "cherry"]
tropical = ["mango", "pineapple", "papaya"]

thislist.extend(tropical)

print(thislist)
```

Output :

```
['apple', 'banana', 'cherry', 'mango', 'pineapple', 'papaya']
```

Remove List Items

Metode ini remove()menghapus item yang ditentukan.

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

Output :

```
['apple', 'cherry']
```

Jika terdapat lebih dari satu item dengan nilai yang ditentukan, remove()metode ini akan menghapus item yang pertama muncul:

```
thislist = ["apple", "banana", "cherry", "banana",
"kiwi"]
thislist.remove("banana")
print(thislist)
```

Output :

```
['apple', 'cherry', 'banana', 'kiwi']
```

Metode pop() menghapus indeks yang ditentukan.

```
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)
```

Output :

```
['apple', 'cherry']
```

Kata kunci del juga menghapus indeks yang ditentukan:

```
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```

Output :

```
['banana', 'cherry']
```

Kata kunci tersebut deljuga dapat menghapus daftar sepenuhnya.

```
thislist = ["apple", "banana", "cherry"]
del thislist
print(thislist)
```

Output :

```
Traceback (most recent call last):
  File "demo_list_del2.py", line 3, in <module>
    print(thislist) #this will cause an error because
NameError: name 'thislist' is not defined
```

Metode clear() mengosongkan daftar.

Daftar itu masih ada, tetapi tidak berisi apa pun.

```
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

Output :

```
[]
```

Loop Lists

dapat mengulang melalui item daftar dengan menggunakan for perulangan:

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x)
```

Output :

```
apple
banana
cherry
```

dapat melakukan perulangan melalui item daftar dengan merujuk pada nomor indeksnya.

Gunakan fungsi ` range() dan ` len()untuk membuat objek iterable yang sesuai.

```
thislist = ["apple", "banana", "cherry"]
for i in range(len(thislist)):
  print(thislist[i])
```

Output :

```
apple
banana
cherry
```

dapat mengulang melalui item daftar dengan menggunakan while perulangan.

Gunakan len() fungsi tersebut untuk menentukan panjang daftar, lalu mulai dari 0 dan lakukan perulangan melalui item-item dalam daftar dengan merujuk pada indeksnya.

Ingat untuk meningkatkan indeks sebanyak 1 setelah setiap iterasi.

```
thislist = ["apple", "banana", "cherry"]
i = 0
while i < len(thislist):
  print(thislist[i])
  i = i + 1
```

Output :

```
apple
banana
cherry
```

List Comprehension menawarkan sintaks terpendek untuk melakukan perulangan melalui daftar:

```
thislist = ["apple", "banana", "cherry"]
[print(x) for x in thislist]
```

Output :

```
apple
banana
cherry
```

List Comprehension

Tanpa list comprehension, Anda harus menulis forpernyataan dengan uji kondisional di dalamnya:

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = []

for x in fruits:
  if "a" in x:
    newlist.append(x)

print(newlist)
```

Output :

```
['apple', 'banana', 'mango']
```

Dengan list comprehension, Anda dapat melakukan semua itu hanya dengan satu baris kode:

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = [x for x in fruits if "a" in x]

print(newlist)
```

Ouput:

```
['apple', 'banana', 'mango']
```

The Syntax

seperti filter yang hanya menerima item yang dievaluasi menjadi True.

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x for x in fruits if x != "apple"]

print(newlist)
```

```
['banana', 'cherry', 'kiwi', 'mango']
```

Kondisi if x != "apple" akan mengembalikan nilai Trueuntuk semua elemen selain "apel", sehingga daftar baru akan berisi semua buah kecuali "apel".

Syarat ini bersifat opsional dan dapat dihilangkan:

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x for x in fruits]

print(newlist)
```

Output :

```
['apple', 'banana', 'cherry', 'kiwi', 'mango']
```

Objek *yang dapat diiterasi* dapat berupa objek apa pun yang dapat diiterasi, seperti daftar, tuple, set, dan lain sebagainya.

```
newlist = [x for x in range(10)]

print(newlist)
```

Output :

