

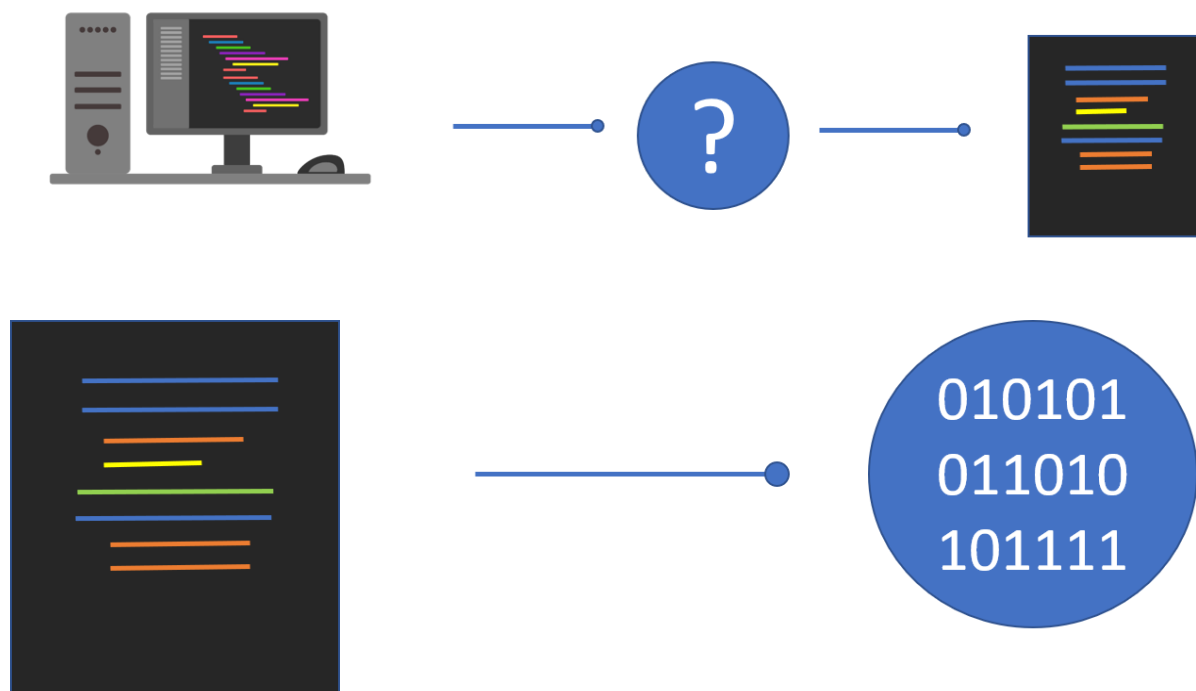
Elemen-Elemen Pemrograman

OBJEKTIF:

1. Mahasiswa Mampu Memahami Bahasa Pemrograman.
2. Mahasiswa Mampu Memulai Pemrograman dengan Python.
3. Mahasiswa Mampu Memahami Python digunakan sebagai Kalkulator.
4. Mahasiswa Mampu Memahami Nilai dan Tipe Data pada Python.
5. Mahasiswa Mampu Memahami Variabel pada Python.
6. Mahasiswa Mampu Memahami Ekspresi pada Python.
7. Mahasiswa Mampu Memahami *Statement Assignment* pada Python.
8. Mahasiswa Mampu Memahami Fungsi *Built-in* pada Python.

1.1 Bahasa Pemrograman

Program adalah rangkaian instruksi-instruksi yang dieksekusi oleh komputer untuk mengerjakan tugas tertentu. Instruksi ini harus ditulis dalam bahasa yang dimengerti komputer yaitu bahasa mesin. Bahasa mesin adalah bahasa yang kosakatanya dalam bilangan biner (bilangan yang terdiri dari angka 0 dan 1). Ilmuwan komputer menciptakan bahasa yang lebih mudah dibaca dan ditulis manusia yang disebut dengan bahasa pemrograman *high-level*.



Program yang ditulis dalam bahasa pemrograman *high-level* tetap harus diterjemahkan ke bahasa mesin.

Masing-masing bahasa pemrograman *high-level* mempunyai program penerjemah yang menerjemahkan program dalam bahasa pemrograman *high-level* ke bahasa mesin.

Bahasa Pemrograman *High-level*

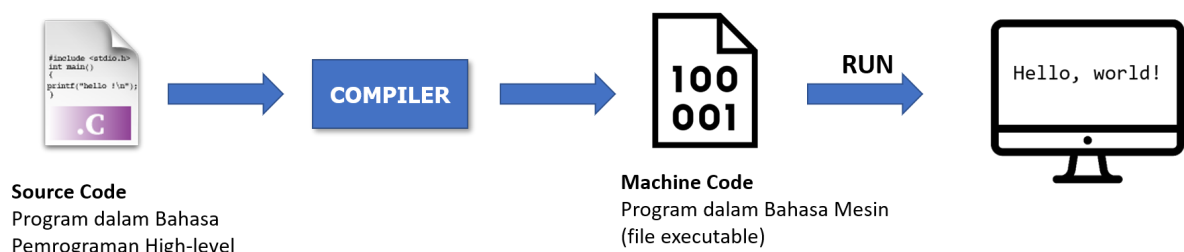
Instruksi dalam bahasa pemrograman *high-level* disebut dengan **statement**. Aturan penulisan statement disebut dengan **syntax**. *Syntax* membuat seolah program yang kita tulis seperti kode. Sehingga program sering disebut dengan kode dan proses menulis program disebut dengan koding (dari Bahasa Inggris *coding*). Program dalam bahasa pemrograman *high-level* harus diterjemahkan ke bahasa mesin. Terdapat dua teknik menerjemahkan, yaitu melalui kompilasi dengan compiler dan melalui interpretasi dengan interpreter.

Bahasa Terkompilasi

Bahasa pemrograman terkompilasi adalah bahasa pemrograman *high-level* yang diterjemahkan dengan compiler. Contoh bahasa pemrograman terkompilasi adalah sebagai berikut:

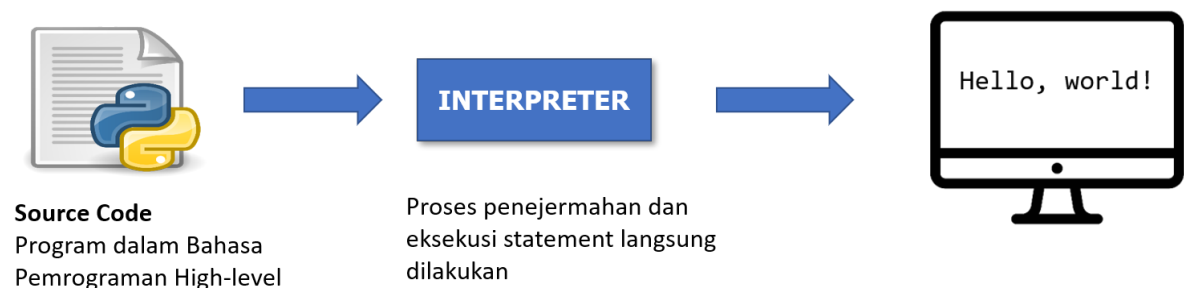
- C
- C++
- Java

Proses kompilasi harus dilakukan setiap melakukan perubahan pada program. Berikut ini merupakan proses kompilasi:



Bahasa Terinterpretasi

Bahasa pemrograman terinterpretasi adalah bahasa pemrograman *high-level* yang diterjemahkan dengan interpreter. Salah satu contoh bahasa pemrograman terinterpretasi adalah Python. Berikut ini merupakan proses interpretasi:

