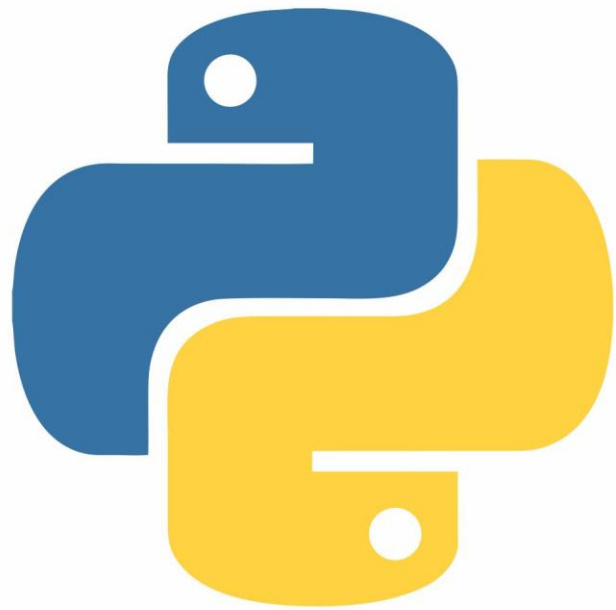


# PYTHON

Pemrograman V



# PYTHON

## Input dan Output

### 4.1 Output

Fungsi bawaan untuk melakukan operasi output adalah `print()`. Seperti yang sudah sering kita praktekan, kita menggunakan fungsi `print()` untuk menampilkan data ke perangkat keluaran standar (layar).

Sintaks lengkap dari fungsi `print()` adalah seperti berikut :

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

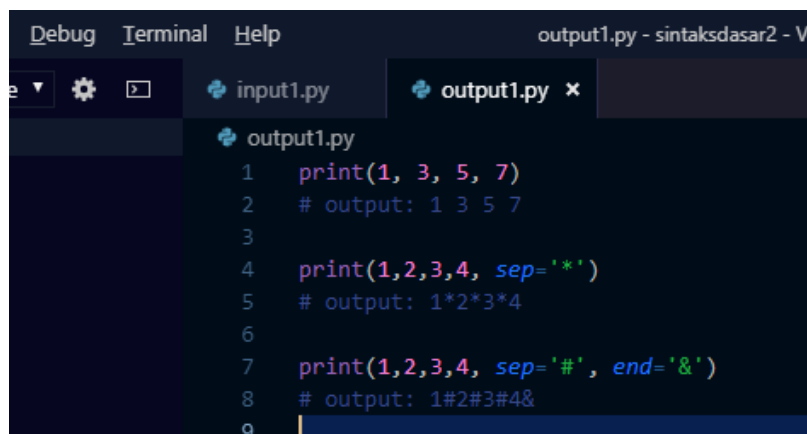
Pada sintaks tersebut, **objects** adalah nilai yang hendak dicetak. Fungsi `print()` akan mengubah semua objek menjadi string terlebih dahulu sebelum dicetak.

**sep** adalah pemisah(separator) yang berfungsi sebagai tanda pemisah antar objek yang dicetak. Defaultnya adalah tanda spasi.

**end** adalah karakter yang dicetak di akhir baris. Defaultnya adalah tanda newline (baris baru).

**file** adalah nama file kemana objek akan dicetak. Defaultnya adalah ke `sys.stdout` (layar).

**flush** adalah opsi apakah keluarannya diflush atau tidak. Digunakan untuk menulis data yang belum tertulis dari buffer ke perangkat output (seperti layar).

A screenshot of a code editor window titled 'output1.py - sintaksdasar2 - Vi'. The editor shows a Python file named 'output1.py' with the following code:

```
1 print(1, 3, 5, 7)
2 # output: 1 3 5 7
3
4 print(1,2,3,4, sep='*')
5 # output: 1*2*3*4
6
7 print(1,2,3,4, sep='#', end='&')
8 # output: 1#2#3#4&
9
```

The code demonstrates the use of the `print()` function with different separators and end characters.