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Expression

Expression tersebut merupakan item saat ini dalam iterasi, tetapi juga merupakan hasilnya, yang dapat Anda manipulasi sebelum akhirnya menjadi item daftar dalam daftar baru:

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x.upper() for x in fruits]

print(newlist)
```

Output :

```
['APPLE', 'BANANA', 'CHERRY', 'KIWI', 'MANGO']
```

Sort Lists

List memiliki sort()metode yang akan mengurutkan daftar secara alfanumerik, menaik, secara default:

```
thislist = ["orange", "mango", "kiwi", "pineapple",
"banana"]

thislist.sort()

print(thislist)
```

Output :

```
['banana', 'kiwi', 'mango', 'orange', 'pineapple']
```

Untuk mengurutkan secara menurun, gunakan argumen kata kunci reverse = True:

```
thislist = ["orange", "mango", "kiwi", "pineapple",
"banana"]

thislist.sort(reverse = True)

print(thislist)
```

Output :

```
['pineapple', 'orange', 'mango', 'kiwi', 'banana']
```

Anda juga dapat menyesuaikan fungsi Anda sendiri dengan menggunakan argumen kata kunci .key = *function*
Fungsi ini akan mengembalikan angka yang akan digunakan untuk mengurutkan daftar (angka terendah terlebih dahulu

```
def myfunc(n):
  return abs(n - 50)

thislist = [100, 50, 65, 82, 23]

thislist.sort(key = myfunc)

print(thislist)
```

Output :

```
[50, 65, 23, 82, 100]
```

Secara default, sort()metode ini peka terhadap huruf besar dan kecil, sehingga semua huruf kapital diurutkan sebelum huruf kecil:

```
thislist = ["banana", "Orange", "Kiwi", "cherry"]

thislist.sort()

print(thislist)
```

Output:

```
['Kiwi', 'Orange', 'banana', 'cherry']
```

Bagaimana jika Anda ingin membalik urutan sebuah daftar, terlepas dari urutan abjadnya?
Metode ini reverse()membalik urutan pengurutan elemen saat ini.

```
thislist = ["banana", "Orange", "Kiwi", "cherry"]

thislist.reverse()

print(thislist)
```

Output:

```
['cherry', 'Kiwi', 'Orange', 'banana']
```

Copy Lists

Anda tidak dapat menyalin daftar hanya dengan mengetik list2 = list1 , karena list2 hanya akan menjadi referensi ke list1, dan perubahan yang dilakukan di list1akan secara otomatis juga dilakukan di list2.

Anda dapat menggunakan metode List bawaan copy()untuk menyalin sebuah daftar.

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy()
print(mylist)
```

Output :

```
['apple', 'banana', 'cherry']
```

Cara lain untuk membuat salinan adalah dengan menggunakan metode bawaan list().

```
thislist = ["apple", "banana", "cherry"]
mylist = list(thislist)
print(mylist)
```

Output:

```
['apple', 'banana', 'cherry']
```

Anda juga dapat membuat salinan daftar dengan menggunakan :operator (slice).

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist[:]
print(mylist)
```

Output:

```
['apple', 'banana', 'cherry']
```

Join Lists

Ada beberapa cara untuk menggabungkan, atau menyatukan, dua atau lebih daftar di Python.

Salah satu cara termudah adalah dengan menggunakan + operator.

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]

list3 = list1 + list2
print(list3)
```

Output:

```
['a', 'b', 'c', 1, 2, 3]
```

Anda dapat menggunakan extend() metode yang tujuannya adalah untuk menambahkan elemen dari satu daftar ke daftar lain:

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

list1.extend(list2)
print(list1)
```

Output:

```
['a', 'b', 'c', 1, 2, 3]
```

PYTHON TUPLES

Tuples digunakan untuk menyimpan beberapa item dalam satu variabel.

Tuples dalah kumpulan yang terurut dan **tidak dapat diubah** .

Tuples ditulis dengan tanda kurung.

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

Output :

