LAPORAN PRAKTIKUM I
PRAKTIKUM STRUKTUR DATA

Materi Pertemuan-2

DISUSUN OLEH:
Yohanes Doly Jaya Laneta Tambunan
25071104585

DOSEN PENGAMPU:
Reny Fitri Yani, S.T., M.T.

ASISTEN PRAKTIKUM:
Rizkillah Ramanda Sinyo
Muhammad Abidillah

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS RIAU
PEKANBARU
2026

DAFTAR ISI

BAB I 4

PELAKSANAAN PRAKTIKUM 4

1.1 List 4

1.1.1 Python list 4

1.1.2 Acces list items 4

1.1.3 Change list items 4

1.1.4 Add list items 5

1.1.5 Remove list items 5

1.1.6 Loop list 5

1.1.7 List Comprehension 6

1.1.8 Sort Lists 6

1.1.9 Copy lists 7

1.1.10 Join lists 7

1.2 Tuple 8

1.2.1 Python tuple 8

1.2.2 Access tuple items 8

1.2.3 Update tuple 9

1.2.4 Unpack tuples 9

1.2.5 Loop tuples 10

1.2.6 Join tuples 10

1.3 Set 10

1.3.1 Set python 11

1.3.2 Acces set items 11

1.3.3 Add set items 11

1.3.4 Remove set items 12

1.3.5 Loop set 12

1.3.6 Join set 12

1.3.7 Python frozenset 14

1.4 Dictionaries 14

1.4.1 Python dictionaries 14

1.4.2 Acces dicitonaries items 15
1.4.3 Change dictionary items 15
1.4.4 Add dictionary items 16
1.4.5 Remove dictionsry items 16
1.4.6 Loop dictionaries 16
1.4.7 Copy dictionaries 17
1.4.8 Nested dictionary 18

BAB I

PELAKSANAAN PRAKTIKUM

1.1 List

List digunakan untuk menyimpan banyak item dalam satu variabel. Item dalam list diurutkan, dapat diubah, dan memungkinkan nilai duplikat.

1.1.1 Python list

1. List dibuat menggunakan tanda kurung siku.

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

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

2. List mengijinkan nilai yang sama.

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

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

1.1.2 Acces list items

1. Item dalam list diindeks dan dapat diakses dengan merujuk pada nomor indeks.

```
thislist = [ "apel" , "pisang" , "ceri" ]
cetak ( daftar ini [ 1 ])
```

```
pisang
```

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

```
cherry
```

2. List 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])
```

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

1.1.3 Change list items

1. Untuk mengubah nilai item tertentu, lihat nomor indeksnya.

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

print(thislist)
```

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

2. Untuk mengubah nilai item dalam rentang tertentu, buat list dengan nilai baru, dan buat ke rentang nomor indeks tempat yang ingin memasukkan nilai baru.

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "mango"]
thislist[1:3] = ["blackcurrant", "watermelon"]
print(thislist)
```

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

1.1.4 Add list items

1. Untuk menyisipkan item list baru. Tanpa mengganti nilai yang sudah ada, dapat menggunakan insert().

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(2, "watermelon")
print(thislist)
```

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

2. Untuk menambahkan item ke akhir list, gunakan append().

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

thislist.append("orange")

print(thislist)
```

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

3. Untuk menambahkan elemen dari daftar lain ke daftar saat ini, gunakan extend().

```
thislist = ["apple", "banana", "cherry"]
tropical = ["mango", "pineapple", "papaya"]
thislist.extend(tropical)
print(thislist)
```

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

1.1.5 Remove list items

1. Metode remove () menghapus item yang ditentukan.

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

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

2. Metode ini pop() menghapus indeks yang ditentukan.

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

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

3. Kata kunci del juga menghapus indeks yang ditentukan, dan juga bisa menghapus seluruh item

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

1.1.6 Loop list

1. List dapat mengulang melalui item list dengan menggunakan for loop.

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

```
apple
banana
cherry
```

2. List dapat melakukan loop melalui item daftar dengan merujuk

pada nomor indeksnya. Gunakan fungsi range() dan len().

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

```
apple
banana
cherry
```

3. Anda dapat mengulang melalui item list dengan menggunakan while loop. Gunakan len() fungsi untuk menentukan panjang list.

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

```
apple
banana
cherry
```

1.1.7 List Comprehension

1. List Comprehension dapat membuat sintaks terpendek untuk melakukan loop melalui list.

```
thislist = [ "apel" , "pisang" , "ceri" ]
[ print ( x ) for x in thislist ]
```

```
apel
pisang
ceri
```

2. Berdasarkan daftar buah-buahan yang ada,kita ingin membuat list baru yang hanya berisi buah-buahan yang namanya mengandung huruf "a". Tanpa list comprehension, harus menulis for pernyataan dengan uji kondisional di dalamnya. List comprehension 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)
```

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

3. Bisa juga membuat list baru dengan huruf besar

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

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

print(newlist)
```

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

1.1.8 Sort Lists

1. Metode sort() dapat digunakan untuk mengurutkan list secara alfanumerik, dan menaik.

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

thislist.sort()

print(thislist)
```

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

2. Contoh penggunaan sort() dalam list yang berisi angka.

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

thislist.sort()

print(thislist)
```

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

3. Gunakan sort(reverse = True) untuk mengurutkan daftar secara menurun.

