



**UNS**  
UNIVERSITAS  
SEBELAS MARET



# **MODUL PRAKTIKUM PROGRAMA KOMPUTER**

**PROGRAM STUDI TEKNIK INDUSTRI  
UNIVERSITAS SEBELAS MARET**

**TIM ASISTEN LABORATORIUM  
PERANCANGAN DAN OPTIMASI  
SISTEM INDUSTRI 2020**

## MODUL II

### TIPE DATA, VARIABEL, DAN INPUT

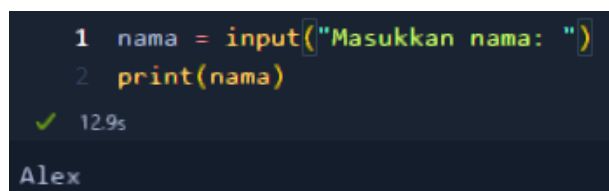
#### A. Tujuan

Berikut merupakan tujuan Praktikum Programa Komputer Modul II.

1. Memahami jenis, implementasi, dan operasi tipe data
2. Memahami pengertian dan penggunaan variabel.
3. Memahami pengertian dan cara penggunaan fungsi *input*.

#### B. Input

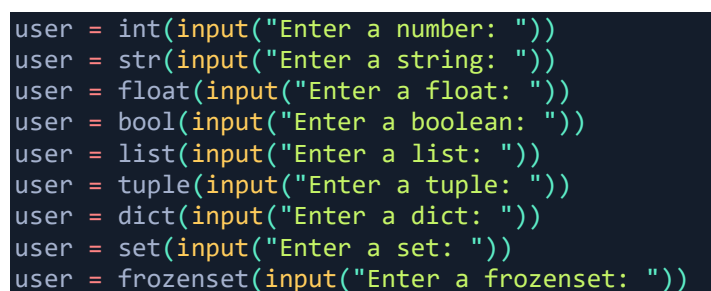
Input adalah salah satu fungsi bawaan dari *Python* yang digunakan untuk menerima masukan atau input dari user yang kemudian hasilnya bisa diproses oleh program untuk menghasilkan output yang relevan. Pada *python* fungsi input ditulis dengan **input()**. Fungsi input dapat menerima berbagai tipe data yang dimasukkan oleh user. Contoh penulisan input:



```
1 nama = input("Masukkan nama: ")
2 print(nama)
✓ 12.9s
Alex
```

**Gambar 1** Contoh Penulisan Input

Secara default fungsi input akan menerima tipe data string, tetapi kita bisa mendefinisikan jenis tipe data yang hanya bisa diterima oleh fungsi input dengan cara mendefinisikannya sebelum fungsi:



```
user = int(input("Enter a number: "))
user = str(input("Enter a string: "))
user = float(input("Enter a float: "))
user = bool(input("Enter a boolean: "))
user = list(input("Enter a list: "))
user = tuple(input("Enter a tuple: "))
user = dict(input("Enter a dict: "))
user = set(input("Enter a set: "))
user = frozenset(input("Enter a frozenset: "))
```

**Gambar 2** Contoh Penulisan Penentuan Tipe Data dalam Input

Jika user memasukkan input tidak sesuai definisi nya maka fungsi input akan error, misalnya pada contoh ini harusnya user memasukkan input berupa angka atau integer (*int*), tetapi karena user memasukkan teks atau string (**str**) maka program error.

```
1 user = int(input("Enter your ID: "))
5.7s

-----
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_14816\3911180306.py in <module>
----> 1 user = int(input("Enter your ID: "))

ValueError: invalid literal for int() with base 10: 'tony'
```

**Gambar 3** Contoh *Error* dalam Input

## C. Tipe Data

*Python* merupakan salah satu bahasa pemrograman yang populer karena kemudahan dan berbagai kelebihan yang ditawarkan dan telah digunakan di berbagai bidang. Dalam pengembangan program, pemahaman yang baik tentang tipe data sangat penting. Tipe data digunakan untuk mengklasifikasikan nilai atau objek dalam program dan juga digunakan untuk menentukan operasi yang dapat dilakukan pada nilai tersebut

Untuk mengecek tipe data pada variabel atau value dapat dilakukan dengan fungsi **type()**

```
1 nama = "Posi"
2 tahun = 2020
3 print(type(nama))
4 print(type(tahun))
5 print(type("TI UNS"))
0.9s

<class 'str'>
<class 'int'>
<class 'str'>
```

**Gambar 4** Pengecekan Tipe Data pada Variabel

### 1. Numerik

Tipe data numerik pada *python* digunakan untuk menyimpan nilai numerik atau sesuatu yang dapat dihitung seperti:

#### a. Integer

Integer adalah tipe data dalam pemrograman yang merepresentasikan bilangan bulat, baik positif, negatif, atau nol, tanpa desimal. Pada *python* integer ditulis angka saja tanpa tanda kutip. Contoh **int** (integer) adalah: **1**, **10**, **-15**, **999**

Contoh operasi sederhana pada integer:

```

1 a = 10
2 b = 20
3 c = a + b
4 print(c)
✓ 0.4s
30

1 a = 9
2 b = 3
3 print(a**b)
✓ 0.1s
729

```

**Gambar 5** Contoh Operasi Sederhana

**b. Float**

Tipe data float adalah tipe data numerik yang digunakan untuk merepresentasikan bilangan pecahan atau desimal. Tipe data *float* dapat merepresentasikan bilangan dengan presisi hingga 7 digit di belakang koma. Contoh tipe data float adalah: **3.14**, **0.5**, **1.030**

```

1 pi = 3.14
2 r = 7
3 luas = pi * (r ** 2)
4 print(f"luas lingkaran dengan jari-jari {r} adalah {luas} cm\u00b2")
✓ 0.6s
luas lingkaran dengan jari-jari 7 adalah 153.86 cm²

```

**Gambar 6** Contoh Tipe Data *Float*

**c. Complex**

Tipe data complex di *python* digunakan untuk merepresentasikan bilangan kompleks. Bilangan kompleks terdiri dari dua bagian, yaitu bagian *real* (*real part*) dan bagian imajiner (*imaginary part*), yang ditulis dalam bentuk **a + bj**, di mana a adalah bagian real dan b adalah bagian imajiner.

```

1 z = 2 + 3j
2
3 # mengambil bagian real dan imajiner dari bilangan kompleks
4 real_part = z.real
5 imaginary_part = z.imag
6
7 print("Bilangan kompleks:", z)
8 print("Bagian real:", real_part)
9 print("Bagian imajiner:", imaginary_part)
10
✓ 0.1s
Bilangan kompleks: (2+3j)
Bagian real: 2.0
Bagian imajiner: 3.0

```

**Gambar 7** Contoh Tipe Data Complex

## 2. *String*

*String* adalah tipe data untuk merepresentasikan teks atau karakter. Pada *python* penulisan string harus diapit dengan tanda petik baik tunggal ('...') maupun gkamu ("..."), tetapi pada satu string **tidak bisa** menggunakan gabungan keduanya ("...')

Contoh string: "Messi", "10", "7X Ballon d'Or", 'PSG'

```
1 artist = "Arctic Monkeys"
2 album = "AM"
3 year = 2013
4 print(f"{artist} released {album} in {year}")
✓ 0.5s
Arctic Monkeys released AM in 2013
```

**Gambar 8** Contoh Tipe Data *String*

Pada *Python* dikenal juga istilah *string method*, *String method* pada *Python* adalah sekumpulan fungsi atau metode bawaan (*built-in*) dalam bahasa pemrograman *Python* yang digunakan untuk memanipulasi atau mengolah data dalam bentuk string. Dengan menggunakan *string method*, kita dapat melakukan berbagai operasi pada string seperti memanipulasi, menggabungkan, memotong, dan lain sebagainya. Berikut adalah daftar *string method*:

**Tabel 1** Daftar *method* pada string

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isascii()	Returns True if all characters in the string are ascii characters
isdecimal()	Returns True if all characters in the string are decimals
isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isprintable()	Returns True if all characters in the string are printable
isspace()	Returns True if all characters in the string are whitespaces

istitle()	Returns True if the string follows the rules of a title
isupper()	Returns True if all characters in the string are upper case
join()	Converts the elements of an iterable into a string
ljust()	Returns a left justified version of the string
lower()	Converts a string into lower case
lstrip()	Returns a left trim version of the string
maketrans()	Returns a translation table to be used in translations
partition()	Returns a tuple where the string is parted into three parts
replace()	Returns a string where a specified value is replaced with a specified value
rfind()	Searches the string for a specified value and returns the last position of where it was found
rindex()	Searches the string for a specified value and returns the last position of where it was found
rjust()	Returns a right justified version of the string
rpartition()	Returns a tuple where the string is parted into three parts
rsplit()	Splits the string at the specified separator, and returns a list
rstrip()	Returns a right trim version of the string
split()	Splits the string at the specified separator, and returns a list
splitlines()	Splits the string at line breaks and returns a list
startswith()	Returns true if the string starts with the specified value
strip()	Returns a trimmed version of the string
swapcase()	Swaps cases, lower case becomes upper case and vice versa
title()	Converts the first character of each word to upper case
translate()	Returns a translated string
upper()	Converts a string into upper case
zfill()	Fills the string with a specified number of 0 values at the beginning

Contoh penggunaan string *method* **.lower()**

```
password = "imperiale2022"

pwd = input("Enter password: ")
if pwd.lower() == password:
    print("Access granted")
else:
    print("Access denied")

Output:
Access granted
```

**Gambar 9** Contoh Penggunaan *String* method.lower()

### 3. Sequence

Sequence adalah tipe data yang mengandung beberapa elemen yang terurut dan dapat diakses menggunakan indeks. Beberapa contoh tipe data sequence adalah list dan tuple.