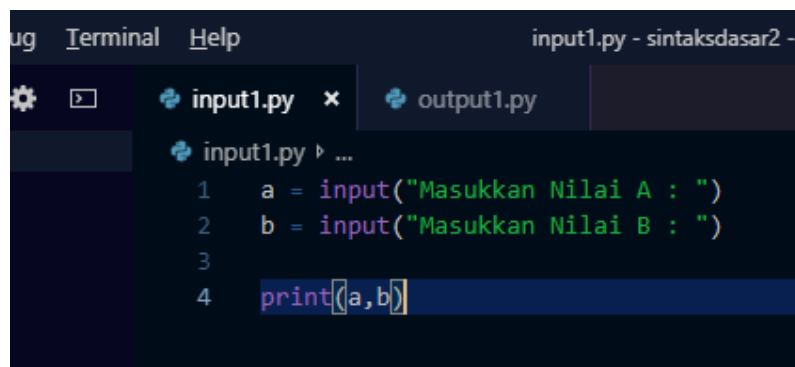
Gambar 24 Contoh Kode Output

### 4.2 Input

Input adalah masukan yang kita berikan ke program. Program akan memprosesnya dan menampilkan hasil outputnya. Input, proses, dan output adalah inti dari semua program komputer.

Fungsi untuk melakukan operasi input adalah fungsi `input()`

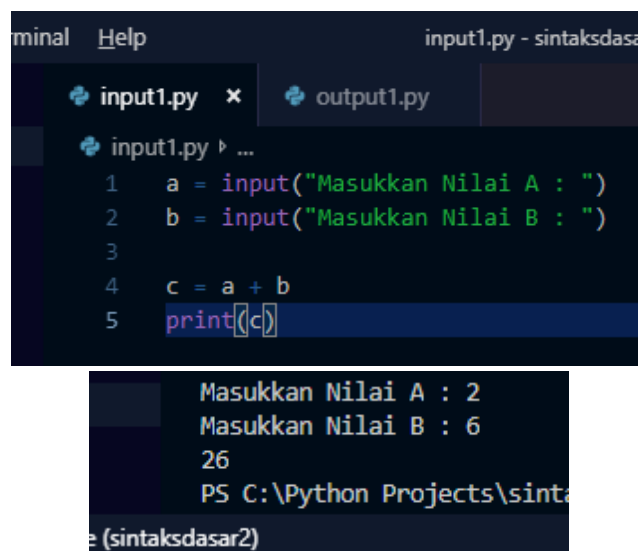
# PYTHON



```
input1.py - sintaksdasar2 -
input1.py x output1.py
input1.py ▸ ...
1 a = input("Masukkan Nilai A : ")
2 b = input("Masukkan Nilai B : ")
3
4 print(a,b)
```

Gambar 25 Contoh Kode Input

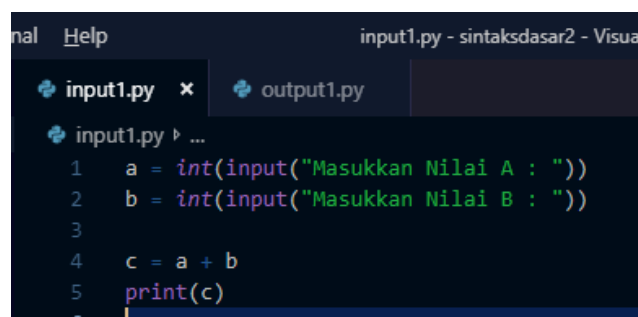
Bila kita menginput bilangan, misalnya integer lewat fungsi `input()`, maka hasil inputan tersebut adalah **string** dan bukan **integer**. Kita harus mengubahnya terlebih dahulu menjadi tipe integer menggunakan fungsi `int()`.



```
input1.py - sintaksdasar2 -
input1.py x output1.py
input1.py ▸ ...
1 a = input("Masukkan Nilai A : ")
2 b = input("Masukkan Nilai B : ")
3
4 c = a + b
5 print(c)
```

```
Masukkan Nilai A : 2
Masukkan Nilai B : 6
26
PS C:\Python Projects\sinta
e (sintaksdasar2)
```

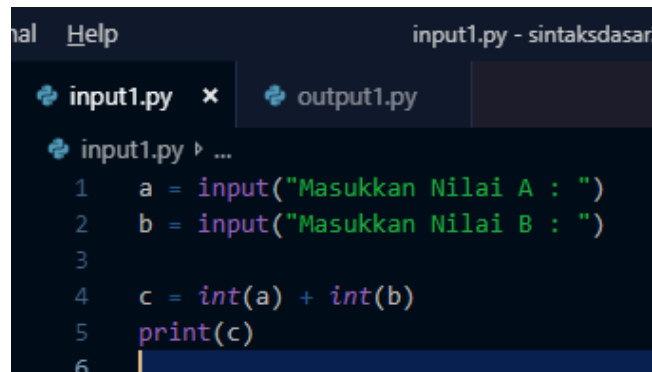
Gambar 26 Contoh Kode Input Integer Tanpa Fungsi `int()`



```
input1.py - sintaksdasar2 - Visual
input1.py x output1.py
input1.py ▸ ...
1 a = int(input("Masukkan Nilai A : "))
2 b = int(input("Masukkan Nilai B : "))
3
4 c = a + b
5 print(c)
6
```