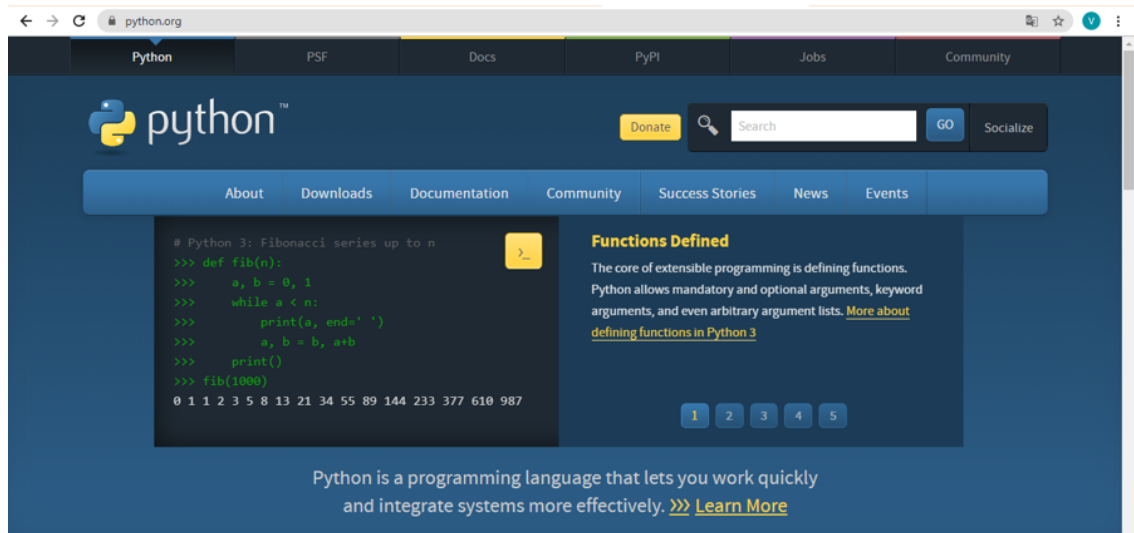


1.2 Memulai Python

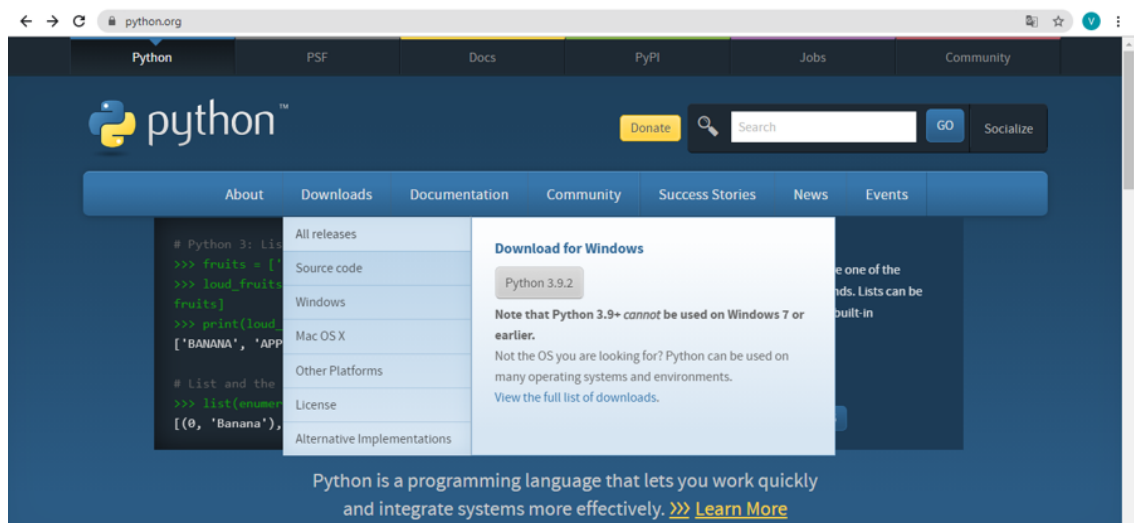
Hal yang pertama harus kita lakukan untuk memulai menulis program dalam Bahasa Python adalah dengan menginstalasi Interpreter Python. *Download* interpreter Python di python.org. File instalasi interpreter Python menyertakan aplikasi IDLE (Integrated Development and Learning Environment). **IDLE** adalah teks editor khusus untuk menulis program Python yang menyertakan berbagai *tools* untuk menjalankan dan menguji program.

Instalasi Python

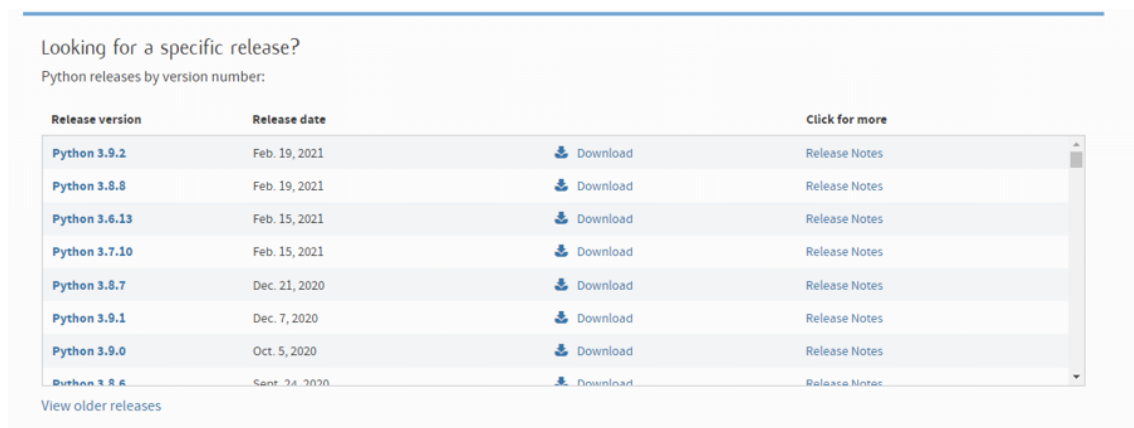
1. Buka website Python www.python.org.



2. Klik tab 'Downloads' lalu klik 'All Release'.



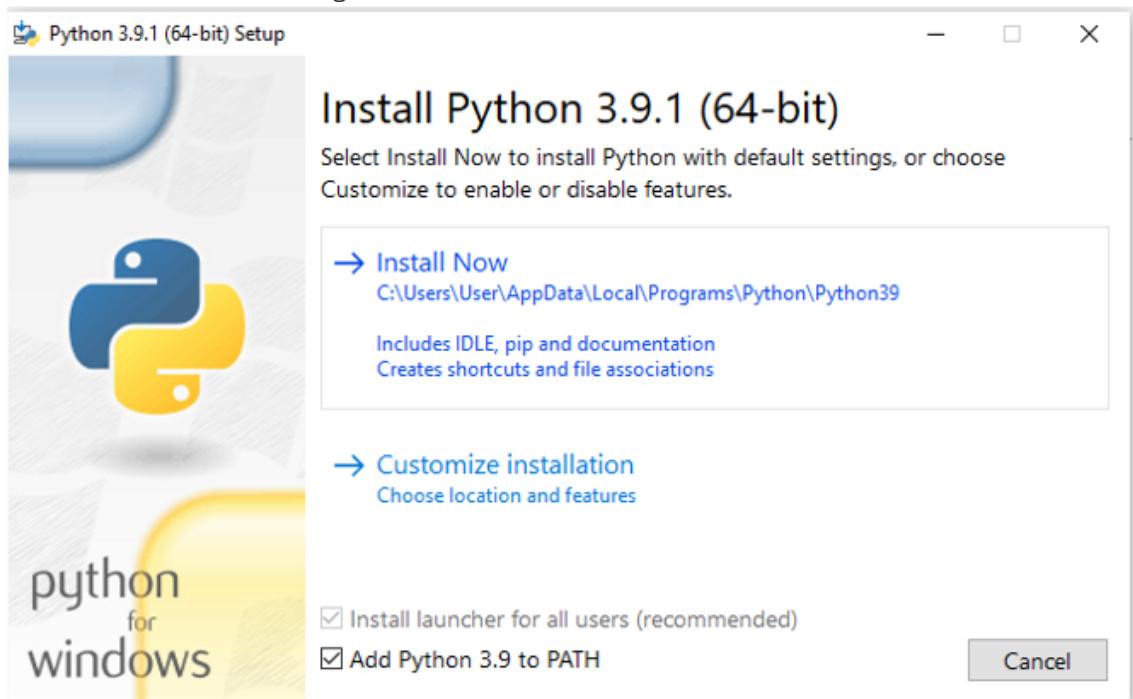
3. Pilih Versi 3.9.1 dan klik 'Download'.