```
('apple', 'banana', 'cherry')
```

Item dalam Tuples diurutkan, tidak dapat diubah, dan memungkinkan nilai duplikat.

Item-item dalam Tuples diindeks, item pertama memiliki indeks [0], item kedua memiliki indeks [1], dan seterusnya.

Tuples bersifat tidak dapat diubah, artinya kita tidak dapat mengubah, menambah, atau menghapus item setelah tuple dibuat.

Karena tuple diindeks, maka tuple dapat memiliki item dengan nilai yang sama:

```
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
print(thistuple)
```

Output :

```
('apple', 'banana', 'cherry', 'apple', 'cherry')
```

Untuk menentukan berapa banyak item yang dimiliki sebuah tuple, gunakan len()fungsi berikut:

```
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
```

Output :

```
3
```

Untuk membuat tuple yang hanya berisi satu item, Anda harus menambahkan koma setelah item tersebut, jika tidak, Python tidak akan mengenalinya sebagai tuple.

```
thistuple = ("apple",)
print(type(thistuple))

#NOT a tuple
thistuple = ("apple")
print(type(thistuple))
```

Output:

```
<class 'tuple'>
<class 'str'>
```

Item dalam tuple dapat berupa tipe data apa pun:

```
tuple1 = ("apple", "banana", "cherry")
tuple2 = (1, 5, 7, 9, 3)
tuple3 = (True, False, False)

print(tuple1)
print(tuple2)
print(tuple3)
```

Output :

```
('apple', 'banana', 'cherry')
(1, 5, 7, 9, 3)
(True, False, False)
```

tuple dapat berisi berbagai tipe data:

```
tuple1 = ("abc", 34, True, 40, "male")

print(tuple1)
```

Output :

```
('abc', 34, True, 40, 'male')
```

Dari sudut pandang Python, tuple didefinisikan sebagai objek dengan tipe data 'tuple':
<class 'tuple'>

Kita juga bisa menggunakan konstruktor tuple() untuk membuat tuple

```
thistuple = tuple(("apple", "banana", "cherry"))
print(thistuple)
```

Output :

```
('apple', 'banana', 'cherry')
```

Access Tuple Items

Anda dapat mengakses item tuple dengan merujuk pada nomor indeks, yang berada di dalam tanda kurung siku:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[1])
```

Output:

```
banana
```

Pengindeksan negatif berarti dimulai dari akhir.
-1 mengacu pada item terakhir, -2 mengacu pada item kedua terakhir, dan seterusnya.

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[-1])
```

Output :

```
cherry
```

dapat menentukan rentang indeks dengan menentukan di mana rentang tersebut dimulai dan di mana rentang tersebut berakhir.
Saat menentukan rentang, nilai yang dikembalikan akan berupa tuple baru dengan item yang ditentuk

```
thistuple = ("apple", "banana", "cherry", "orange",
"kiwi", "melon", "mango")
print(thistuple[2:5])
```

Output :

```
('cherry', 'orange', 'kiwi')
```

Begitu juga dengan indeks negatif

Update Tuples

Setelah sebuah tuple dibuat, Anda tidak dapat mengubah nilainya. Tuple bersifat tidak dapat diubah , atau disebut juga immutable .
Namun ada cara lain. Anda dapat mengubah tuple menjadi list, mengubah list tersebut, dan mengubah list tersebut kembali menjadi tuple.

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)

print(x)
```

Output:

```
("apple", "kiwi", "cherry")
```

Karena tuple bersifat immutable (tidak dapat diubah), tuple tidak memiliki metode bawaan append(), tetapi ada cara lain untuk menambahkan item ke tuple.

1. **Konversi menjadi daftar** : Sama seperti cara *mengubah* tuple, Anda dapat mengonversinya menjadi daftar, menambahkan item Anda, dan mengonversinya kembali menjadi tuple.