#### a. List

List digunakan untuk menyimpan beberapa item dalam satu variabel. Dalam satu list bisa untuk menyimpan banyak tipe data sekaligus. Pada *python*

list ditulis dengan tanda kurung siku dan antar item dipisahkan dengan koma ( , ) [..., ..., ]. Item pada list berupa urutan, dapat diubah, dan memungkinkan nilai duplikat. Item pada list dapat diakses dengan teknik *slicing* menggunakan indeks dari list, item pertama pada list memiliki indeks [0], item kedua memiliki indeks [1] dst. Selain bisa diakses menggunakan teknik *slicing*, isi dari list juga dapat dimodifikasi misalnya diganti, ditambah, digabung, atau dihapus menggunakan list *methods*

`ini_list = [1, "dua", 3.0, "empat", 5]`

contoh *slicing* pada list:

Tabel 2 Contoh *Slicing* List

<code>x[0]</code>	Mengambil elemen paling awal, dengan index 0 dari list x.
<code>x[5]</code>	Mengambil elemen dengan index 5 dari list x
<code>x[-1]</code>	Mengambil elemen dengan index paling belakang ke-1 dari list x
<code>x[3:5]</code>	Membuat list dari anggota elemen list x dengan index 3 hingga sebelum index 5 (index 3-4)
<code>x[:5]</code>	Membuat list dari anggota elemen List x paling awal hingga sebelum index 5 (index 0-4)
<code>x[-3:]</code>	Membuat list dari anggota elemen List x mulai index ke-3 dari belakang hingga paling belakang.
<code>x[1:7:2]</code>	Membuat list dari anggota elemen List x dengan index 1 hingga sebelum index 7, dengan "step" 2

```
1 ini_list = [1, "dua", 3.0, "empat", 5, "decimal", 90, [1, 2, 3, 4, 5]]
2
3 print(ini_list[0])
4 print(ini_list[1:3])
5 print(ini_list[2: ])
6 print(ini_list[-1])
7
8 # mengakses item pada list dalam list
9 print(ini_list[-1][-1])
✓ 0.8s
1
['dua', 3.0]
[3.0, 'empat', 5, 'decimal', 90, [1, 2, 3, 4, 5]]
[1, 2, 3, 4, 5]
5
```

Gambar 10 *Slicing* pada list

➤ **Method Built-in Pada List Python**

Python menyertakan *method built-in* sebagai berikut :

Tabel 3 *Method* pada List

Method	Penjelasan
append()	Menambahkan elemen di akhir <i>list</i>
clear()	Menghapus semua elemen dari <i>list</i>
copy()	Mengembalikan salinan <i>list</i>
count()	Mengembalikan jumlah elemen dengan nilai yang ditentukan
extend()	Meambahkan elemen <i>list</i> (atau yang dapat diubah), ke akhir <i>list</i> saat ini
index()	Mengembalikan indeks elemen pertama dengan nilai yang ditentukan
insert()	Menambahkan elemen pada posisi yang ditentukan
pop()	Menghapus elemen pada posisi yang ditentukan
remove()	Menghapus item pertama dengan nilai yang ditentukan
reverse()	Membalik urutan <i>list</i>
sort()	Mengurutkan <i>list</i>

### ➤ Modifikasi List

1. Menambahkan elemen baru dalam *list Python*.

Ada 3 cara umum yang biasa digunakan dalam menambahkan data pada *list* **append()**, **extend()**, dan **insert()** adapun fungsi dan contoh penggunaannya sebagai berikut.

**Tabel 4** *Method* untuk *New Item* pada *List*

Method	Keterangan
append()	Menambahkan elemen baru di akhir <i>list</i> .
extend()	Menambahkan elemen baru di akhir <i>list</i> secara individual.
insert()	Menambahkan elemen baru pada indeks tertentu.

### ➤ **append(<object>)**

**append()** adalah salah satu *method* yang dapat digunakan untuk manipulasi *list* yaitu untuk menambahkan elemen baru di akhir *list*. Perhatikan contoh berikut.

