TUGAS PRAKTIKUM 2

LAPORAN PRAKTIKUM 2

***Dosen Pengampu** : Reny Fitri Yani, S.T., M.T.*

***Asisten** : Rizkillah Ramanda Sinyo & Muhammad Abidillah*

OLEH:

AMANDA PATRICIA

25071206390

TEKNIK INFORMATIKA – C

TP.2025/2026

FAKULTAS TEKNIK

UNIVERSITAS RIAU

A. PYTHON LIST

1. Python List

- **List**

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

Item dalam **list** tersusun secara berurutan, bisa diubah isinya, dan boleh memiliki nilai yang sama.

Setiap item dalam **list** memiliki indeks, di mana item pertama berada pada indeks [0], item kedua pada indeks [1], dan seterusnya.

- **Allow Duplicates**

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

Isi di dalam list boleh lebih dari satu kali, apabila muncul lebih dari satu kali tidak akan menyebabkan error.

- **List Length**

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

List Length berarti berapa banyak atau jumlah data di dalam sebuah list.

- **List Items – Data Types**

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

Dalam penggunaan List Items – Data Types bisa berupa berbagai jenis data, tidak harus satu jenis. Di dalam satu list bisa ada string, int, dan boolean sekaligus.

- **Type()**

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

Penggunaan dari type adalah untuk meng-cek jenis data dari suatu variabel.

- **The List() Constructor**

```
thislist = list(("apple", "banana", "cherry")) # note the double round-
brackets
print(thislist)
```

The list() constructor merupakan cara lain untuk membuat list, bukan hanya memmakai [].

2. Access List Items

- **Access Items**

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

Access items digunakan untuk mengambil data tertentu dari list, agar kita dapat memakai data yang kita butuhkan saja, bukan keseluruhan dari isi list.

- **Negative Indexing**

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

Negative indexing merupakan cara pengambilan data dari belakang list atau item paling terakhir.

- **Range of Indexes**

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

Range of index digunakan ketika ingin mengambil Sebagian isi list atau hanya beberapa data tertentu (tidak semua isi list).

- **Range of Negetive Indexes**

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

Range of indexes berarti mengambil beberapa data dari belakang list sekaligus.

- **Check if Item Exists**

```
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
  print("Yes, 'apple' is in the fruits list")
```

Check if items exists berguna untuk meng-cek apakah terdapat suatu data yang dicari ada atau tidak.

3. Change List Items

- **Change Item Value**

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

Change item value digunakan untuk mengganti isi data di dalam lis, apabila kita ingin mengubah data tertentu tanpa membuat list baru.

- **Change a Range if Item Values**

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

Change a Range of Item Values berarti mrngganti beberapa data sekaligus dalam list. Digunakan jika ingin mengubah satu item hanya dengan satu cara.

- **Insert Items**

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

Insert items berarti menambhakan data ke dalam list pada posisi tertentu, tanpa menhapus data yang sudah ada.

4. Add List Items

- **Append Items**

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

Append items artinya menambahkan satu item ke bagian paling akhir list.

- **Insert Items**

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

Insert items digunakan ketika ingin menambahkan item ke list di posisi tertentu, tidak selalu di akhir.

- **Extend List**

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

Extend list artinya menambahkan banyak item kedalam list,a atau menggabungkan dua list menjadi satu.

- **Add Any Iterable**

```
thislist = ["apple", "banana", "cherry"]
thistuple = ("kiwi", "orange")
thislist.extend(thistuple)
print(thislist)
```

Add any iterable artinya menambahkan data dari tipe lain (misalnya list, tuple, set) ke dalam list.

5. Remove List Items

- **Remove()**

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

Remove() digunakan untuk menghapus item pertama, sesuai dengan nilai ata data yang diberikan.

- **Pop**

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

Pop digunakan untuk menghapus item tertentu. Jika indeks tidak ditulis maka yang akan terhapus adalah item terakhir.

- **Del**

```
del thislist[0]
del thislist
```

Del digunakan ketika ingin menghapus item atau seluruh list.

- **Clear()**

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

Apabila menggunakan clear maka akan menghasilkan list menjadi kosong dan itemnya hilang. Secara sederhana diartikan, list tetap ada, tetapi tidak ada isinya.

6. Loop List

- **Loop Through a List**

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

Loop through a list artinya menampilkan ata memproses isi list satu per satu. Jadi, setiap isi list akan ditampilkan satu per satu.

- **Loop Through the Index Numbers**

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

Apabila menggunakan loop through the index numbers maka akan menampilkan isi list berdasarkan posisi.

- **Using a While Loop**

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

Using a while loop berarti menampilkan isi list satu per satu sebagai output, selama syarat masih benar.