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.append("orange")
thistuple = tuple(y)

print(thistuple)
```

Output:

```
('apple', 'banana', 'cherry', 'orange')
```

2. **Menambahkan tuple ke tuple** . Anda diperbolehkan menambahkan tuple ke tuple, jadi jika Anda ingin menambahkan satu item (atau banyak), buat tuple baru dengan item tersebut, dan tambahkan ke tuple yang sudah ada:

```
thistuple = ("apple", "banana", "cherry")
y = ("orange",)
thistuple += y

print(thistuple)
```

Output :

```
('apple', 'banana', 'cherry', 'orange')
```

Tuple **bersifat tetap** , jadi Anda tidak dapat menghapus item darinya, tetapi Anda dapat menggunakan solusi yang sama seperti yang kita gunakan untuk mengubah dan menambahkan item tuple:

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.remove("apple")
thistuple = tuple(y)

print(thistuple)
```

Output:

```
('banana', 'cherry')
```

Unpack Tuples

Saat kita membuat tuple, kita biasanya menetapkan nilai padanya. Ini disebut "packing" tuple:

```
fruits = ("apple", "banana", "cherry")

print(fruits)
```

Output:

```
('apple', 'banana', 'cherry')
```

di Python, kita juga diperbolehkan untuk mengekstrak nilai kembali ke dalam variabel. Ini disebut "unpacking"

```
fruits = ("apple", "banana", "cherry")

(green, yellow, red) = fruits

print(green)
print(yellow)
print(red)
```

Output :

```
apple
banana
cherry
```

Jika jumlah variabel kurang dari jumlah nilai, Anda dapat menambahkan awalan *pada nama variabel dan nilai-nilai tersebut akan diberikan ke variabel sebagai daftar:

```
fruits = ("apple", "banana", "cherry", "strawberry",
"raspberry")

(green, yellow, *red) = fruits

print(green)
print(yellow)
print(red)
```

Output :

```
apple
banana
['cherry', 'strawberry', 'raspberry']
```

Jika tanda bintang ditambahkan ke nama variabel selain yang terakhir, Python akan menetapkan nilai ke variabel tersebut hingga jumlah nilai yang tersisa sama dengan jumlah variabel yang tersisa.

```
fruits = ("apple", "mango", "papaya", "pineapple",
"cherry")

(green, *tropic, red) = fruits

print(green)
print(tropic)
print(red)
```

Output :

```
apple
['mango', 'papaya', 'pineapple']
cherry
```

Loop Tuples

Anda dapat mengulang melalui item tuple dengan menggunakan forperulangan.

```
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
  print(x)
```

Output :

```
apple
banana
cherry
```

dapat melakukan perulangan melalui item tuple dengan merujuk pada nomor indeksnya.

Gunakan fungsi ` range() dan ` len() untuk membuat objek iterable yang sesuai.

```
thistuple = ("apple", "banana", "cherry")
for i in range(len(thistuple)):
  print(thistuple[i])
```

Output:

```
apple
banana
cherry
```

Dapat mengulang melalui item tuple dengan menggunakan while perulangan.
Gunakan len() fungsi tersebut untuk menentukan panjang tuple, lalu mulai dari 0 dan lakukan perulangan melalui item-item tuple dengan merujuk pada indeksnya.
Ingat untuk meningkatkan indeks sebanyak 1 setelah setiap iterasi.

```
thistuple = ("apple", "banana", "cherry")
i = 0
while i < len(thistuple):
  print(thistuple[i])
  i = i + 1
```

Output:

```
apple
banana
cherry
```

Join Tuples

Untuk menggabungkan dua atau lebih tupel, Anda dapat menggunakan + operator:

```
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
print(tuple3)
```

Output:

```
('a', 'b', 'c', 1, 2, 3)
```

Mengalikan tuple

Jika Anda ingin mengalikan isi sebuah tuple sebanyak jumlah tertentu, Anda dapat menggunakan * operator:

```
fruits = ("apple", "banana", "cherry")
mytuple = fruits * 2

print(mytuple)
```

Output :

```
('apple', 'banana', 'cherry', 'apple', 'banana', 'cherry')
```

Python Sets

set digunakan untuk menyimpan beberapa item dalam satu variabel.

ditulis dengan tanda kurung kurawal.

```
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

Output :

```
{'banana', 'cherry', 'apple'}
```

Item yang ditetapkan tidak berurutan, tidak dapat diubah, dan tidak mengizinkan nilai duplikat.

Item dalam set tidak dapat diubah, artinya kita tidak dapat mengubah item setelah set tersebut dibuat.

Suatu himpunan tidak boleh memiliki dua item dengan nilai yang sama.