Harap dicatat bahwa *method list*, termasuk salah satunya adalah **append()**, hanya melakukan modifikasi *list* di *list* itu sendiri dan tidak bisa menghasilkan *list* baru. Perhatikan contoh berikut.

```
ini_list = [1, "dua", 3]
ini_list.append("empat")
print(my_list) # Output: [1, "dua", 3, "empat"]
```

**Gambar 11** Contoh Penggunaan *Append*

### ➤ **extend(<iterable>)**



Method **extend()** berfungsi sama seperti **append()** yakni untuk menambahkan elemen baru pada list namun elemen tersebut ditambahkan secara individual. Nantinya elemen yang kita letakkan di itu akan dipisahkan per karakter. Perhatikan contoh berikut.

Lalu bagaimana jika ingin menggunakan *method* **extend()** namun mengharapkan output yang tidak terpisah seperti itu? Caranya adalah dengan membatasi string tersebut dengan tanda kurung siku ([]).

```
ini_list = [1, "dua", 3]
elemen_baru = ["empat", 5, "enama"]
ini_list.extend(elemen_baru)
print(ini_list) # Output: [1, "dua", 3, "empat", 5, "enam"]
```

**Gambar 12** Metode Extend List

### ➤ **insert(<index>, <object>)**

Method **insert()** digunakan apabila kita ingin menyisipkan elemen baru di list pada indeks tertentu. Misalnya kita ingin menambahkan 'apple' pada indeks ke-2.

```
ini_list = [1, "dua", 3]
ini_list.insert(1, "empat")
print(ini_list) # Output: [1, "empat", "dua", 3]
```

**Gambar 13** Metode Insert List

### ➤ **Menghapus Elemen Dalam List Python.**

Untuk menghapus nilai di dalam list *python*, Kamu dapat menggunakan *command* **pop()** jika Kamu tahu persis elemen yang Kamu hapus. Kamu dapat menggunakan metode **remove()** jika Kamu tidak tahu persis item mana yang akan dihapus.

### ➤ **Menggunakan *method***

Ada 2 *method* umum yang dapat digunakan untuk menghapus elemen list, yaitu **remove()** dan **pop()**. Lihat tabel di bawah ini.

**Tabel 5** *Method* untuk Menghapus *Item* pada List

Method	Keterangan
remove()	Menghapus item berdasarkan nama item/objek
pop()	Menghapus item terakhir dari list, serta bisa juga untuk menghapus item berdasarkan indeks.

Khusus untuk *method* **pop()** bisa digunakan dengan dua cara. Simak contoh penggunaannya sebagai berikut.

➤ **remove(<object>)**

*Method* **remove()** akan menghapus item di dalam sebuah list berdasarkan nama item atau elemen tersebut.

```
ini_list = [1, "dua", 3, "empat", 5]
ini_list.remove(3)
print(my_list) # Output: [1, "dua", "empat", 5]
```

**Gambar 14** Metode Remove List

➤ **pop(<index>)**

Jika kita ingin menghapus elemen list berdasarkan indeks, kita bisa menggunakan *method* **pop()**. Perhatikan cara penggunaan *method* **pop()** dengan indeks tertentu.

Jika tidak memasukkan nomor indeks ketika menggunakan **pop()**, maka yang akan terhapus adalah objek atau elemen paling akhir yang ada di dalam list.

```
ini_list = [1, "dua", 3, "empat", 5]
elemen_terakhir = ini_list.pop()
print(elemen_terakhir) # Output: 5
print(ini_list) # Output: [1, "dua", 3, "empat"]

ini_list.pop(1)
print(ini_list) # Output: [1, 3, "empat"]
```

**Gambar 14** Metode Pop List

## b. Tuple

Tuple memiliki beberapa persamaan dengan list hanya saja tipe data ini bersifat unchangeable atau item didalam tuple tidak bisa diubah setelah didefinisikan. Penulisan tuple pada *python* menggunakan tanda kurung dan antar item dipisahkan dengan koma (**..., ...**). Item pada tuple dapat diakses dengan teknik slicing juga hanya saja item nya tidak bisa diubah.