- **Looping Using List Comprehension**

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

Looping using list comprehension dingunakan untuk menghasilkan list atau output baru di dalam list lama.

7. List Comprehension

- **List Comprehension**

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

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

print(newlist)
```

List comprehension merupakan cara singkat membuat list baru dari list yang sudah ada sebelumnya.

- **Struktur (Syntax)**

```
newlist = [expression for item in iterable if condition == True]
```

- **Contoh lainnya**

i. Membuat list dari angka:

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

ii. Mengambil angka kurang dari 5:

```
newlist = [x for x in range(10) if x < 5]
```

iii. Memngubah menjadi huruf kapital

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

8. Sort List

- **Sort List Alphanumerically**

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

Sort list alphanumerically digunakan untuk mengurutkan isi list menjadi huruf atau abjad atau angka dari kecil ke besar.

- **Sort Descending**

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

Sort descending digunakan untuk mengurutkan isi list dari yang terbesar atau terakhir ke yang terkecil atau awal.

- **Costumize Sort Function**

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

thislist = [100, 50, 65, 82, 23]
thislist.sort(key = myfunc)
print(thislist)
```

Costumize sort function digunakan untuk mengatur urutan list sesuai kemauan kita menggunakan fungsi khusus.

- **Reverse Order**

```
thislist = ["banana", "Orange", "Kiwi", "cherry"]
thislist.reverse()
print(thislist)
```

Reverse order artinya membalikkan urutan isi list dari akhir ke awal.

9. Copy List

- **Copy()**

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

Copy digunakan untuk membuat list baru denga nisi ynag sama seperti asal.

- **List()**

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

List digunakan untuk membuat list baru dari list lama menggunkan fungsi list().

- **Slicing[:]**

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

Slicing digunakan untuk menyalin semua elemen dari list ke list baru dengan cara potong semua ([:]).

10. Join List

- **Operator +**

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]
list3 = list1 + list2
print(list3)
```

Digunakan untuk membuat list baru dari dua list. List lama tetap sama, hasilnya list aru gabungan keduanya.

- **Loop + append()**

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]
for x in list2:
  list1.append(x)
print(list1)
```

Digunakan untuk menambahkan item dari list kedua ke list pertama satu per satu. List pertama berubah, sedangkan list kedua tetap sama.

- **Extend()**

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]
list1.extend(list2)
print(list1)
```

Digunakan untuk menambahkan semua item dari list kedua langsung ke list pertama sekaligus.

B. PYTHON TUPLES

1. Tuple

- **Creating a Tuple**

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

Digunakan untuk membuat tuple baru, dengan urutan yang tetap dan tidak bisa diubah.

- **Tuple Length**

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

Digunakan untuk men-cek berapa jumlah item di dalam tuple tersebut.

- **Tuple Items – Data Types**

```
thistuple = ("apple", 1, True)
print(thistuple)
```

Apabila menggunakan metode ini maka tuple bisa campur tipe data. Bisa berupa angka, kata, True atau False.

- **Using the Tuple() Constructor**

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

Ini merupakan metode lain dari membuat tuple menggunakan tuple().

- **Count() Method**

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

Count() digunakan untuk menghitung berapa jumlah suatu item muncul.

- **Index() Method**

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

Index() digunakan untuk mengetahui posisi pertama item di tuple.

2. Access Tuple Item

- **Access Tuple Item**

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

Digunakan untuk menampilkan item ke-2 dari tuple, karena indeks mulai dari 0.

- **Negative Indexing**

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

Negative indexing digunakan untuk menampilkan item tuple terakhir dari tuple menggunakan indeks negatif.

- **Range of Indexes (Slicing)**

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

Digunakan untuk menampilkan beberapa item sekaligus dari index ke-2 sampai ke-4 (tidak termasuk index ke-5).

- **Range with Negative Indexes**

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

Digunakan untuk menampilkan beberapa item dari urutan ke-4 dari belakang sampai ke-2 dari belakang.

3. Update Tuples

- **Change Tuples Values**

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

print(x)
```

Digunakan untuk mengganti isi tuple dengan cara sementara di ubah menjadi list, lalu elemennya diubah, dan dikembalikan menadi tuple.

- **Add Items**

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

Add items digunakan untuk menambahkan elemen baru ke tuple.

- **Remove Items**

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

Remove items digunakan untuk menghapus elemen dari teple dengan membuat tuple baru.

4. Unpack Tuples

- **Unpack a Tuple**

```
fruits = ("apple", "banana", "cherry")
(green, yellow, red) = fruits

print(green)   # apple
print(yellow)  # banana
print(red)     # cherry
```

Output menunjukkan setiap variabel mendapat nilai sesuai posisi di tuple.