```
thisset = {"apple", "banana", "cherry", "apple"}

print(thisset)
```

Output :

```
{'banana', 'cherry', 'apple'}
```

True dan 1dianggap memiliki nilai yang sama

```
thisset = {"apple", "banana", "cherry", True, 1, 2}

print(thisset)
```

Ouput :

```
{True, 2, 'banana', 'cherry', 'apple'}
```

False dan 0dianggap memiliki nilai yang sama

```
thisset = {"apple", "banana", "cherry", False, True, 0}

print(thisset)
```

Ouptu :

```
{False, True, 'cherry', 'apple', 'banana'}
```

Untuk menentukan berapa banyak item yang dimiliki suatu set, gunakan len() fungsi tersebut.

```
thisset = {"apple", "banana", "cherry"}

print(len(thisset))
```

Output :

```
3
```

Item dalam himpunan dapat berupa tipe data apa pun:

```
set1 = {"apple", "banana", "cherry"}
set2 = {1, 5, 7, 9, 3}
set3 = {True, False, False}

print(set1)
print(set2)
print(set3)
```

Output :

```
{'cherry', 'apple', 'banana'}
{1, 3, 5, 7, 9}
{False, True}
```

Suatu himpunan dapat berisi berbagai tipe data:

```
set1 = {"abc", 34, True, 40, "male"}

print(set1)
```

Output :

```
{True, 34, 40, 'male', 'abc'}
```

Kita juga bisa menggunakan konstruktor set() untuk membuat himpunan.

```
thisset = set(("apple", "banana", "cherry"))
print(thisset)
```

Output :

```
{'apple', 'cherry', 'banana'}
```

Access Set Items

dapat melakukan perulangan melalui item-item dalam himpunan menggunakan sebuah for loop, atau menanyakan apakah nilai tertentu ada dalam himpunan, dengan menggunakan inkata kunci.

```
thisset = {"apple", "banana", "cherry"}

for x in thisset:
  print(x)
```

Output:

```
apple
cherry
banana
```

Periksa apakah "banana" ada dalam himpunan tersebut:

```
thisset = {"apple", "banana", "cherry"}

print("banana" in thisset)
```

Output:

```
True
```

Periksa apakah "banana" TIDAK ada dalam himpunan:

```
thisset = {"apple", "banana", "cherry"}

print("banana" not in thisset)
```

Output:

```
False
```

Add Set Items

Untuk menambahkan satu item ke dalam suatu set, gunakan add() metode ini.

```
thisset = {"apple", "banana", "cherry"}

thisset.add("orange")

print(thisset)
```

Output:

```
{'cherry', 'banana', 'apple', 'orange'}
```

Untuk menambahkan item dari set lain ke set saat ini, gunakan update() metode tersebut.

```
thisset = {"apple", "banana", "cherry"}
tropical = {"pineapple", "mango", "papaya"}

thisset.update(tropical)

print(thisset)
```

output:

```
{'apple', 'cherry', 'banana', 'pineapple', 'papaya', 'mango'}
```

Objek dalam update()metode tersebut tidak harus berupa himpunan, melainkan dapat berupa objek iterable apa pun (tuple, list, dictionary, dll.).

```
thisset = {"apple", "banana", "cherry"}
mylist = ["kiwi", "orange"]

thisset.update(mylist)

print(thisset)
```

Output :

```
{'banana', 'cherry', 'apple', 'orange', 'kiwi'}
```

Remove Set Items

Untuk menghapus item dalam suatu set, gunakan metode remove(), atau discard().

```
thisset = {"apple", "banana", "cherry"}

thisset.remove("banana")

print(thisset)
```

Output:

```
{'cherry', 'apple'}
```

Hapus "banana" dengan menggunakan discard() metode:

```
thisset = {"apple", "banana", "cherry"}

thisset.discard("banana")

print(thisset)
```

Output:

```
{'cherry', 'apple'}
```

dapat menggunakan pop()metode ini untuk menghapus suatu item, tetapi metode ini akan menghapus item secara acak, sehingga Anda tidak dapat memastikan item mana yang akan dihapus.

Nilai kembalian dari pop()metode ini adalah item yang dihapus.