**ini\_tuple = (1, "dua", 3.0, "empat", 5, "decimal", 90, (1, 2, 3, 4, 5))**

```
1 ini_tuple = (1, "dua", 3.0, "empat", 5, "decimal", 90, (1, 2, 3, 4, 5))
2
3 print(ini_tuple[3])
✓ 0.1s
empat
```

**Gambar 15** Contoh Penggunaan Tuple

Tuple juga memiliki *method* yaitu **count()** dan **index()**. *Method* **count()** digunakan untuk mengembalikan berapa kali nilai tertentu muncul dalam sebuah tuple dan *method* **index()** digunakan untuk mencari tuple untuk nilai tertentu dan mengembalikan posisi di mana ia ditemukan

**Tabel 6 Python Tuple Method**

Method	Penjelasan
count()	Mengembalikan berapa kali nilai tertentu muncul dalam sebuah tuple
index()	Mencari tuple untuk nilai tertentu dan mengembalikan posisi di mana ia ditemukan

```
1 thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)
2 x = thistuple.count(5)
3 print(x)
4
✓ 0.3s
2
```

**Gambar 15** Contoh Penggunaan *count()*

```
1 thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)
2 x = thistuple.index(5)
3 print(x)
4
✓ 0.2s
5
```

**Gambar 16** Contoh Penggunaan *index()*

#### 4. Dictionary

Dictionary adalah salah satu tipe data di *Python* yang berfungsi sebagai kumpulan pasangan *key-value* yang bersifat mutable (dapat diubah). *Key* pada dictionary bersifat unik dan tidak boleh duplikat. Sementara *value* pada dictionary bisa berupa tipe data apapun seperti integer, string, float, list, tuple, atau bahkan tipe data dictionary itu sendiri. Contoh dictionary dan bagaimana aturan penulisannya pada *python*:

```
1 employee = {
2     'name': 'Erik',
3     'age': 53,
4     'address': 'Jl Mangkunegaran 10'
5 }
6 print(employee['name'])
7 print(employee['age'])
✓ 0.2s
Erik
53
```

**Gambar 17** Contoh *Dictionary*

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

**Gambar 18** Aturan Penulisan *Dictionary*

Kita dapat langsung mengupdate isi dictionary dengan memberikan key dan value nya:

```
employee['hobbies'] = ['fishing', 'reading', 'swimming']
employee['children'] = {'son': 'Ethan', 'daughter': 'Ella'}
print(employee)
```

output:

```
{'name': 'Erik', 'age': 53, 'address': 'Jl Mangkunegaran 10',
'email': 'ETH@manunited.com', 'hobbies': ['fishing', 'reading',
'swimming'], 'children': {'son': 'Ethan', 'daughter': 'Ella'}}
```

**Gambar 19** Update *Dictionary*

Untuk mengakses item pada dictionary juga menggunakan teknik slicing, bedanya pada dictionary harus menyebutkan dulu key dari value yang ingin diakses

<pre>1 print(employee['hobbies']) 2 print(employee['children']['son'])</pre> <p>✓ 0.7s</p> <p>['fishing', 'reading', 'swimming'] Ethan</p>	<pre>1 # slicing pada dictionary 2 print(employee['hobbies'][1:]) 3 print(employee['children']['daughter'])</pre> <p>✓ 0.6s</p> <p>['reading', 'swimming'] Ella</p>
------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Gambar 10** Menghapus *Item Dictionary*

### ➤ Modifikasi Dictionary

Pada dictionary *python* terdapat juga *built-in* method yang merupakan sekumpulan fungsi atau metode yang dapat digunakan untuk memanipulasi data dalam dictionary, seperti menambahkan elemen baru, mengubah elemen yang sudah ada, menghapus elemen, atau mengambil nilai dari sebuah kunci tertentu Beberapa metode dictionary yang sering digunakan di *python* antara lain:

*Python* menyertakan *method built-in* sebagai berikut :

**Tabel 7** Dictionary Method

Method	Penjelasan
<code>clear()</code>	Menghapus semua elemen dari <i>dictionary</i>
<code>copy()</code>	Mengembalikan salinan <i>dictionary</i>
<code>fromkeys()</code>	Mengembalikan <i>dictionary</i> dengan kunci dan nilai yang ditentukan
<code>get()</code>	Mengembalikan nilai kunci yang ditentukan
<code>items()</code>	Mengembalikan daftar yang berisi tuple untuk setiap pasangan nilai kunci
<code>keys()</code>	Mengembalikan <i>list</i> yang berisi kunci <i>dictionary</i>
<code>pop()</code>	Menghapus elemen dengan kunci yang ditentukan
<code>popitem()</code>	Menghapus pasangan <i>key-value</i> yang terakhir dimasukkan
<code>setdefault()</code>	Mengembalikan nilai kunci yang ditentukan. Jika kunci tidak ada: masukkan kunci, dengan nilai yang ditentukan
<code>update()</code>	Memperbarui kamus dengan pasangan <i>key-value</i> yang ditentukan
<code>values()</code>	Mengembalikan daftar semua nilai dalam <i>dictionary</i>