- **Using Asterisk**

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

(green, yellow, *red) = fruits

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

Output menunjukkan *red menangkap semua item sisa sebagai list.

- *** in the Middle**

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

(green, *tropic, red) = fruits

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

Output menunjukkan *tropic menangkap semua item tengah, sementara variabel lain tetap sesuai posisi.

5. Loop Tuples

- **Loop Through a Tuple**

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

Output mencetak setiap item di tuple satu per satu sesuai urutan.

- **Loop Through the Index Numbers**

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

Output mencetak setiap item dengan mengakses berdasarkan indeksnya.

- **Using a While Loop**

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

Output mencetak item sama seperti for, tapi dengan loop while dan indeks manual.

6. Join Tuples

- **Join Two Tuples**

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

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

Output menggabungkan isi tuple1 dan tuple2 jadi satu tuple baru ('a', 'b', 'c', 1, 2, 3).

- **Multiply Tuples**

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

Output mengulang seluruh isi tuple dua kali dalam tuple baru.

C. PYTHON SETS

1. Set

- **Create a Set**

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

Output menunjukkan isi set tapi urutannya bisa beda karena set itu *unordered* (tak berurutan).

- **Loop Through a Set**

```
thisset = {"apple", "banana", "cherry"}
for x in thisset:
  print(x)
```

Output mencetak setiap item di set tanpa indeks karena set tidak punya urutan tetap.

- **Check if Item Exists**

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

Output True jika item itu ada di set.

- **Add Set Items**

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

Output menunjukkan set baru yang sudah ditambah "orange".

- **Add Multiple Items**

```
thisset = {"apple", "banana", "cherry"}
thisset.update(["orange", "mango"])
print(thisset)
```

Output menu njukkan set yang ditambah beberapa item sekaligus.

- **Get the Length of a Set**

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

Digunakan untuk melihat jumlah item di set.

- **Join Two Sets (union)**

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}
set3 = set1.union(set2)
print(set3)
```

Output adalah set baru yang gabungkan semua item dari set1 dan set2 tanpa duplikasi.

- **Join Two Sets (update)**

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set1.update(set2)
print(set1)
```

Output menunjukkan set1 sudah diperluas dengan semua item dari set2.

2. Access Set Items

- **Loop Through a Set**

```
thisset = {"apple", "banana", "cherry"}
for x in thisset:
  print(x)
```

Mencetak setiap item di set karena kamu tidak bisa akses berdasarkan indeks.

- **Check if an Item Is Present**

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

Output True kalau item yang dicari ada di set.

- **Check if an Item Is Not Present**

```
thisset = {"apple", "banana", "cherry"}
print("banana" not in thisset)
```

Output False karena “banana” sebenarnya ada di set.

3. Add Set Items

- **Add One Item with** add**()**

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

Output menunjukkan set sudah bertambah item baru "orange" (urutan bisa berbeda karena set tidak berurutan).

- **Add Items From Another Set with update()**

```
thisset = {"apple", "banana", "cherry"}
tropical = {"pineapple", "mango", "papaya"}
thisset.update(tropical)
print(thisset)
```

Output menunjukkan set digabungkan dengan semua item dari tropical.

- **Add Any Iterable with Update()**

```
thisset = {"apple", "banana", "cherry"}
mylist = ["kiwi", "orange"]
thisset.update(mylist)
print(thisset)
```

Output menunjukkan set ditambah semua item dari list mylist.

4. Remove Set Items

- **Remove()**

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

Output menunjukkan set setelah item "banana" dihapus (akan error jika item tidak ada).

- **Discard()**

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

Output menunjukkan set setelah item "banana" dihapus tanpa error walau item tidak ada.

- **Pop()**

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

Output pertama adalah item yang dihapus secara acak, output kedua adalah set yang tersisa.

- **Clear()**

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

Output adalah set kosong setelah semua item dihapus.

5. Loop Sets

- **Loop Through a Set**

```
thisset = {"apple", "banana", "cherry"}
for x in thisset:
  print(x)
```

Mencetak setiap item di set satu per satu tetapi urutannya bisa berbeda karena set tidak berurutan.

6. Join Sets

- **Union()**

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set3 = set1.union(set2)
print(set3)
```

Output adalah set baru yang berisi semua item dari set1 dan set2 tanpa duplikasi.

- **Intersection**

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
z = x.intersection(y)
print(z)
```

Output adalah set baru yang hanya berisi item yang ada di kedua set.