Gambar 27 Fungsi `int()` Cara Pertama

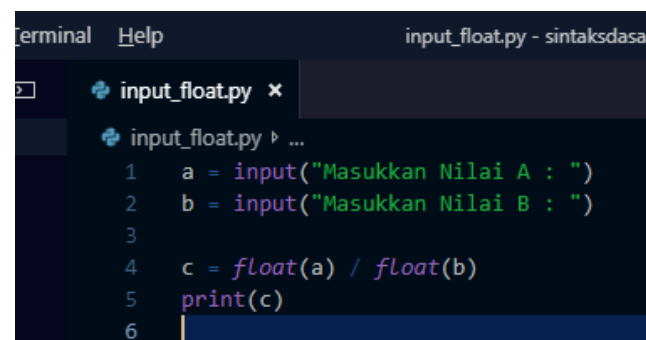
# PYTHON



```
input1.py - sintaksdasar
input1.py x output1.py
input1.py ▸ ...
1 a = input("Masukkan Nilai A : ")
2 b = input("Masukkan Nilai B : ")
3
4 c = int(a) + int(b)
5 print(c)
6
```

Gambar 28 Fungsi int() Cara Kedua

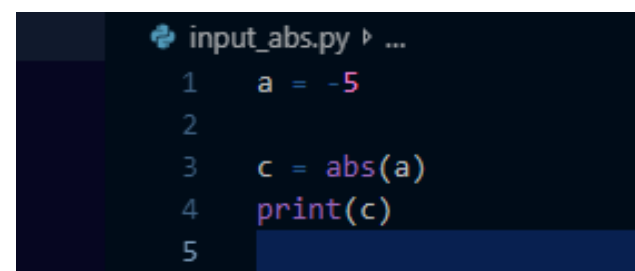
Kemudian ada juga **float()** untuk menginput bilangan pecahan.



```
input_float.py - sintaksdasar
input_float.py x
input_float.py ▸ ...
1 a = input("Masukkan Nilai A : ")
2 b = input("Masukkan Nilai B : ")
3
4 c = float(a) / float(b)
5 print(c)
6
```

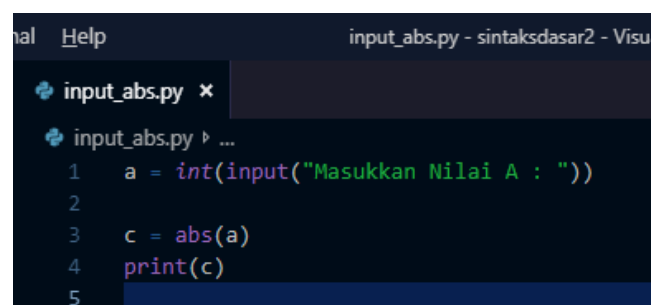
Gambar 29 Fungsi float()

**abs()**. Dengan menggunakan fungsi abs() kita bisa hilangkan tipe data yang ada minusnya .



```
input_abs.py ▸ ...
1 a = -5
2
3 c = abs(a)
4 print(c)
5
```

Gambar 30 Fungsi abs() Statis

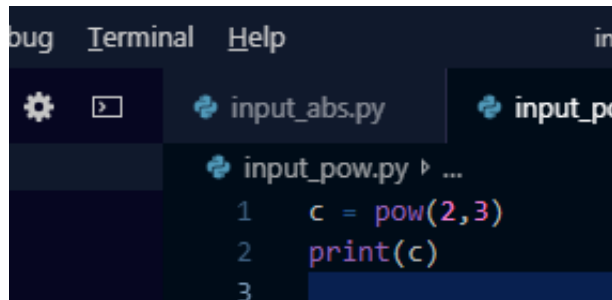


```
input_abs.py - sintaksdasar2 - Visu
input_abs.py x
input_abs.py ▸ ...
1 a = int(input("Masukkan Nilai A : "))
2
3 c = abs(a)
4 print(c)
5
```