➤ Menambahkan Elemen Dictionary *Python*

Kamu dapat memperbarui dictionary dengan menambahkan entri baru atau pasangan nilai kunci, memodifikasi entri yang ada, atau menghapus entri yang ada seperti ditunjukkan pada contoh sederhana yang diberikan di bawah ini.

a. **setdefault()**

Metode ini mengembalikan nilai dari *key* yang dicari di dalam dictionary. Jika *key* ditemukan, maka nilai yang sesuai dengan *key* akan dikembalikan. Jika *key* tidak ditemukan, maka metode ini akan menambahkan *key* tersebut ke dalam dictionary bersama dengan nilai default yang diberikan oleh pengguna.

```
# Dictionary kosong
data = {}

# Menambahkan kunci "nama" dengan nama "Dendy"
data.setdefault("nama", "Dendy")

# Menambahkan kunci "umur" dengan nilai 21
data.setdefault("umur", 21)

# Menambahkan kunci "nama" dengan nama "Safri"
# (tidak akan berpengaruh karena kunci "nama" sudah ada)
data.setdefault("nama", "Safri")

# Mencetak nilai dictionary
print(data) # Output: {'nama': 'Dendy', 'umur': 21}
```

Gambar 21 Penggunaan `setdefault()`

b. **update()**

Metode ini digunakan untuk menambahkan atau mengganti kunci dan nilai pada dictionary dengan kunci dan nilai dari dictionary lain atau urutan tuple. Jika kunci sudah ada di dictionary, nilai lama akan diganti dengan nilai baru. Contoh penggunaannya adalah sebagai berikut:

```
# membuat dictionary
data = {'nama': 'Dendy', 'umur': 21}

# menambahkan data dari dictionary lain
data.update({'alamat': 'Jl. Kabut'})
print(data) # output: {'nama': 'Dendy', 'umur': 21, 'alamat': 'Jl. Kabut'}

# mengganti data yang sudah ada
data.update({'nama': 'Dendy Halim'})
print(data) # output: {'nama': 'Dendy Halim', 'umur': 21, 'alamat': 'Jl. Kabut'}
```

**Gambar 22** Penggunaan update()

➤ Hapus Elemen Dictionary *Python*

Kita dapat menghapus elemen dictionary individual atau menghapus keseluruhan isi Dictionary. Kita juga dapat menghapus seluruh dictionary dalam satu operasi. Untuk menghapus seluruh dictionary secara eksplisit, cukup gunakan **del** statement. Berikut adalah contoh sederhana :

a. **dict.pop()**

Metode ini digunakan untuk menghapus kunci dan nilai tertentu dari dictionary dan mengembalikan nilai yang dihapus. Jika kunci tidak ditemukan pada dictionary, akan mengembalikan nilai *default* (jika diberikan). Contoh penggunaannya adalah sebagai berikut:

```
# membuat dictionary
data = {'nama': 'Dendy', 'umur': 21, 'alamat': 'Jl. Kabut'}

# menghapus data dengan kunci tertentu
alamat = data.pop('alamat')
print(alamat) # output: 'Jl. Kabut'
print(data) # output: {'nama': 'Dendy', 'umur': 21}

# menghapus data dengan kunci yang tidak ditemukan
telepon = data.pop('no. handphone', 'Tidak ditemukan')
print(telepon) # output: 'Tidak ditemukan'
```

**Gambar 23** Penggunaan dict.pop()

b. **del**

Metode **del** pada *Python* digunakan untuk menghapus elemen dari sebuah dictionary atau variabel yang terdefinisi sebelumnya. Metode ini dapat digunakan dengan dua cara berbeda, yaitu:

```
# membuat dictionary
data = {'nama': 'Dendy', 'umur': 21, 'alamat': 'Jl. Kabut'}

# menghapus elemen dengan kunci 'alamat'
del data['alamat']

# mencetak dictionary setelah elemen dihapus
print(data) # Output: {'nama': 'Dendy', 'umur': 21}
```

Gambar 24 Penggunaan del()

## 5. Set

Tipe data set dalam bahasa *python* adalah tipe data yang digunakan untuk menyimpan beberapa nilai dalam satu variabel dengan memperhatikan ketentuan-ketentuan khusus seperti berikut:

- Nilai yang disimpan dalam variabel tersebut harus unik atau tidak boleh sama satu sama lainnya.
- Nilai yang sudah dimasukkan ke dalam variabel tidak dapat diubah lagi.
- Tipe data set bersifat tidak berurutan sehingga tidak bisa diakses menggunakan *index*.

Pembuatan tipe data set dapat dilakukan menggunakan 2 cara, yaitu dengan kurung kurawal **{}** atau dengan menggunakan **list** yang di-*passing* ke dalam **set()**.