4. Pilih sesuai sistem operasi dan bit pada perangkat anda. Download dan tunggu sampai proses download selesai.

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		429ae95d24227f8fa1560684fad6fca7	25372998	SIG
XZ compressed source tarball	Source release		61981498e75ac8f00adcb908281fadb6	18897104	SIG
macOS 64-bit Intel installer	Mac OS X	for macOS 10.9 and later	74f5cc5b5783ce8fb2ca55f11f3f0699	29795899	SIG
macOS 64-bit universal2 installer	Mac OS X	for macOS 10.9 and later, including macOS 11 Big Sur on Apple Silicon (experimental)	8b19748473609241e60aa3618bbaf3ed	37451735	SIG
Windows embeddable package (32-bit)	Windows		96c6fa81fe8b650e68c3dd41258ae317	7571141	SIG
Windows embeddable package (64-bit)	Windows		e70e5c22432d8f57a497cde5ec2e5ce2	8402333	SIG
Windows help file	Windows		c49d9b6ef88c0831ed0e2d39bc42b316	8787443	SIG
Windows installer (32-bit)	Windows		dde210ea04a31c27488605a9e7cd297a	27126136	SIG
Windows installer (64-bit)	Windows	Recommended	b3fce2ed8bc315ad2bc49eae48a94487	28204528	SIG

5. Centang pilihan 'Add Python 3.9 to PATH' lalu klik 'Install Now' . Tunggu sampai proses instalasi selesai dan bisa digunakan.



Menjalankan Interpreter Python

Kita dapat menjalankan interpreter Python dalam dua modus, yaitu:

MODUS INTERAKTIF

```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help()", "copyright()", "credits()" or "license()" for more information.
>>>
```

Statement Python dapat langsung ditulis dan dieksekusi.

Eksekusi adalah istilah pemrograman yang berarti menjalankan.

MODUS SCRIPT

```
print('Hello, world!')
```

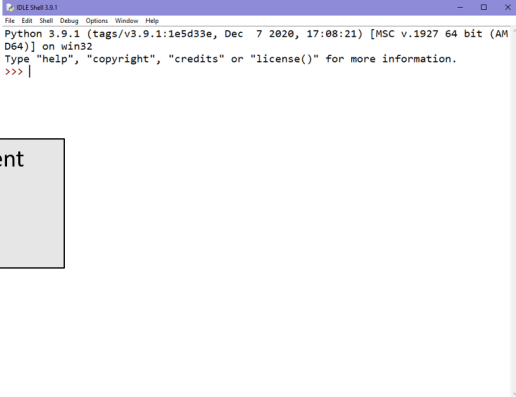
Program dituliskan pada teks editor dan disimpan ke sebuah file. Program dijalankan dengan menjalankan file program dengan interpreter.

Modus Interaktif

Ketika kita membuka IDLE, kita akan mendapatkan Window Shell Python.

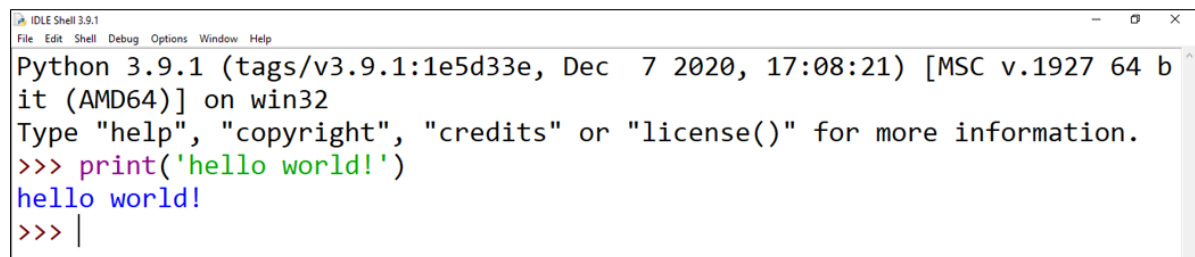
Python prompt →

Jika kita mengetikkan statement Python pada prompt Python, interpreter akan langsung mengeksekusinya



Shell Python memungkinkan kita untuk menggunakan interpreter Python secara interaktif

Mengetikkan `print('Hello world!')`



```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print('hello world!')
hello world!
>>> |
```

Modus Script

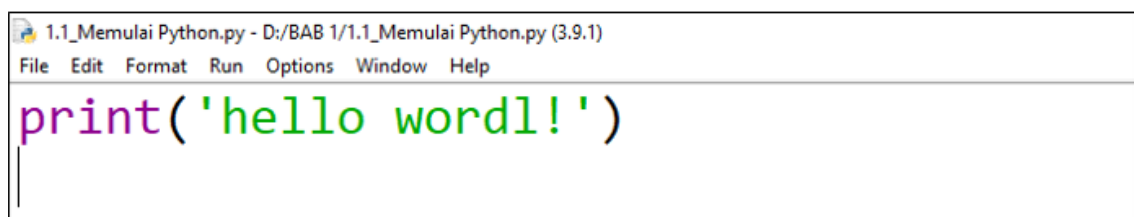
Kita membuat sebuah *file* berisi kumpulan statement Python (disebut script/program), lalu menjalankannya dengan perintah:

```
C:\>python nama_program.py
```

Kita menyimpan program Python pada sebuah file dengan ekstensi **.py**

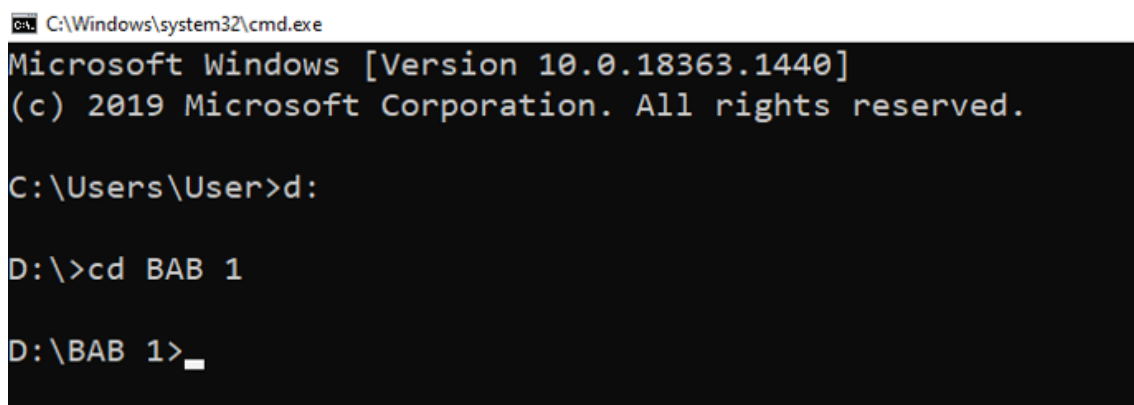
Terdapat dua cara dalam membuka modus script, yaitu dengan CMD dan IDLE. Berikut cara membuka modus script dengan CMD, yaitu:

- Pertama buat suatu file pada modus script IDLE



```
1.1_Memulai Python.py - D:/BAB 1/1.1_Memulai Python.py (3.9.1)
File Edit Format Run Options Window Help
print('hello wordl!')
```