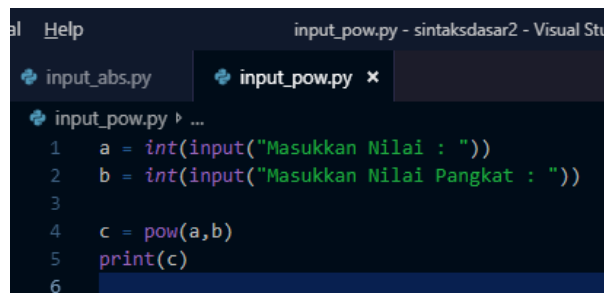
Gambar 31 Fungsi abs() Dinamis

# PYTHON

**pow()**. Adalah fungsi untuk menghitung pangkat.

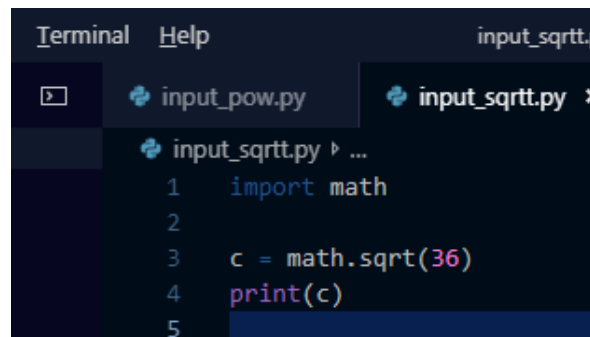


Gambar 32 Fungsi pow() Statis

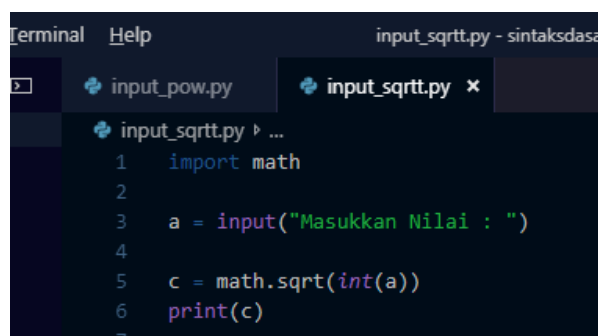


Gambar 33 Fungsi pow() Dinamis

**sqrt()**. Fungsi untuk mencari akar kuadrat dari suatu pangkat.




Gambar 34 Fungsi sqrt() Statis



Gambar 35 Fungsi sqrt() Dinamis

# PYTHON

Kemudian ada fungsi lain seperti **max()**, untuk menampilkan nilai paling akhir. **min()**, menampilkan nilai paling awal. **round()** atau **ceil()** untuk pembulatan keatas. Dan **floor()** untuk pembulatan kebawah.

A screenshot of a code editor window titled 'input\_fungsi\_lain.py - sintaksdasar2 -'. The editor shows a Python script with the following code:

```
1 import math
2
3 print(max(2,1,5)) #output-nya 5
4
5 print(min(2,1,5)) #output-nya 1
6
7 print(round(5.8)) #output-nya 6
8
9 print(math.floor(5.8)) #output-nya 5
10
11 print(math.ceil(5.8)) #output-nya 6
12
```

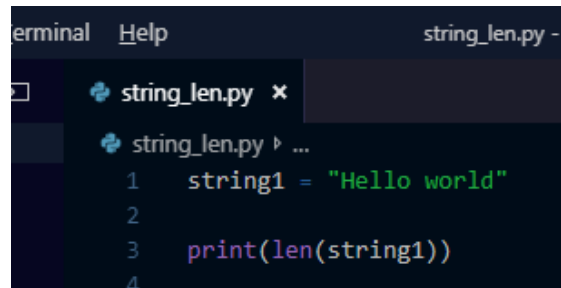
Gambar 36 Fungsi Lain

Pada gambar diatas terdapat baris kode **import math**. Modul math berisi fungsi-fungsi matematika.