Contohnya adalah sebagai berikut:

```
angkatan_industri = {'2019', '2020', '2021', '2022'}
print(angkatan_industri)
```

Modul ×

C:\Users\nurki\PycharmProjects\pythonProject\venv\Scripts\python.exe

{'2019', '2021', '2022', '2020'}

Gambar 25 Penulisan Tipe Data Set

```
1 list_makanan = ['soto', 'mi ayam', 'soto', 'nasi goreng']
2 set_makanan = set(list_makanan)
3 print(list_makanan)
4 print(set_makanan)
5 #perhatikan perbedaan antara hasil print list dan set!
```

Modul ×

C:\Users\nurki\PycharmProjects\pythonProject\venv\Scripts\python.exe

['soto', 'mi ayam', 'soto', 'nasi goreng']

{'nasi goreng', 'mi ayam', 'soto'}

Gambar 26 Penggunaan list() yang Diubah ke Set

- **Method Build-in Pada Set Python**

*Python* menyertakan *method built-in* sebagai berikut :

**Tabel 8 Python Set Method**

Method	Penjelasan
<code>add()</code>	Menambahkan elemen ke set
<code>clear()</code>	Menghapus semua elemen dari set
<code>copy()</code>	Mengembalikan salinan set
<code>difference()</code>	Mengembalikan set yang berisi perbedaan antara dua set atau lebih
<code>difference_update()</code>	Menghapus item dalam set ini yang juga disertakan dalam set lain yang ditentukan
<code>discard()</code>	Menghapus item yang ditentukan
<code>intersection()</code>	Mengembalikan satu set, yaitu persimpangan dari dua atau lebih set
<code>intersection_update()</code>	Menghapus item dalam set ini yang tidak ada di set lain yang ditentukan
<code>isdisjoint()</code>	Mengembalikan yang mana dua set memiliki persimpangan atau tidak
<code>issubset()</code>	Mengembalikan apakah set lain berisi set ini atau tidak
<code>issuperset()</code>	Mengembalikan apakah set ini berisi set lain atau tidak
<code>pop()</code>	Menghapus elemen dari set
<code>remove()</code>	Menghapus elemen yang ditentukan
<code>symmetric_difference()</code>	Mengembalikan satu set dengan perbedaan simetris dari dua set
<code>symmetric_difference_update()</code>	Menyisipkan perbedaan simetris dari set ini dan lainnya
<code>union()</code>	Mengembalikan set yang berisi gabungan set
<code>update()</code>	Perbarui set dengan set lain, atau iterable lainnya

- **Menambahkan Elemen Set Python**

Kamu dapat memperbarui set dengan menambahkan entri baru atau pasangan nilai kunci, memodifikasi set dapat menggunakan metode **`add()`** untuk menambah entri yang ada seperti ditunjukkan pada contoh sederhana yang diberikan di bawah ini.

- **`add()`**

Metode **`add()`** pada objek set di *Python* digunakan untuk menambahkan elemen baru ke dalam set. Metode ini membutuhkan satu parameter yaitu elemen yang ingin ditambahkan ke dalam set.

```
# Membuat Set
ini_set = {1, 2, 4}
print(ini_set) # output: {1, 2, 4}

# Menambahkan element set secara urut
ini_set.add(3)
print(ini_set) # output: {1, 2, 3, 4}

# Element yang ada tidak dapat ditulis 2 kali
ini_set.add(1)
print(ini_set) # output: {1, 2, 3, 4}
```

**Gambar 27** Penggunaan **`add()`**

- **Hapus Elemen Set Python**

Kamu dapat memperbarui set dengan menghapus entri yang ada dengan menggunakan metode **`discard()`** dan **`remove()`**. Kamu juga menghapus element tertentu dari set yang ada pada set lainnya.



- **discard()**

Metode **discard()** digunakan untuk menghapus elemen tertentu dari set jika elemen tersebut ada di dalam set. Namun, jika elemen tersebut tidak ada di dalam set, metode ini tidak akan melakukan apa-apa.

```
# Membuat Set
ini_set = {1, 2, 3, 4}

# Menghapus element 3 pada set
ini_set.discard(3)
print(ini_set) # output: {1, 2, 4}

# Menghapus element yang tidak ada pada set
ini_set.discard(5)
print(ini_set) # output: {1, 2, 4}
```

Gambar 28 Penggunaan discard()

- **remove()**

Metode **remove()** juga digunakan untuk menghapus elemen tertentu dari set jika elemen tersebut ada di dalam set. Namun, jika elemen tersebut tidak ada di dalam set, metode ini akan menimbulkan error.

```
# Membuat Set
ini_set = {1, 2, 3, 4}

# Menghapus element 3 pada set
ini_set.remove(3)
print(ini_set) # output: {1, 2, 4}

# Menghapus element yang tidak ada pada set
ini_set.remove(5)
print(ini_set) # akan menimbulkan KeyError
```