- Lalu buka CMD dan masuk ke direktori file yang sudah disimpan



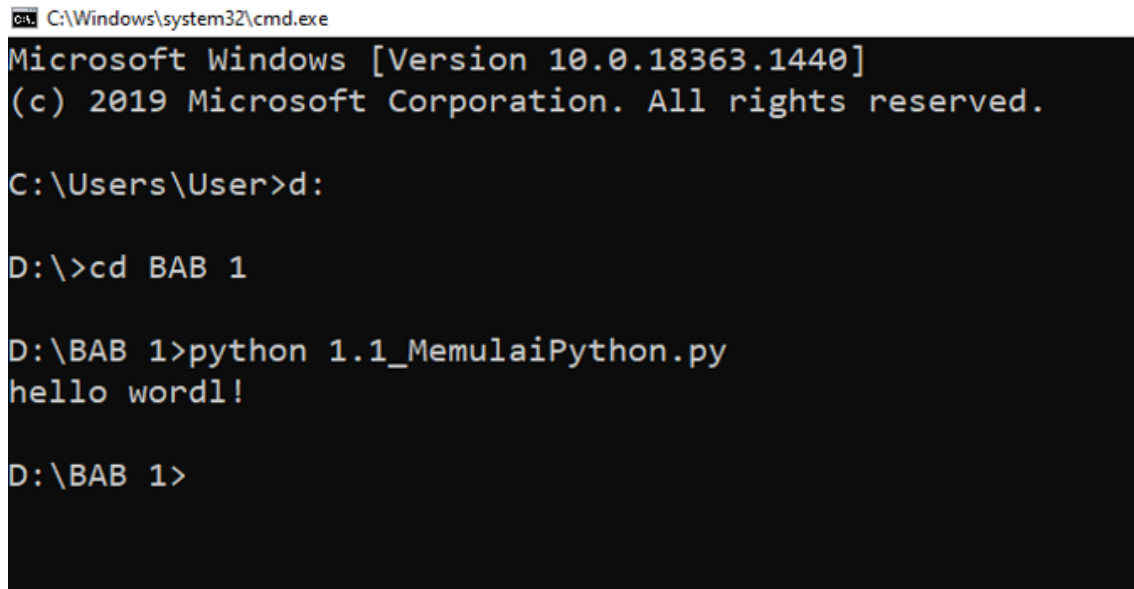
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\User>d:

D:\>cd BAB 1

D:\BAB 1>_
```

- Untuk menjalankan file Python yang sudah tersimpan. Ketikkan python (nama_file).



```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.1440]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\User>d:

D:\>cd BAB 1

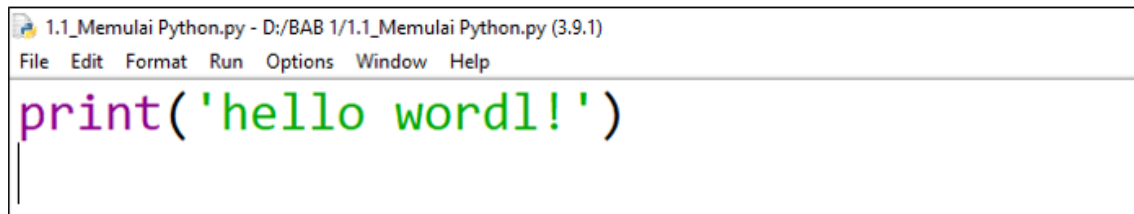
D:\BAB 1>python 1.1_MemulaiPython.py
hello word1!

D:\BAB 1>

```

Berikut cara membuka modus script dengan IDLE, yaitu:

- Pertama buat suatu file pada modus script IDLE



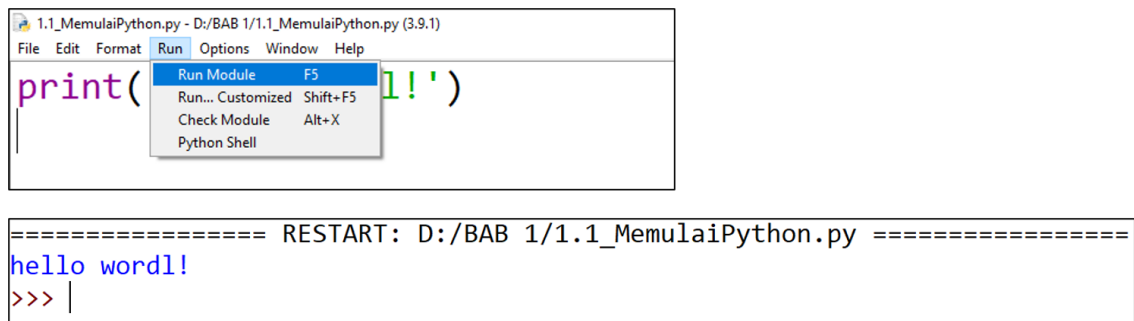
```

1.1_Memulai Python.py - D:/BAB 1/1.1_Memulai Python.py (3.9.1)
File Edit Format Run Options Window Help

print('hello word1!')

```

- Untuk menjalankan program klik Run > Run Module (F5)



```

1.1_MemulaiPython.py - D:/BAB 1/1.1_MemulaiPython.py (3.9.1)
File Edit Format Run Options Window Help
Run Module F5
Run... Customized Shift+F5
Check Module Alt+X
Python Shell

print('hello word1!')

===== RESTART: D:/BAB 1/1.1_MemulaiPython.py =====
hello word1!
>>>

```

1.3 Python sebagai Kalkulator

Dalam modus interaktif, kita dapat menggunakan shell Python sebagai kalkulator. Kita dapat melakukan operasi aritmatika seperti penjumlahan, pengurangan, perkalian, pembagian, dsb. Kita akan menggunakan shell interaktif IDLE untuk mencoba Python sebagai Kalkulator.

```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

Operator

```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 4 + 13
17
>>> 40 + 2
42
>>> 43 - 1
42
>>> 6 * 7
42
>>>
```

Ekspresi
penulisan operator dengan dua nilai

Ekspresi di-**evaluasi** ke sebuah nilai

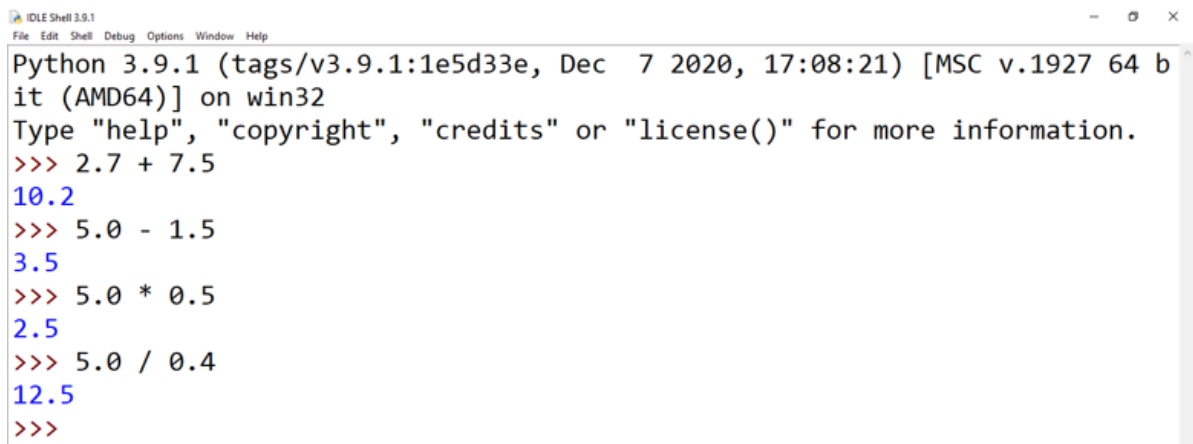
Ln: 11 Col: 4

Operator Aritmatika

Operator yang digunakan untuk melakukan operasi aritmatika disebut dengan operator aritmatika.