```
thisset = {"apple", "banana", "cherry"}

x = thisset.pop()

print(x)

print(thisset)
```

Output:

```
banana
{'cherry', 'apple'}
```

metode ini clear() mengosongkan himpunan:

```
thisset = {"apple", "banana", "cherry"}

thisset.clear()

print(thisset)
```

Output:

```
set()
```

Kata kunci tersebut delakan menghapus seluruh himpunan:

```
thisset = {"apple", "banana", "cherry"}

del thisset

print(thisset)
```

Output:

```
Traceback (most recent call last):
  File "demo_set_del.py", line 5, in <module>
    print(thisset) #this will raise an error because
NameError: name 'thisset' is not defined
```

Loop Sets

Anda dapat melakukan perulangan melalui item-item dalam himpunan dengan menggunakan for perulangan:

```
thisset = {"apple", "banana", "cherry"}

for x in thisset:
  print(x)
```

Outpupt:

```
banana
cherry
apple
```

Join Sets

Ada beberapa cara untuk menggabungkan dua himpunan atau lebih di Python.

Metode " union() and" update() menggabungkan semua item dari kedua himpunan.

Metode ini intersection() HANYA menyimpan data duplikat.

Metode ini difference() mempertahankan item dari himpunan pertama yang tidak ada di himpunan lainnya.

Metode ini symmetric_difference() menyimpan semua item KECUALI item duplikat.

Metode union() mengembalikan himpunan baru yang berisi semua item dari kedua himpunan tersebut.

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}

set3 = set1.union(set2)
print(set3)
```

Output:

```
{1, 3, 'b', 'c', 'a', 2}
```

Anda dapat menggunakan | operator sebagai pengganti union() metode, dan Anda akan mendapatkan hasil yang sama

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}

set3 = set1 | set2
print(set3)
```

Output:

```
{3, 2, 'c', 'a', 1, 'b'}
```

Gabungkan beberapa himpunan dengan union()metode:

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set3 = {"John", "Elena"}
set4 = {"apple", "bananas", "cherry"}

myset = set1.union(set2, set3, set4)
print(myset)
```

Output:

```
{1, 2, 3, 'Elena', 'b', 'John', 'a', 'bananas', 'cherry', 'c', 'apple'}
```

Metode ini union()memungkinkan Anda untuk menggabungkan sebuah himpunan dengan tipe data lain, seperti daftar atau tupel.
Hasilnya akan berupa himpunan.
Contoh

```
x = {"a", "b", "c"}
y = (1, 2, 3)

z = x.union(y)
print(z)
```

Output:

```
{'a', 2, 1, 'b', 3, 'c'}
```

Metode ini update()memasukkan semua item dari satu set ke dalam set lainnya.
Perubahan tersebut update()mengubah himpunan asli, dan tidak mengembalikan himpunan baru.

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}

set1.update(set2)
print(set1)
```

OUTPUT:

```
{'a', 3, 1, 2, 'b', 'c'}
```

Metode ini intersection()akan mengembalikan himpunan baru yang hanya berisi item-item yang ada di kedua himpunan tersebut.

```
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set3 = set1.intersection(set2)
print(set3)
```

Output:

```
{'apple'}
```

dapat menggunakan &operator sebagai pengganti metode, dan Anda akan mendapatkan hasil yang sama. intersection()

```
set1 = {"apple", "banana" , "cherry"}
set2 = {"google", "microsoft", "apple"}

set3 = set1 & set2
print(set3)
```

Output :

```
{'apple'}
```

Metode ini intersection_update()juga hanya akan menyimpan data duplikat, tetapi akan mengubah himpunan asli alih-alih mengembalikan himpunan baru.