# PYTHON

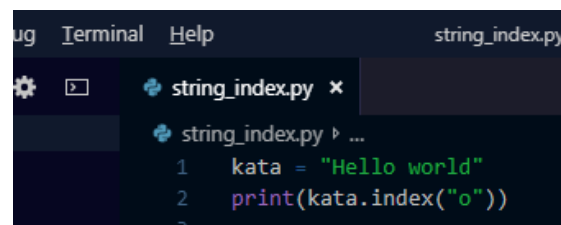
## Operasi String

**len()**. Berfungsi untuk mengembalikan panjang (jumlah anggota) dari suatu objek.



Gambar 37 Fungsi len()

**index()**. Mencari posisi suatu nilai.



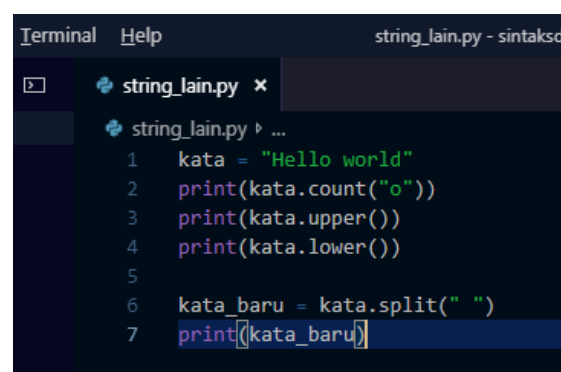
Gambar 38 Fungsi index()

**count()**. Menghitung kemunculan nilai tertentu.

**upper()**. Mengubah string menjadi huruf kapital.

**lower()**. Mengubah string menjadi huruf kecil.

**split()**. Memisah string menjadi item



Gambar 39 Fungsi Lain

## Range Slice.

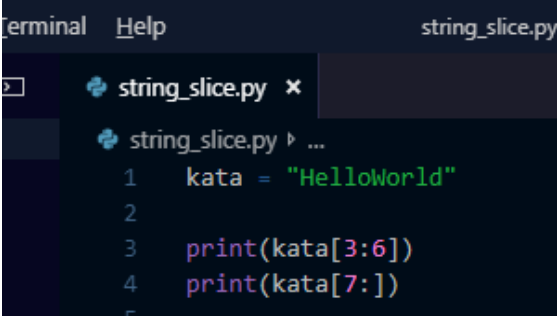
Range Slice menampilkan range karakter dari a mendekati b (limit b), yang diformulasikan dengan.

**nama\_variabel[a:b]**

# PYTHON

a = index karakter yang mulai dicetak.

b = batas akhir karakter yang dicetak, b tidak dicetak atau kosongkan untuk mencetak sampai karakter terakhir.



```
terminal  Help  string_slice.py
string_slice.py x
string_slice.py ▶ ...
1  kata = "HelloWorld"
2
3  print(kata[3:6])
4  print(kata[7:])
5
```

**Gambar 40** Range Slice



# PYTHON

## List

List adalah salah satu tipe data built-in Python, yang dapat digunakan kapan saja tanpa harus meng-import modul terlebih dahulu. List sebenarnya **bukan array** melainkan sebuah collection yang dapat menampung **berbagai objek** dengan **tipe data**. Biasanya array berisi nilai yang hanya satu tipe data saja.

```
# list kosong  
my_list = []
```

```
# list berisi integer  
my_list = [1,2,3,4,5]
```

```
# list berisi tipe campuran  
my_list = [1, 3.5, "Hello"]
```

List juga bisa berisi list lain. Ini disebut list bersarang.

```
# list bersarang  
my_list = ["hello", [2,4,6], ['a','b']]
```

### 5.1 Mengakses Anggota List

Kita bisa mengakses anggota list dengan menggunakan indeksinya dengan format **namalist[indeks]**. Indeks list dimulai dari 0. List yang memiliki 5 anggota akan memiliki indeks mulai dari 0 s/d 4. Mencoba mengakses anggota list di luar itu akan menyebabkan error **IndexError**.

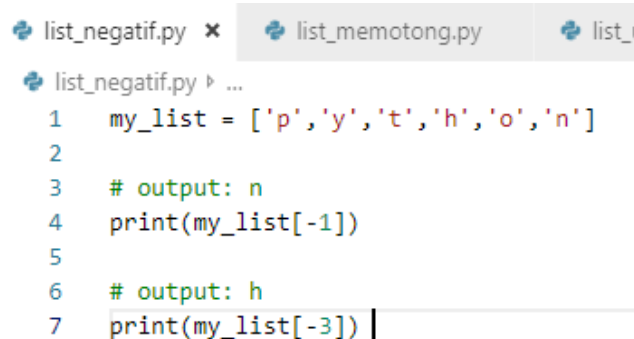
```
list_akses.py ▶ ...  
1  my_list = ["Saya", "belajar", "python", "programming", 2019]  
2  # output: Saya  
3  print(my_list[0])  
4  
5  #output: python  
6  print(my_list[2])  
7  
8  # list dalam list  
9  your_list = ["hello", [1,2,3], "python"]  
10  
11 # output 1  
12 print(your_list[1][0])  
13  
14 # output 3  
15 print(your_list[1][2])  
16  
17 # output hello  
18 print(your_list[0])  
19  
20 # IndexError  
21 #my_list[6]
```