Operator Aritmatika	Arti
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian
%	Modulus (Sisa hasil bagi) Contoh: <code>20 % 3</code> menghasilkan <code>2</code>
//	Pembagian <i>floor</i> . Pembagian yang hasilnya dibulatkan ke bilangan bulat terbesar lebih kecil Contoh: <code>10 // 3</code> menghasilkan <code>3</code> .
**	Pemangkatan Contoh: <code>2 ** 3</code> berarti

Kita dapat melakukan operasi aritmatika terhadap bilangan desimal



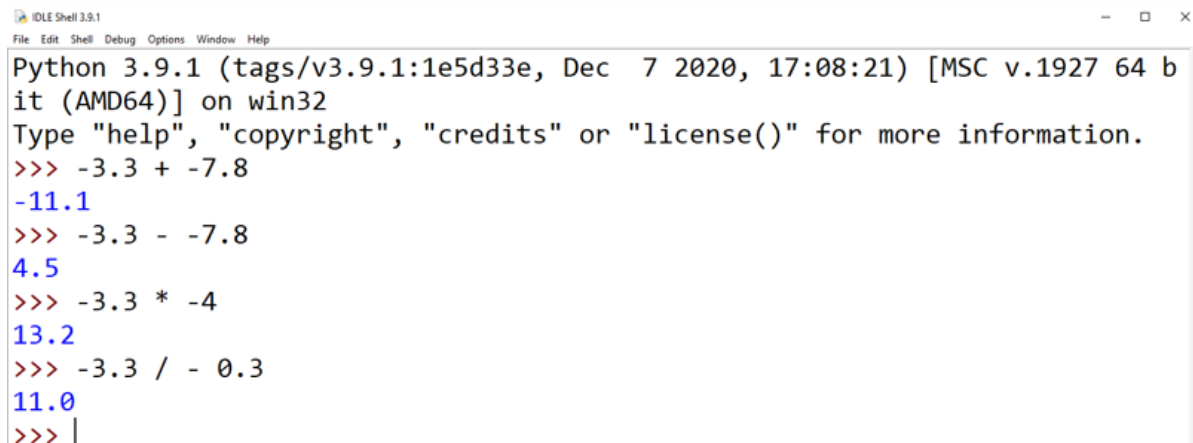
```

Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 2.7 + 7.5
10.2
>>> 5.0 - 1.5
3.5
>>> 5.0 * 0.5
2.5
>>> 5.0 / 0.4
12.5
>>>

```

Kita menuliskan angka desimal dengan menggunakan titik sebagai pemisah desimal (semua bahasa pemrograman mengikuti standar Amerika Serikat yang menggunakan titik sebagai pemisah desimal)

Kita dapat melakukan operasi aritmatika terhadap bilangan negatif



```

Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -3.3 + -7.8
-11.1
>>> -3.3 - -7.8
4.5
>>> -3.3 * -4
13.2
>>> -3.3 / - 0.3
11.0
>>> |

```

Angka negatif ditulis dengan menggunakan operator - sebelum angka tanpa spasi

Operator Precedence

Kita dapat menuliskan ekspresi dengan dua atau lebih operator.

```
>>> 20 + 4 * 10
```

Bagaimana ekspresi di atas dievaluasi? Apakah $20 + 4$ dievaluasi terlebih dahulu lalu dikalikan 10? Apakah $4 * 10$ dievaluasi terlebih dahulu lalu ditambahkan 20? Ekspresi dengan dua atau lebih operator dievaluasi menurut aturan operator precedence (tingkat keutamaan operator).

Operator	Precedence (Tingkat Keutamaan)
<code>**</code>	3 (Tertinggi)
<code>*</code> , <code>/</code> , <code>//</code> , <code>%</code>	2
<code>+</code> , <code>-</code>	1 (Terendah)

Operator yang lebih tinggi tingkat keutamaannya dievaluasi terlebih dahulu

Operator `*` lebih tinggi tingkat keutamaannya dibandingkan operator `+`
→ perkalian dievaluasi terlebih dahulu dibandingkan penjumlahan

$$\begin{array}{c} 20 + 4 * 10 \\ \quad \quad \quad \downarrow \\ 20 + 40 \\ \quad \quad \quad \downarrow \\ 60 \end{array}$$

Pada tabel terdapat operator-operator dengan tingkat keutamaan yang sama. Bagaimana proses evaluasi ekspresi dengan dua atau lebih operator yang mempunyai tingkat keutamaan sama (misalkan `+` dan `-`)? Interpreter mengevaluasinya dari kiri ke kanan.

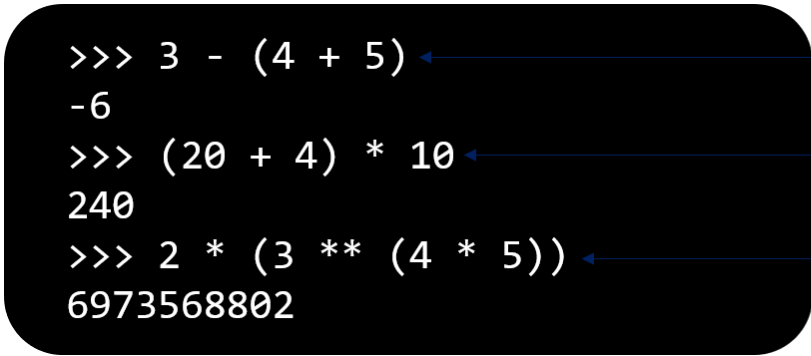
Operator `+` dan `-` mempunyai tingkat keutamaan yang sama

$$\begin{array}{c} 3 - 4 + 5 \\ \quad \quad \downarrow \\ -1 + 5 \\ \quad \quad \downarrow \\ 4 \end{array}$$

Operator	Precedence (Tingkat Keutamaan)
<code>**</code>	3 (Tertinggi)
<code>*</code> , <code>/</code> , <code>//</code> , <code>%</code>	2
<code>+</code> , <code>-</code>	1 (Terendah)

Jika menemukan ekspresi dengan dua atau lebih operator dengan tingkat keutamaan yang sama, interpreter mengevaluasinya dari kiri ke kanan

Kita tidak perlu menghafalkan tingkat keutamaan operator. Kita dapat menggunakan tanda kurung jika kita ingin mengutamakan bagian ekspresi dievaluasi terlebih dahulu.



```

>>> 3 - (4 + 5)
-6
>>> (20 + 4) * 10
240
>>> 2 * (3 ** (4 * 5))
6973568802

```

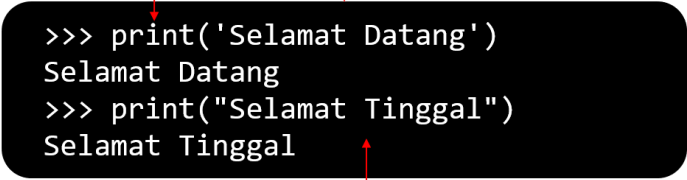
Annotations on the right side of the image:

- 3 - 9 (points to the expression `3 - (4 + 5)`)
- 24 × 10 (points to the expression `(20 + 4) * 10`)
- 2 × 3²⁰ (points to the expression `2 * (3 ** (4 * 5))`)

Selalu gunakan tanda kurung untuk ekspresi yang panjang untuk memudahkan membacanya.

1.4 Nilai dan Tipe Data

Tujuan program komputer adalah memproses informasi. Informasi yang diproses dalam komputer disebut dengan **data**. Setiap data mempunyai nilai dan tipe data. Angka-angka yang kita tuliskan, seperti 3 dan 7.25 adalah **nilai**. Nilai 3 mempunyai tipe data `int` (singkatan dari *integer* yang berarti bilangan bulat). Nilai desimal 7.25 mempunyai tipe data float (singkatan dari *floating point* yang berarti titik mengambang). Dinamakan titik mengambang karena pada bilangan desimal titik pemisah decimal dapat berada diantara digit berapapun. Nilai teks seperti "Hello, world!" mempunyai tipe data str (singkatan dari string yang berarti untaian). Dinamakan untaian karena teks menguntai karakter-karakter. Kita menuliskan nilai tipe data str dengan menuliskan teks di dalam tanda kutip tunggal atau tanda kutip ganda.