```
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set1.intersection_update(set2)
print(set1)
```

Output:

```
{'apple'}
```

Nilai Truedan 1 dianggap sebagai nilai yang sama. Hal yang sama berlaku untuk Falsedan 0

```
set1 = {"apple", 1, "banana", 0, "cherry"}
set2 = {False, "google", "microsoft", "apple", True}

set3 = set1.intersection(set2)
print(set3)
```

Outpput:

```
{False, True, 'apple'}
```

Metode ini akan mengembalikan himpunan baru yang hanya berisi item dari himpunan pertama yang tidak ada di himpunan lainnya.difference()

```
set1 = {"apple", "banana" , "cherry"}
set2 = {"google", "microsoft", "apple"}

set3 = set1.difference(set2)
print(set3)
```

Output:

```
{'banana', 'cherry'}
```

Anda dapat menggunakan - operator sebagai pengganti metode, dan Anda akan mendapatkan hasil yang sama. difference()

```
set1 = {"apple", "banana" , "cherry"}
set2 = {"google", "microsoft", "apple"}

set3 = set1 - set2
print(set3)
```

Output:

```
{'banana', 'cherry'}
```

Metode ini difference_update()akan mempertahankan item dari himpunan pertama yang tidak ada di himpunan lain, tetapi akan mengubah himpunan asli alih-alih mengembalikan himpunan baru.

```
set1 = {"apple", "banana" , "cherry"}
set2 = {"google", "microsoft", "apple"}

set1.difference_update(set2)
print(set1)
```

Output:

```
{'banana', 'cherry'}
```

Python frozenset

frozensetadalah versi himpunan yang tidak dapat diubah.
Seperti himpunan, ia berisi elemen-elemen unik, tidak berurutan, dan tidak dapat diubah.
Tidak seperti himpunan, elemen tidak dapat ditambahkan atau dihapus dari frozenset.

Gunakan frozenset()konstruktor untuk membuat frozenset dari iterable apa pun.

```
x = frozenset({"apple", "banana", "cherry"})

#display x:
print(x)

#display the data type of x:
print(type(x))
```

Output:

```
frozenset({'banana', 'cherry', 'apple'})
<class 'frozenset'>
```

Python Dictionaries

Dictionaries digunakan untuk menyimpan nilai data dalam pasangan kunci:nilai.
Dictionaries adalah kumpulan yang terurut*, dapat diubah, dan tidak mengizinkan duplikat.
Dictionaries ditulis dengan tanda kurung kurawal, dan memiliki kunci dan nilai:

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

Output:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Item-item dalam Dictionaries diurutkan, dapat diubah, dan tidak mengizinkan duplikasi.
Item-item dalam Dictionaries disajikan dalam pasangan kunci:nilai, dan dapat dirujuk dengan menggunakan nama kuncinya.

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict["brand"])
```

Output:

```
Ford
```

Nilai duplikat akan menimpa nilai yang sudah ada:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964,
  "year": 2020
}
print(thisdict)
```

Output:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

Untuk menentukan berapa banyak item yang dimiliki sebuah Dictionaries, gunakan fungsi berikut: len()

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964,
  "year": 2020
}
print(len(thisdict))
```

Ouput:

```
3
```

Nilai dalam item Dictionaries dapat berupa tipe data apa pun:

```
thisdict = {
  "brand": "Ford",
  "electric": False,
  "year": 1964,
  "colors": ["red", "white", "blue"]
}

print(thisdict)
```

Oootput:

```
{'brand': 'Ford', 'electric': False, 'year': 1964, 'colors': ['red', 'white', 'blue']}
```

Dari sudut pandang Python, Dictionaries didefinisikan sebagai objek dengan tipe data 'dict':

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(type(thisdict))
```

Output:

```
<class 'dict'>
```

Kita juga bisa menggunakan konstruktor dict() untuk membuat Dictionaries.

```
thisdict = dict(name = "John", age = 36, country =
"Norway")

print(thisdict)
```

Output:

```
{'name': 'John', 'age': 36, 'country': 'Norway'}
```

Access Dictionary Items

dapat mengakses item-item dalam Dictionaries dengan merujuk pada nama kuncinya, yang berada di dalam tanda kurung siku:

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = thisdict["model"]
print(x)
```

Output:

```
Mustang
```

Ada juga metode lain get()yang akan memberikan hasil yang sama:

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = thisdict.get("model")
print(x)
```

Output:

```
Mustang
```

Metode ini values()akan mengembalikan daftar semua nilai dalam Dictionaries tersebut.

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

x = thisdict.values()

print(x)
```