**Gambar 41** Mengakses Anggota List

# PYTHON

## 5.2 List Dengan Indeks Negatif

Python mendukung indeks negatif, yaitu urutan dimulai dari anggota terakhir. Indeks anggota paling belakang adalah -1, kemudian -2, dan seterusnya.

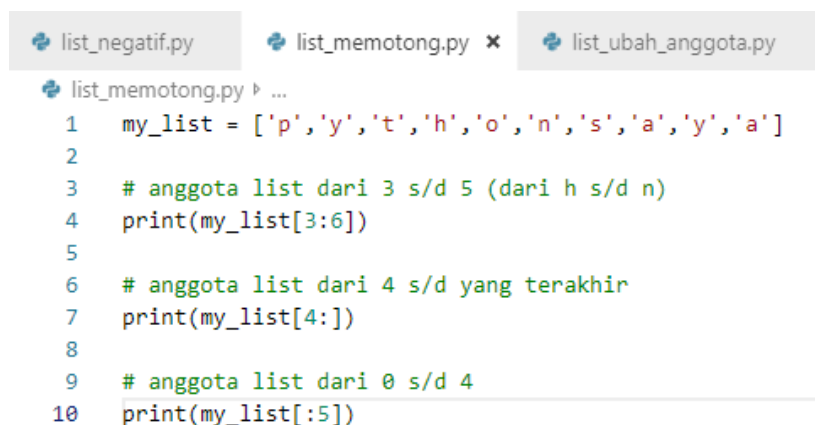


```
list_negatif.py x list_memotong.py list_u...
list_negatif.py ▸ ...
1 my_list = ['p', 'y', 't', 'h', 'o', 'n']
2
3 # output: n
4 print(my_list[-1])
5
6 # output: h
7 print(my_list[-3]) |
```

**Gambar 42** List Dengan Indeks Negatif

## 5.3 Memotong (Slicing) List

Kita bisa mengakses anggota list dari range tertentu dengan menggunakan operator slicing titik dua (:).



```
list_negatif.py list_memotong.py x list_ubah_anggota.py
list_memotong.py ▸ ...
1 my_list = ['p', 'y', 't', 'h', 'o', 'n', 's', 'a', 'y', 'a']
2
3 # anggota list dari 3 s/d 5 (dari h s/d n)
4 print(my_list[3:6])
5
6 # anggota list dari 4 s/d yang terakhir
7 print(my_list[4:])
8
9 # anggota list dari 0 s/d 4
10 print(my_list[:5])
```

**Gambar 43** Memotong List

## 5.4 Mengubah Anggota List

List adalah tipe data yang bersifat mutable, artinya anggotanya bisa diubah. Ini berbeda dengan string dan tuple yang bersifat immutable.

# PYTHON

```
list_negatif.py  list_memotong.py  I
list_ubah_anggota.py ▸ ...
1  # misal ada nilai yang salah
2  ganjil = [1,3,4,7,9]
3  print("Item Awal : ", ganjil)
4
5  # ubah item ke 3 (indeks ke 2)
6  ganjil[2] = 5
7  print(ganjil)
8
9  # mengubah sekali banyak
10 ganjil[2:5] = [11,13,15]
11 print(ganjil)
```

**Gambar 44** Mengubah Anggota List

## 5.5 Menambahkan Anggota List

Fungsi `append()` berguna untuk menambahkan anggota ke dalam list. Selain itu, ada metode `extend()` untuk menambahkan anggota list ke dalam list.