Annotations on the left side of the image:

- Statement print digunakan untuk mencetak teks ke layar (points to the `print` function)

Annotations on the right side of the image:

- Nilai string dapat dituliskan dengan diapit tanda kutip tunggal (points to the single quotes in `'Selamat Datang'`)
- Nilai string juga dapat dituliskan dengan diapit tanda kutip ganda (points to the double quotes in `"Selamat Tinggal"`)

```

>>> print('Selamat Datang')
Selamat Datang
>>> print("Selamat Tinggal")
Selamat Tinggal

```

Tabel berikut mendaftar tipe-tipe data Python yang sering digunakan:

Kelompok	Tipe Data	Keterangan	Contoh
Numerik	<code>int</code>	Integer (bilangan bulat)	29, -34
	<code>float</code>	Floating point (bilangan riil) Dituliskan dalam pecahan desimal dengan tanda titik sebagai pemisah desimal	3.14, -2.75
Boolean	<code>bool</code>	Boolean - tipe data yang hanya dapat bernilai <code>True</code> (benar) atau <code>False</code> (salah)	True, False
Teks	<code>str</code>	String (untaian karakter-karakter)	'AB' "Hello" '1.4'

Kita dapat menggunakan statement `type()` untuk mencari tahu tipe data dari suatu data.

```
>>> type(-34)
<class 'int'>
>>> type(3.14)
<class 'float'>
>>> type(-2.75)
<class 'float'>
>>> type(True)
<class 'bool'>
>>> type('Hello')
<class 'str'>
>>> type('1.4')
<class 'str'>
```

Tipe Data Menentukan Operasi Terhadapnya

Kita perlu mengetahui tipe data untuk mengetahui operasi-operasi apa yang dapat dilakukan terhadap tipe data tersebut. Misalnya, operator aritmatika hanya dapat dilakukan terhadap tipe data numerik (`int` dan `float`). Jika kita mencoba melakukan operasi aritmatika terhadap data string kita akan mendapatkan error, contohnya seperti berikut ini:

```
>>> '12' / '3'
Traceback (most recent call last):
  File "<pyshe11#30>", line 1, in <module>
    '12' / '3'
TypeError: unsupported operand type(s) for /: 'str' and 'str'``
```

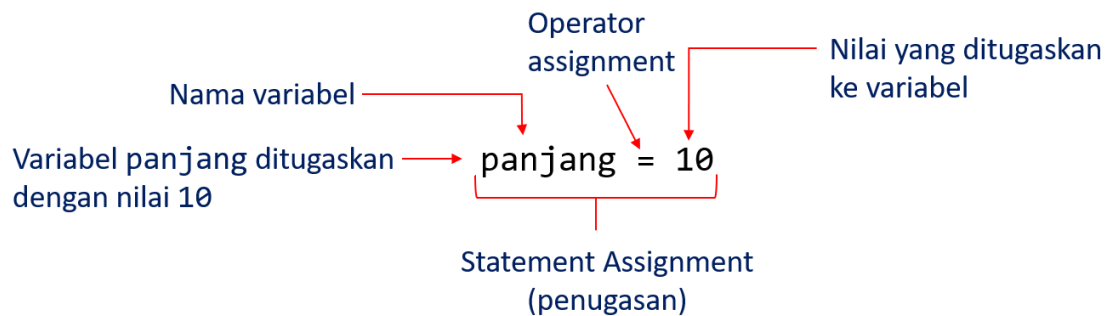
Beberapa (tidak semua) operator aritmatika mempunyai arti lain ketika digunakan terhadap string. Misalnya, operator `+` berarti konkatenasi (menyambung) dua string.

```
>>> '12' + '3'
'123'
>>> 'Hello' + 'world'
'HelloWorld'
```

Kita akan membahas operasi-operasi apa saja yang dapat dilakukan terhadap tipe data string pada bab lain.

1.5 Variabel

Variabel adalah nama yang diasosiasikan ke suatu nilai. Kita menggunakan variabel untuk menyimpan suatu nilai yang nantinya nilai tersebut dapat kita rujuk. Berikut ilustrasi dalam membuat variabel dalam Python:



Contoh:

```
>>> panjang = 10
>>> radius = 17.5
>>> pengguna = 'Budi Susilo'
>>> status = True
```

Kita dapat menggunakan statement `print()` untuk mencetak nilai yang disimpan variabel:

```
>>> panjang = 10
>>> print(panjang)
10
```

Catatan: Nama variabel ditulis dalam tanda kurung tanpa diapit tanda kutip.

Penamaan Variabel

Kita menamakan variabel mengikuti ketentuan penamaan identifier. **Identifier** (pengidentifikasi) adalah nama yang dipilih programmer untuk menamakan bagian-bagian program seperti variabel, fungsi, class, method, dsb.

Ketentuan penamaan identifier:

- Hanya dapat menggunakan kombinasi huruf kecil (a s.d z), huruf besar (A s.d Z), digit (0 s.d 9), atau garis bawah (underscore) `_`
- Tidak dimulai dengan angka
- Tidak menggunakan **keyword** (kata-kata yang sudah digunakan dan mempunyai arti tertentu dalam bahasa Python)

Terdapat 36 keyword dalam Python (versi 3.9), yaitu:

Keyword			
False	break	for	not
None	class	from	or
True	continue	global	pass
peg_parser	def	if	raise
and	del	import	return
as	elif	in	try
assert	else	is	while
async	except	lambda	with

Berikut ini contoh-contoh penamaan variabel yang salah dan benar:

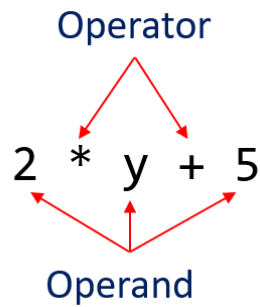
Nama Variabel	Keterangan
<code>var1</code>	Valid
<code>my_number</code>	Valid
<code>9ball</code>	Tidak valid, karena identifier tidak boleh dimulai dengan angka
<code>try</code>	Tidak valid, karena <code>try</code> merupakan keyword Python
<code>\$_nilai</code>	Tidak valid, karena identifier tidak boleh menggunakan karakter lain selain huruf, angka, dan <code>_</code>

Biasakan untuk menggunakan nama variabel yang menjelaskan nilai yang disimpan. Nama variabel `jumlah_kelulusan_2020` lebih baik daripada hanya menamakan sebuah variabel dengan `x`. Berikut ini adalah konvensi programmer untuk nama variabel:

- Gunakan huruf kecil
- Untuk nama variabel dengan lebih dari satu suku kata, pisahkan masing-masing suku kata dengan garis bawah (`_`). Contoh: `nilai_ujian`, `harga_per_unit`

1.6 Ekspresi

Ekspresi adalah kombinasi variabel, nilai, dan operator yang dievaluasi ke suatu nilai. Berikut contoh dari sebuah ekspresi:



Operator adalah simbol yang digunakan untuk melakukan operasi tertentu. **Operand** adalah data (variabel / nilai) yang terhadapnya dilakukan operasi tertentu. Ekspresi dievaluasi menghasilkan sebuah nilai.

Contoh lain dari ekspresi (dalam modus interaktif):

```
>>> y = 5
>>> 2 * y + 5
15
```

Interpreter mengevaluasi ekspresi ini dengan mensubstitusi variabel `y` dengan nilai yang disimpan lalu melakukan kalkulasi.

Catatan: Pada modus interaktif, ketika kita menuliskan ekspresi pada prompt Python dan menekan ENTER, nilai hasil evaluasi ditampilkan pada baris selanjutnya.