Output:

```
dict_values(['Ford', 'Mustang', 1964])
```

Metode ini items()akan mengembalikan setiap item dalam Dictionaries, sebagai tuple dalam sebuah daftar

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

x = thisdict.items()

print(x)
```

Output:

```
dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])
```

Change Dictionary Items

Anda dapat mengubah nilai item tertentu dengan merujuk pada nama kuncinya:

Ubah "tahun" menjadi 2018:

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

thisdict["year"] = 2018

print(thisdict)
```

Output:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2018}
```

Perbarui "year" mobil dengan menggunakan update() metode berikut:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.update({"year": 2020})

print(thisdict)
```

Output:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

Add Dictionary Items

Menambahkan item ke dalam kamus dilakukan dengan menggunakan kunci indeks baru dan menetapkan nilai padanya:

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["color"] = "red"
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

Remove Dictionary Items

Ada beberapa metode untuk menghapus item dari kamus:

Metode ini pop()menghapus item dengan nama kunci yang ditentukan:

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.pop("model")
print(thisdict)
```

Output:

```
{'brand': 'Ford', 'year': 1964}
```

Metode ini popitem()menghapus item terakhir yang dimasukkan

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.popitem()
print(thisdict)
```

Output:

```
{'brand': 'Ford', 'model': 'Mustang'}
```

Kata kunci tersebut delmenghapus item dengan nama kunci yang ditentukan:

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict["model"]
print(thisdict)
```

Outpt:

```
{'brand': 'Ford', 'year': 1964}
```

Loop Dictionaries

Anda dapat melakukan perulangan melalui kamus dengan menggunakan forperulangan.

Saat melakukan perulangan melalui kamus, nilai yang dikembalikan adalah *kunci-kunci* kamus tersebut, tetapi ada juga metode untuk mengembalikan *nilai-nilainya* .

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
for x in thisdict:
  print(x)
```

Output:

```
brand
model
year
```

Cetak semua *nilai* dalam kamus, satu per satu

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
for x in thisdict:
  print(thisdict[x])
```

```
Ford
Mustang
1964
```

juga dapat menggunakan values()metode ini untuk mengembalikan nilai-nilai dari sebuah kamus:

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
for x in thisdict.values():
  print(x)
```

Output:

```
Ford
Mustang
1964
```

Copy Dictionaries

Anda tidak dapat menyalin kamus hanya dengan mengetik dict2 = dict1, karena: dict2hanya akan menjadi *referensi* ke dict1, dan perubahan yang dilakukan di dict1akan secara otomatis juga dilakukan di dict2.

Ada beberapa cara untuk membuat salinan, salah satunya adalah dengan menggunakan metode Kamus (Dictionary) bawaan copy().

Buat salinan kamus dengan copy()metode:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict = thisdict.copy()
print(mydict)
```

Output:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Cara lain untuk membuat salinan adalah dengan menggunakan fungsi bawaan dict().

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict = dict(thisdict)
print(mydict)
```

Output:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

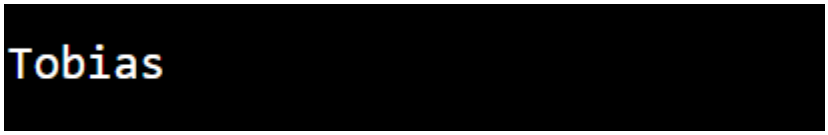
Nested Dictionaries

Untuk mengakses item dari kamus bersarang, Anda menggunakan nama kamus tersebut, dimulai dengan kamus terluar:

```
myfamily = {
  "child1" : {
    "name" : "Emil",
    "year" : 2004
  },
  "child2" : {
    "name" : "Tobias",
    "year" : 2007
  },
  "child3" : {
    "name" : "Linus",
    "year" : 2011
  }
}

print(myfamily["child2"]["name"])
```

Output:



PENUTUPAN

Sekian laporan yang dapat saya bikin,mohon maaf atas ada kesalahan penulisan,semoga laporan ini bermanfaat untuk penulis dan juga pembaca,sekian dari saya terimakasih.