Gambar 29 Penggunaan remove()

- **difference\_update()**

Metode **difference\_update()** digunakan untuk menghapus semua elemen yang ada di dalam set kedua dari set pertama.

```
# Membuat Set
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}

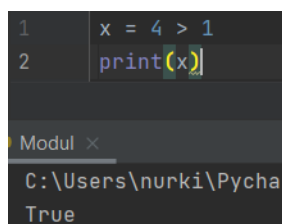
# Menghapus semua set1 yang terdapat pada set2
set1.difference_update(set2)
print(set1) # output: {1, 2}
```

Gambar 30 Penggunaan difference\_update()

## 6. Boolean

Tipe data boolean disebut juga tipe data logika karena tipe data ini dapat menerima operasi logika. Tipe data ini hanya dapat memunculkan 2 nilai yaitu **True** dan **False**. Hal tersebut membuat tipe data boolean dapat digunakan untuk menyusun algoritma pemrograman terutama yang berkaitan dengan perulangan dan kondisi.

Berikut merupakan contoh untuk penggunaan tipe data boolean untuk menguji variabel **a** yang berisi nilai **4 lebih besar dari 1** dengan output **True**.



```

1 x = 4 > 1
2 print(x)

```

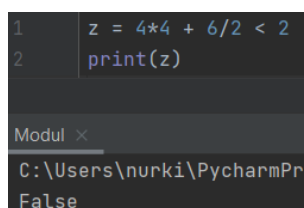
Modul x

C:\Users\nurki\Pychar

True

**Gambar 31** Contoh Penggunaan Tipe Data Boolean True

Sedangkan, berikut percobaan dengan output **False**.



```

1 z = 4*4 + 6/2 < 2
2 print(z)

```

Modul x

C:\Users\nurki\PycharmPro

False

**Gambar 32** Contoh Penggunaan Tipe Data Boolean False

## 7. Binary

Dalam *python*, dikenal juga tipe data *binery* atau biner yang merepresentasikan suatu nilai hanya dengan 2 angka, yaitu 0 dan 1. Dalam sistem biner, 1000 (angka biner yang dibaca satu nol nol nol) memiliki nilai yang sama dengan 8 (angka bulat atau desimal). Dalam *python*, data tipe binary dimulai dengan suatu prefix '0b' kemudian diikuti oleh bilangan biner.

Untuk mengkonversi nilai integer ke biner adalah dengan menggunakan cara manual dan **bin()** seperti contoh di bawah ini.

```

1 #perhitungan manual
2 #contoh : ubahlah bilangan desimal 80 menjadi bilangan biner!
3 # 80/2 = 40, sisa 0
4 # 40/2 = 20, sisa 0
5 # 20/2 = 10, sisa 0
6 # 10/2 = 5, sisa 0
7 # 5/2 = 4, sisa 1
8 # 4/2 = 2, sisa 0
9 # 2/2 = 1 (karena terakhir, maka 1 juga dianggap sisa)
10 #setelah itu, susun sisa-sisa tersebut dari bawah ke atas dan jadikan baris!
11 # jadi bilangan biner dari 80 adalah 1010000
12
13 a = 80
14 bin_a = bin(a)
15 print(bin_a)
16 #atau bisa langsung menggunakan :
17 print(bin(a))
18
Modul x
C:\Users\nurki\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/nurki
0b1010000
0b1010000

```

**Gambar 33** Contoh Penggunaan Tipe Data Binary (bin())

Sedangkan, ada cara untuk mengubah bilangan biner menjadi integer dengan cara manual adalah sebagai berikut. (untuk cara dengan *python* akan menggunakan pelajaran dari modul selanjutnya)

```

#perhitungan manual
#contoh : ubahlah bilangan biner 10101010 menjadi bilangan integer!
#hitung mulai dari digit biner yang paling belakang!
# 0, 0*2^0 = 0
# 1, 1*2^1 = 2
# 0, 0*2^2 = 0
# 1, 1*2^3 = 8
# 0, 0*2^4 = 0
# 1, 1*2^5 = 32
# 0, 0*2^6 = 0
# 1, 0*2^7 = 128
#angka 2 merupakan masukkan atau bilangan baku untuk menghitung bilangan biner.
#setelah itu, jumlahkan seluruh hasil perhitungan di atas!
# jadi bilangan integer dari 10101010 adalah 0+2+0+8+0+32+0+128=170

```

**Gambar 34** Contoh Konversi Binary ke Integer Manual