Variabel yang berdiri sendiri termasuk sebagai sebuah ekspresi, berikut contohnya:

```
>>> y = 5
>>> y
5
```

Nilai yang berdiri sendiri juga termasuk sebagai sebuah ekspresi, berikut contohnya:

```
>>> 'Hello'
'Hello'
```

Ekspresi vs Statement

Ekspresi berbeda dengan statement; ekspresi dievaluasi ke suatu nilai sedangkan statement dieksekusi untuk melakukan sesuatu. Dalam program, ekspresi tidak ditulis berdiri sendiri namun sebagai bagian dari statement. Misal, kita menuliskan ekspresi pada ruas kanan statement assignment untuk menugaskan sebuah variabel dengan nilai hasil evaluasi ekspresi tersebut:

```
celcius = (5/9)*(fahrenheit - 32)
```

Ekspresi sebagai bagian dari statement assignment

Misal, kita menuliskan ekspresi pada statement print untuk menampilkan nilainya:

```
print(3.14 * y)
```

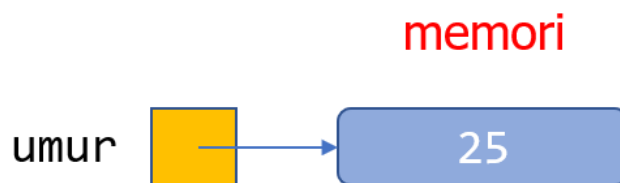
Ekspresi sebagai bagian dari statement print

1.7 Statement Assignment

Statement assignment (penugasan) digunakan untuk membuat variabel dan menugaskannya dengan suatu nilai.

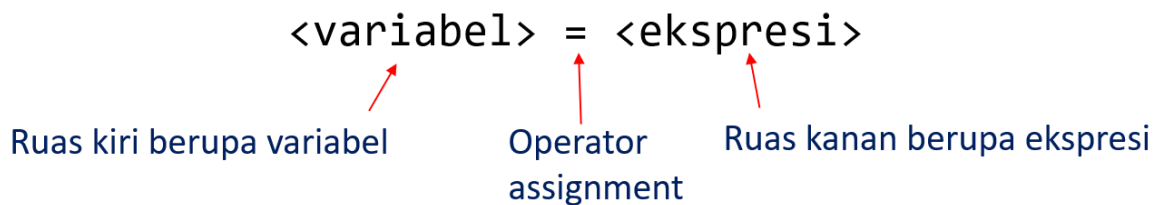
```
umur = 25
```

Ketika statement assignment di atas dieksekusi, komputer menyimpan nilai 25 ke suatu tempat dalam memori, membuat variabel umur, dan mereferensikan variabel tersebut ke lokasi memori tempat nilai 25 disimpan.

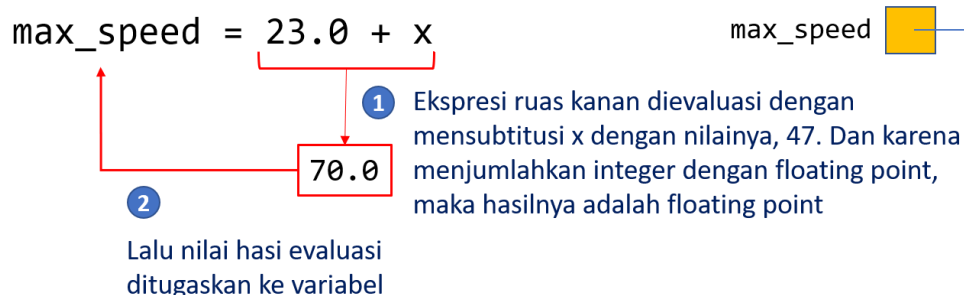
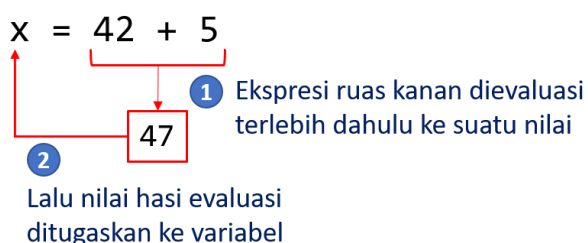


Model memori yang mengilustrasikan asosiasi variabel dengan suatu nilai dalam memori

Berikut ini merupakan bentuk umum penulisan statement assignment:



Contoh:



```
>>> 25 = umur
SyntaxError: cannot assign to literal
```

Variabel dapat diubah nilainya, oleh karena itu disebut dengan variabel yang berarti berubah. Kita mengubah nilai variabel dengan mengugaskannya dengan nilai baru

```
>>> panjang = 10
>>> panjang = 25
```



Ini bukan persamaan namun penugasan

$x = 25$
 $y = 50 + x$
 $x = 30$

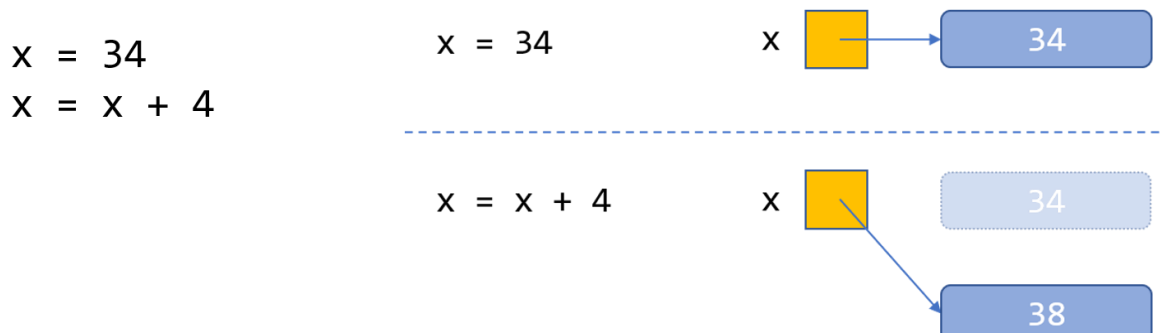
Variabel y ditugaskan dengan nilai 75. Nilai 75 didapat dari hasil penjumlahan 50 dengan nilai x, yaitu 25.

Nilai variabel y tidak berubah meskipun nilai variabel x berubah

```

graph LR
    subgraph "Initial State"
        x1[x = 25]
        y1[y = 50 + x]
    end
    subgraph "After x = 30"
        x2[x = 30]
        y2[y = 75]
        y3[y = 75]
    end
    x1 --> y1
    x2 --> y2
    x2 --> y3
  
```

Kita dapat mempunyai statement assignment yang kedua ruasnya terdapat variabel yang sama:



Operator Assignment Teraugmentasi

Kita dapat menyingkat penulisan statement assignment yang kedua ruasnya terdapat variabel yang sama dengan menggunakan operator assignment teraugmentasi.

`x = x + 4`



`x += 4`



Operator Assignment Teraugmentasi

Semua operator aritmatika mempunyai operator assignment teraugmentasi terkait.

Operator Assignment Teraugmentasi	Contoh Statement	Statement Ekuivalen
<code>+=</code>	<code>c += 7</code>	<code>c = c + 7</code>
<code>-=</code>	<code>d -= 4</code>	<code>d = d - 4</code>
<code>*=</code>	<code>e *= 5</code>	<code>e = e * 5</code>
<code>**=</code>	<code>f **= 3</code>	<code>f = f ** 3</code>
<code>/=</code>	<code>g /= 2</code>	<code>g = g / 2</code>
<code>//=</code>	<code>h //= 2</code>	<code>h = h // 2</code>
<code>%=</code>	<code>i %= 9</code>	<code>i = i % 9</code>

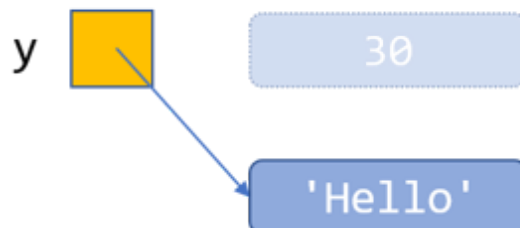
Dalam Python, variabel dapat ditugaskan ulang dengan tipe data berbeda:

```
>>> y = 30
>>> print(y)
30
>>> y = 'Hello'
>>> print(y)
Hello
```