```
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort(reverse = True)
print(thislist)
```

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

4. Contoh penggunaan sort(reverse = True) pada list yang berisi angka.

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

thislist.sort(reverse = True)

print(thislist)
```

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

5. Sort juga dapat menyesuaikan fungsi dengan menggunakan 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)
```

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

1.1.9 Copy lists

1. Dalam lists tidak dapat menyalin list hanya dengan mengetik list 2 = list 1, karena hanya akan menjadi refrensi ke list1, dan perubahan yang dilakukan di list1akan secara otomatis juga dilakukan di list2. Kita dapat menggunakan copy() untuk menyalin sebuah list.

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

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

2. Bisa juga menggunakan list().

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

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

1.1.10 Join lists

1. Cara menggabungkan list satu dengan list yang lainnya dengan menggunakan + operator.

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

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

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

2. Cara lainnya dengan menggunakan append().

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

for x in list2:
  list1.append(x)

print(list1)
```

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

3. Cara lainnya dengan menggunakan extend().

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

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

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

1.2 Tuple

Tuple digunakan untuk menyimpan beberapa item dalam satu variabel. Tuple adalah kumpulan yang terurut dan tidak dapat diubah .Tuple ditulis dengan tanda kurung.

1.2.1 Python tuple

1. Python ditulis dengan tanda kurung.

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

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

2. Tuple dapat memiliki nilai duplikat.

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

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

1.2.2 Access tuple items

1. Sama dengan list, kita dapat mengakses item tuple dengan merujuk pada nomor indeks.

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

```
banana
```

2. Tuple bisa menentukan rentang indeks.

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

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

1.2.3 Update tuple

1. Tuple bersifat tidak dapat diubah.Namun ada cara lain. Dengan cara mengubah tuple menjadi list, dan mengubah list tersebut kembali menjadi tuple.

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

print(x)
```

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

2. Kita dapat menambahkan item tuple dengan mengonversinya menjadi list dan mengonversinya kembali menjadi tuple.

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

print(thistuple)
```

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

3. Kita diperbolehkan menambahkan tuple ke tuple.

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

print(thistuple)
```

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

4. Kita dapat menghapus tuple dengan mengonversinya menjadi list.

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

print(thistuple)
```

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

1.2.4 Unpack tuples

1. Di python kita dapat mengekstrak nilai kembali ke dalam variabel. Ini disebut "unpacking".

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

(green, yellow, red) = fruits

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

```
apple
banana
cherry
```

2. Jika jumlah variabel kurang dari jumlah nilai, dapat menggunakan 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)
```

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

Variabel *red akan mempirnt nama dari kata ke 3 sampai kata terakhir.

1.2.5 Loop tuples

1. Kita dapat mengulang melalui item tuple dengan menggunakan for loop.

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

```
apple
banana
cherry
```

2. Kita juga dapat melakukan perulangan melalui item tuple dengan memakai nomor indeks menggunakan range() dan len().

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

```
apple
banana
cherry
```

3. Kita dapat mengulang melalui item tuple dengan menggunakan while loop dengan menggunakan len().

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

```
apple
banana
cherry
```

1.2.6 Join tuples

1. Dengan menggunakan operator +

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

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

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

2. **Jika ingin mengalikan isi sebuah tuple sebanyak jumlah tertentu menggunakan * operator.**

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

print(mytuple)
```

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

1.3 Set

Set digunakan untuk menyimpan beberapa item dalam satu variabel. Set adalah salah satu dari 4 tipe data bawaan di Python yang digunakan untuk menyimpan kumpulan data. Set adalah koleksi yang tidak terurut, tidak dapat diubah, dan tidak terindeks.

1.3.1 Set python

1. Set ditulis dengan tanda kurung kurawal.

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

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

2. Nilai True dan 1dianggap sebagai nilai yang sama dalam set, dan diperlakukan sebagai duplikat. Begitu juga dengan nilai False dan 0.

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

print(thisset)
```

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

1.3.2 Acces set items

1. Kita tidak dapat mengakses item dalam suatu himpunan dengan merujuk pada indeks atau kunci. Namun dengan menggunakan sebuah for loop, atau menanyakan apakah nilai tertentu ada dalam himpunan, dengan menggunakan in kata kunci.

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

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

```
banana
cherry
apple
```

2. Kita bisa membuktikan adanya suatu nama pada set.

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

print("banana" in thisset)
```

```
True
```

1.3.3 Add set items

1. Gunakan add() untuk menambahkan satu item ke dalam suatu set.

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

thisset.add("orange")

print(thisset)
```

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

2. Gunakan update() untuk menambahkan item dari set lain ke set saat ini.

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

thisset.update(tropical)

print(thisset)
```