Kita juga bisa menggunakan operator `+` untuk menggabungkan dua list, dan operator `*` untuk melipatgandakan list.

```
list_negatif.py  list_memotong.py
list_tambah_anggota.py ▸ ...
1  ganjil = [1,3,5,7]
2
3  ganjil.append(9)
4  print(ganjil)
5  [1,3,5,7,9]
6
7  ganjil.extend([11,13,15])
8  print(ganjil)
9  [1,3,5,7,9,11,13,15]
```

**Gambar 45** Menambah Anggota List

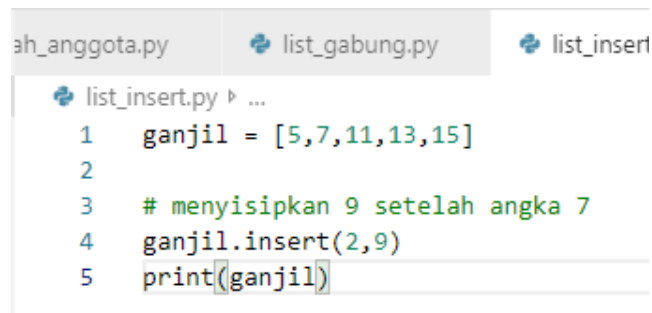
```
list_negatif.py  list_memotong.py
list_gabung.py ▸ ...
1  genap = [2, 4, 6]
2  print(genap + [8, 10, 12])
3  # output [2, 4, 6, 8, 10, 12]
4
5  print(['p','y'] * 2)
6  # output ['p','y','p','y']
```

**Gambar 46** Menggabung List dengan Operator

# PYTHON

## 5.6 Menyisipkan Anggota List

Fungsi `insert()` berfungsi untuk menyisipkan anggota list pada indeks tertentu.



```
list_insert.py ▶ ...
1 ganjil = [5,7,11,13,15]
2
3 # menyisipkan 9 setelah angka 7
4 ganjil.insert(2,9)
5 print(ganjil)
```

**Gambar 47** Menyisipkan Anggota List

## 5.7 Menghapus Anggota List

Kita bisa menggunakan metode `remove()`, `pop()`, atau kata kunci `del` untuk menghapus anggota list. Selain itu kita bisa menggunakan `clear()` untuk mengosongkan list.

Fungsi `pop()` selain menghapus anggota list, juga mengembalikan nilai indeks anggota tersebut. Hal ini berguna bila kita ingin memanfaatkan indeks dari anggota yang terhapus untuk digunakan kemudian.



```
list_tambah_anggota.py list_gabung.py list_insert.py list_hap
list_hapus_anggota.py ▶ ...
1 my_list = ['p', 'y', 't', 'h', 'o', 'n', 's', 'a', 'y', 'a']
2 my_list.remove('p')
3 # output ['y', 't', 'h', 'o', 'n', 's', 'a', 'y', 'a']
4 print(my_list)
5
6 my_list = ['p', 'y', 't', 'h', 'o', 'n', 's', 'a', 'y', 'a']
7 my_list.remove('n')
8 # remove hanya menghapus elemen pertama yang dijumpai
9 # output: ['p', 'y', 't', 'h', 'o', 's', 'a', 'y', 'a']
10 print(my_list)
11
12 # Output 'y'
13 print(my_list.pop(1))
14
15 del my_list[2]
16 print(my_list)
17
18 my_list.clear()
19 # Output []
20 print(my_list)
```

**Gambar 48** Menghapus Anggota List

## 5.8 Mengurutkan Anggota List

Pada saat kita perlu mengurutkan atau menyortir anggota list, kita bisa menggunakan metode `sort()`. Untuk membalik dengan urutan sebaliknya bisa dengan menggunakan argumen `reverse=True`.

# PYTHON

```
list_tambah_anggota.py  list_gabung.py  list_insert.py  list_ha
list_urut_anggota.py ▶ ...
1  alfabet = ['a', 'b', 'd', 'f', 'e', 'c', 'h', 'g', 'j', 'i']
2  alfabet.sort()
3  print(alfabet)
4  # output ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
5
6  alfabet.sort(reverse=True)
7  print(alfabet)
8  # output ['j', 'i', 'h', 'g', 'f', 'e', 'd', 'c', 'b', 'a']
```

**Gambar 49** Mengurutkan Anggota List

## 5.9 Membalik Urutan List

Selain mengurutkan, kita juga bisa membalikkan urutan list dengan menggunakan metode `reverse()`.

```
list_tambah_anggota.py  list_gabung.py  list_insert.py
list_balik_urutan.py ▶ ...
1  alfabet = ['a', 'c', 'd', 'e', 'b']
2  alfabet.reverse()
3  print(alfabet)
4  # output ['b', 'e', 'd', 'c', 'a']
5
```

**Gambar 50** Membalik Urutan List