`y = 30`




`y = 'Hello'`



Menugaskan Lebih dari Satu Variabel dengan Nilai Sama

Kita dapat menugaskan lebih dari satu variabel dengan nilai yang sama dalam satu statement assignment seperti berikut:

a = b = c = 10



Statement ini menugaskan nilai 10 ke variabel a, b, dan c

Berikut ini bentuk umum statement assignment yang menugaskan lebih dari satu variabel dengan nilai sama:

```
<var1> = <var2> = ... = <varn> = <ekspresi>
```


Statement Assignment Simultan

Kita dapat menuliskan statement assignment simultan yang menugaskan beberapa variabel sekaligus:

```
<var1>, <var2>, ..., <varn> = <eksp1>, <eksp2>, ..., <eksprn>
```

Contoh:

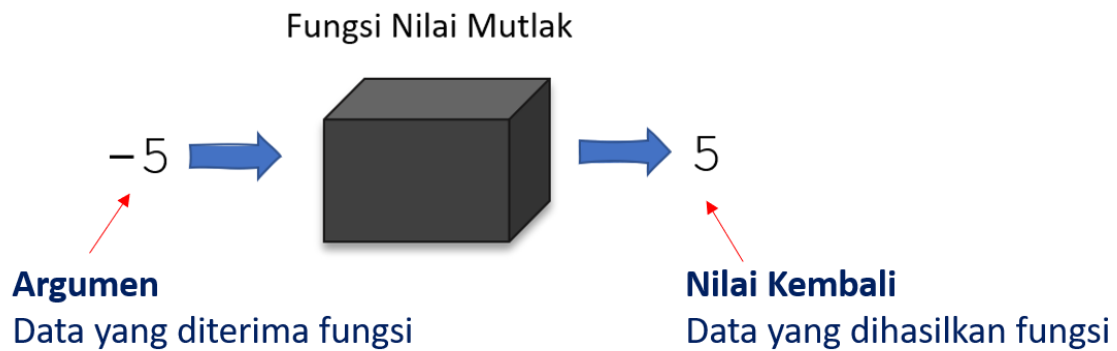
jumlah, selisih = x + y, x - y



```
>>> x = 2
>>> y = 4
>>> x, y = y, x
>>> print(x)
4
>>> print(y)
2
```

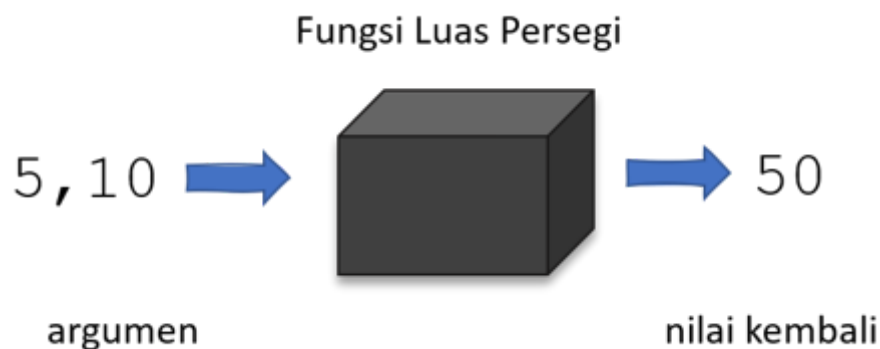
1.8 Fungsi Built-in

Fungsi adalah sekumpulan statement yang melakukan tugas tertentu. Python menyediakan berbagai fungsi yang sudah ditulis sebelumnya, disebut dengan fungsi built-in (fungsi bawaan).



Fungsi mempunyai nama, menerima **argumen**, dan memberikan **nilai kembali**

Sebuah fungsi dapat juga menerima lebih dari satu argumen



Untuk menjalankan suatu fungsi, kita **memanggil** fungsi tersebut dengan namanya

`<nama_fungsi>(<argumen>, <argumen>, ...)`

Argumen-argumen ditulis di dalam tanda kurung setelah nama fungsi dengan antar argumen dipisahkan koma

Fungsi Built-in: `abs()`

Salah satu fungsi built-in Python adalah fungsi `abs()` yang menerima sebuah argumen berupa tipe numerik dan mengembalikan nilai absolut (mutlak) dari argumen. Contoh pemanggilan fungsi `abs()`:

```
>>> abs(-9)
9
```

Kita dapat juga menuliskan argumen ke fungsi berupa ekspresi:

```
>>> suhu_siang = 38
>>> suhu_malam = 31
>>> abs(suhu_malam - suhu_siang)
7
```

Dalam menulis program, umumnya kita ingin memanfaatkan nilai kembali dari pemanggilan fungsi, sehingga kita menugaskan nilai kembali ke suatu variabel.

```
>>> beda_suhu = abs(suhu_malam - suhu_siang)
```

Nilai kembali dari pemanggilan fungsi umumnya ditugaskan ke sebuah variabel

Karena pemanggilan fungsi mengembalikan suatu nilai, kita dapat menggunakan pemanggilan fungsi dalam ekspresi.

```
>>> jarak = abs(-9) + abs(25)
>>> print(jarak)
34
```

Fungsi Built-in: `round()`

Fungsi built-in Python lainnya adalah fungsi `round()` yang membulatkan nilai floating point ke integer terdekat:

```
>>> round(3.8)
4
>>> round(3.3)
3
>>> round(3.5)
4
>>> round(-3.3)
-3
>>> round(-3.5)
-4
```

Fungsi `round()` dapat menerima dua argumen, dimana argumen kedua berupa integer yang menentukan presisi pembulatan:

```
>>> round(3.141592653, 2)
3.14
```

Argumen kedua bernilai 2 berarti pembulatan ke 2 angka setelah pemisah desimal

Fungsi Konversi Data

Fungsi `int()` digunakan untuk mengkonversi tipe data lain ke tipe data integer. Fungsi `float()` digunakan untuk mengkonversi tipe data lain ke tipe data floating point.

```
>>> num = int(34.6)
>>> print(num)
34
>>> desimal = float(21)
>>> print(desimal)
21.0
>>> lebar = int('20')
>>> print(lebar)
20
```

Dokumentasi Fungsi Built-in

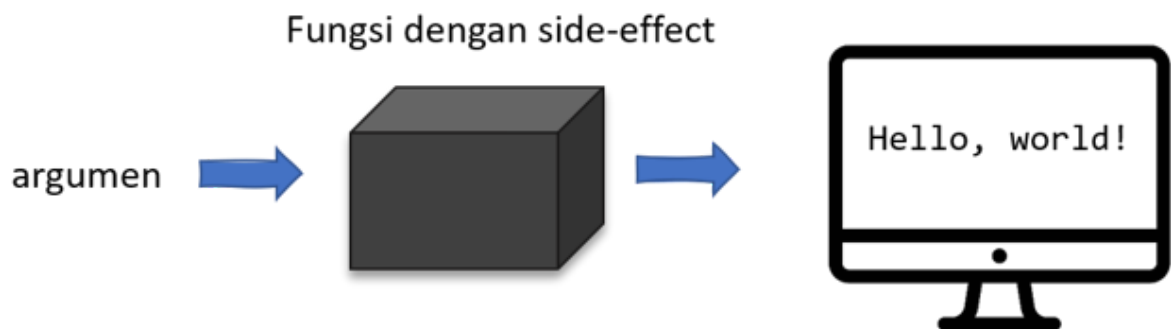
Kita dapat melihat argumen dan nilai kembali suatu fungsi built-in dengan melihat dokumentasinya menggunakan fungsi `help()`.

```
>>> help(abs)
Help on built-in function abs in module builtins:

abs(x, /)
    Return the absolute value of the argument.
```

Fungsi dengan Side-Effect

Aksi yang dilakukan oleh fungsi selain mengembalikan nilai disebut dengan **side-effect**.



Salah satu contoh fungsi dengan side-effect adalah fungsi `print()` yang menampilkan teks ke layar. Contoh lain adalah fungsi `input()` yang digunakan untuk membaca input dari keyboard

REFERENSI

[1] Gaddis, Tony. 2012. Starting Out With Python Second Edition. United States of America: Addison-Wesley.