- **Symmetric Difference**

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
z = x.symmetric_difference(y)
print(z)
```

Output adalah set baru yang hanya berisi item yang unik di salah satu set, tapi tidak di kedua set.

7. Python Frozenset

- **Create a Frozenset**

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

Output menunjukkan frozenset yang dibuat dari set biasa dan tipe datanya adalah frozenset.

D. PYTHON DICTONARIES

1. Dictonary

- **Create a Dictonary**

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

Output menunjukkan dictionary dengan 3 item.

- **Dictonary Items**

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

Output menampilkan nilai dari *key* "brand".

- **Duplicate Keys**

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

Output menunjukkan bahwa *duplicate key* "year" ditimpa dan nilai akhirnya adalah 2020.

- **Dictonary Length**

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

Output adalah jumlah item di dictionary.

- **Data Types**

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

Output menunjukkan bahwa dictionary dapat menyimpan berbagai tipe data sebagai values.

- **Dictonary Constructor**

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

Output menunjukkan dictionary yang dibuat menggunakan *dict()* constructor.

2. Access Dictonary Items

- **Accessing Items**

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

Output menampilkan nilai dari *key* "model" yaitu "Mustang".

- **Get()**

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

Output juga menampilkan nilai dari key "model" tapi pakai metode get().

- **Keys()**

```
car = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = car.keys()
print(x)
car["color"] = "white"
print(x)
```

Output pertama memperlihatkan daftar keys, dan output kedua menunjukkan daftar yang sudah diperbarui setelah ditambah key baru.

- **Values()**

```
car = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = car.values()
print(x)
car["year"] = 2020
print(x)
```

Output pertama adalah daftar values, dan output kedua menunjukkan daftar yang berubah setelah nilai year diubah.

- **Items()**

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

Output pertama adalah daftar semua sebagai tuple, dan output kedua menunjukkan daftar yang diperbarui setelah elemen baru ditambahkan.

- **Check if Key Exist**

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
if "model" in thisdict:
  print("Yes, 'model' is one of the keys in the thisdict dictionary")
```

Output mencetak teks konfirmasi karena "model" ada di dictionary.

3. Change Dictonary Items

- **Change Values**

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["year"] = 2018
print(thisdict)
```

Output menunjukkan dictionary yang sudah mengubah value untuk key "year" menjadi 2018.

- **Update Dictonary**

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

Output menunjukkan dictionary yang diperbarui dengan update(), mengganti value key "year" menjadi 2020.

4. Add Items

- **Adding Items**

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

Output menunjukkan dictionary yang sudah ditambah item baru "color": "red".

- **Update Dictonary**

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

Output menunjukkan dictionary yang sudah diperbarui dengan update() sehingga item "color": "red" ditambahkan.

5. Remove Dictonary Items

- **Pop()**

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

Output menunjukkan dictionary setelah item dengan *key* "model" dihapus.

- **Popitem()**

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

Output menunjukkan dictionary setelah last inserted item dihapus.

- **Del Keyword()**

Menghapus item tertentu:

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

Output menunjukkan dictionary setelah item dengan *key* "model" dihapus menggunakan del.

Hapus dictonary:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict
print(thisdict) #this will cause an error because "thisdict" no longer
exists.
```

Output akan error karena dictionary **tidak lagi ada** setelah del thisdict.

- **Clear()**

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

Output adalah dictionary kosong setelah semua item dihapus.

6. Loop Dictonaries

- **Loop Through a Dictonary**

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

Output mencetak semua key di dictionary.

- **Loop Through Dictonary Values**

```
for x in thisdict.values():
  print(x)
```

Output mencetak semua **value** di dictionary.

- **Loop Through Dictonary Keys**

```
for x in thisdict.keys():
  print(x)
```

Output mencetak semua key pakai .keys().

- **Loop Through Dictonary Items**

```
for x, y in thisdict.items():
  print(x, y)
```

Output mencetak semua pasangan key:value**.**

7. Copy Dictonaries

- **Copy a Dictonary**

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

Output menunjukkan dictionary baru (mydict) yang merupakan **salinan** dari thisdict.

- **Copy a Dictonary**

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

Output menunjukkan dictionary baru yang disalin dari thisdict dengan menggunakan dict() constructor.

8. Nested Dictonaries

- **Nested Dictonaries**

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

Output menunjukkan dictionary yang berisi dictionary lain sebagai nilai (nested).

- **Nested Dictonaries – Create With Existing Distonaries**

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

Output adalah dictionary baru yang berisi tiga dictionary yang sudah dibuat sebelumnya.

- **Access Items in Nested Dictonaries**

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

Output menampilkan nilai "name" dari dictionary "child2" di dalam nested dictionary.

- **Loop Through Nested Dictonaries**

```
for x, obj in myfamily.items():
  print(x)
  for y in obj:
    print(y + ":", obj[y])
```

Output mencetak setiap key di level luar diikuti dengan semua key–value di level dalam.