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

3. Penggunaan update() tidak harus di set, bisa juga di tuple, list, dan dictionary.

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

thisset.update(mylist)

print(thisset)
```

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

1.3.4 Remove set items

1. Gunakan remove() atau discard() untuk menghapus item dalam suatu set.

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

thisset.remove("banana")

print(thisset)
```

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

2. Gunakan pop() untuk menghapus item secara acak.

```
thisset = {"apple", "banana", "cherry"}
x = thisset.pop()
print(x) #removed item
print(thisset) #the set after removal
```

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

3. Gunakan clear() untuk menghapus semua item pada suatu set.

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

thisset.clear()

print(thisset)
```

```
set()
```

1.3.5 Loop set

1. Gunakan for loop untuk melakukan perulangan melalui item item dalam set.

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

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

```
apple
banana
cherry
```

1.3.6 Join set

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

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

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

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

2. Dapat juga menggunakan operator | untuk menggantikan union()

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

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

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

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

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

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

```
{'apple'}
```

4. Dapat juga menggunakan operator & untuk menggantikan intersection().

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

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

```
{'apple'}
```

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

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

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

```
{'apple'}
```

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

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

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

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

7. Dapat juga menggunakan operator – untuk menggantikan difference().

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

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

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

8. Metode difference_update() untuk menimpan item dari himpunan pertama yang tidak ada di himpunan lainnya.

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

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

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

9. Metode symmetric_difference() akan menyimpan elemen-elemen yang tidak terdapat di kedua himpunan tersebut.

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

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

```
{'google', 'banana', 'microsoft', 'cherry'}
```

10. Dapat juga menggunakan operator ^ untuk menggantikan symmetric_difference().

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

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

```
{'google', 'banana', 'cherry', 'microsoft'}
```

1.3.7 Python frozenset

Frozenset adalah versi set yang tidak dapat diubah. Seperti himnan, ia berisi elemen-elemen unik, tidak berurutan, dan tidak dapat diubah . Tidak seperti set, elemen tidak dapat ditambahkan atau dihapus dari frozenset.

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

#display x:
print(x)

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

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

1.4 Dictionaries

Dictionaries digunakan untuk menyimpan nilai data dalam pasangan kunci:nilai. Dictionaries adalah kumpulan yang terurut, dapat diubah, dan tidak mengizinkan duplikat.

1.4.1 Python dictionaries

1. Dictionaries ditulis dengan tanda kurung kurawal, dan memiliki kunci dan nilai.

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

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

2. 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"])
```

```
Ford
```

1.4.2 Acces dicitonaries items

1. Dictionaries 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)
```

```
Mustang
```

2. Bisa juga menggunakan get().

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

```
Mustang
```

3. Gunakan keys() untuk mengembalikan daftar semua kunci dalam dicitonaries tersebut.

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

x = thisdict.keys()

print(x)
```

```
dict_keys(['brand', 'model', 'year'])
```

4. Gunkan values() untuk mengembalkan daftar semua nilai dalam dictionaries

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

x = thisdict.values()

print(x)
```

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

5. Gunakan items() untuk mengembalikan setiap item dalam dictionaries, sebagai tuple dalam sebuah list.

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

x = thisdict.items()

print(x)
```

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

1.4.3 Change dictionary items

1. Cara mengubah nilai item tertentu yaitu dengan merujuk pada nama kuncinya.

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

thisdict["year"] = 2018

print(thisdict)
```

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

2. Gunakan update() untuk memperbarui kamus dengan item item yang diberikan yang harus berupa kunci-nilai.

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

print(thisdict)
```

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

1.4.4 Add dictionary items

1. Menambahkan item ke dalam dictionary 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'}
```

1.4.5 Remove dictionsry items

1. Gunakan pop() untuk menghapus item-item tertentu.

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

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

2. Gunakan popitem() untuk mengahpus item terakhir.

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

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

3. Gunakan del atau clear() untuk mengosongkan semua dictionaries.

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

```
{}
```

1.4.6 Loop dictionaries

1. Gunakan fot loop untuk melakukan perulangan melalui dictionaries,

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

```
brand
model
year
```

2. Gunakan vakues() untuk mengembalikan nilai nilai dari sebuah dictionaries.

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

```
Ford
Mustang
1964
```

3. Gunakan keys() untuk mengembalikan kunci kunci dari sebuah dictionaries.

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

```
brand
model
year
```

4. Gunakan items() untuk melakukan perulangan melalui kunci dan nilai.

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

```
brand Ford
model Mustang
year 1964
```

1.4.7 Copy dictionaries

1. Dictionaries tidak dapat menyalin kamus hanya dengan mengetik dict2 = dict1, karena dict2 hanya akan menjadi *referensi* ke dict1, dan perubahan yang dilakukan di dict1 akan secara otomatis juga dilakukan di dict2. Gunakan copy().

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

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

2. Dapat juga menggunakan fungsi bawaan dict()

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

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

1.4.8 Nested dictionary

1. Sebuah kamus dapat berisi kamus-kamus lain, ini disebut kamus bersarang (nested dictionary).

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

print(myfamily)
```