

Pertemuan 10

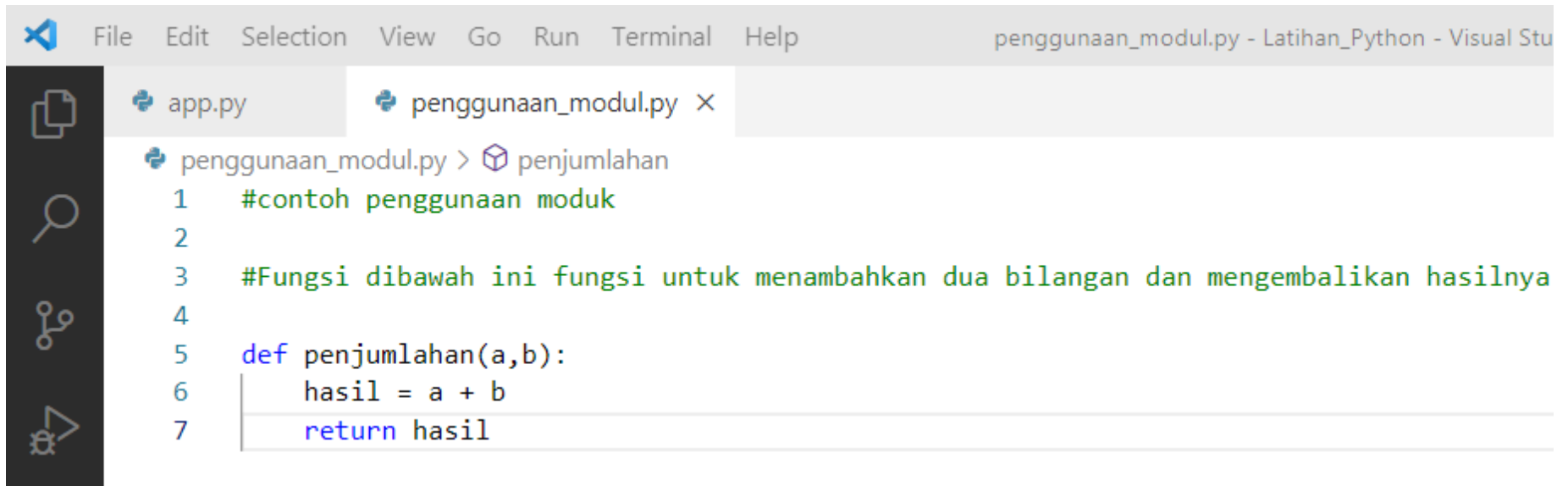
Modul & Eksepsi

Modul

Modul adalah sebuah file dengan ekstensi *.py*, yang berisikan kode program dalam bahasa pemrograman python. Nama modul sesuai dengan nama file yang digunakan. misalnya: *contoh_penggunaan_modul.py*, disebut modul dan nama modulnya adalah *contoh_penggunaan_modul*. Ada dua syntax untuk menggunakan sebuah modul, pertama menggunakan *import* dan kedua menggunakan *from*, bisa juga dengan menggunakan keduanya.

Python memiliki banyak modul bawaan, misalnya modul **math**, **os**, **sys** dan lain sebagainya. Modul – modul tersebut berada di dalam direktori Lib ditempat Python terinstall. Python juga memiliki ribuan modul siap pakai yang tersedia luas di internet, salah satunya di pypi.python.org. Modul digunakan untuk memecah sebuah program besar menjadi file – file yang lebih kecil agar lebih mudah dikelola. Modul membuat kode bersifat *reusable*, artinya satu modul bisa dipakai berulang dimana saja jika diperlukan. Berikut ini adalah contoh penggunaan modul.

Contoh Penggunaan Modul



```
File Edit Selection View Go Run Terminal Help    penggunaan_modul.py - Latihan_Python - Visual Stu
app.py  penggunaan_modul.py x
penggunaan_modul.py > penjumlahan
1  #contoh penggunaan moduk
2
3  #Fungsi dibawah ini fungsi untuk menambahkan dua bilangan dan mengembalikan hasilnya
4
5  def penjumlahan(a,b):
6      hasil = a + b
7      return hasil
```

Cara Lain untuk Mengimpor Modul

- Ada beberapa sintaks yang bisa digunakan untuk mengimpor modul, yaitu sebagai berikut:
- Cara import standar, formatnya **import** nama_modul
- Cara import dengan rename (alias), formatnya **import** nama_modul **as** alias
- Cara mengimport sebagian, formatnya **from...import** something
- Cara mengimport semua isi modul, formatnya **import** *

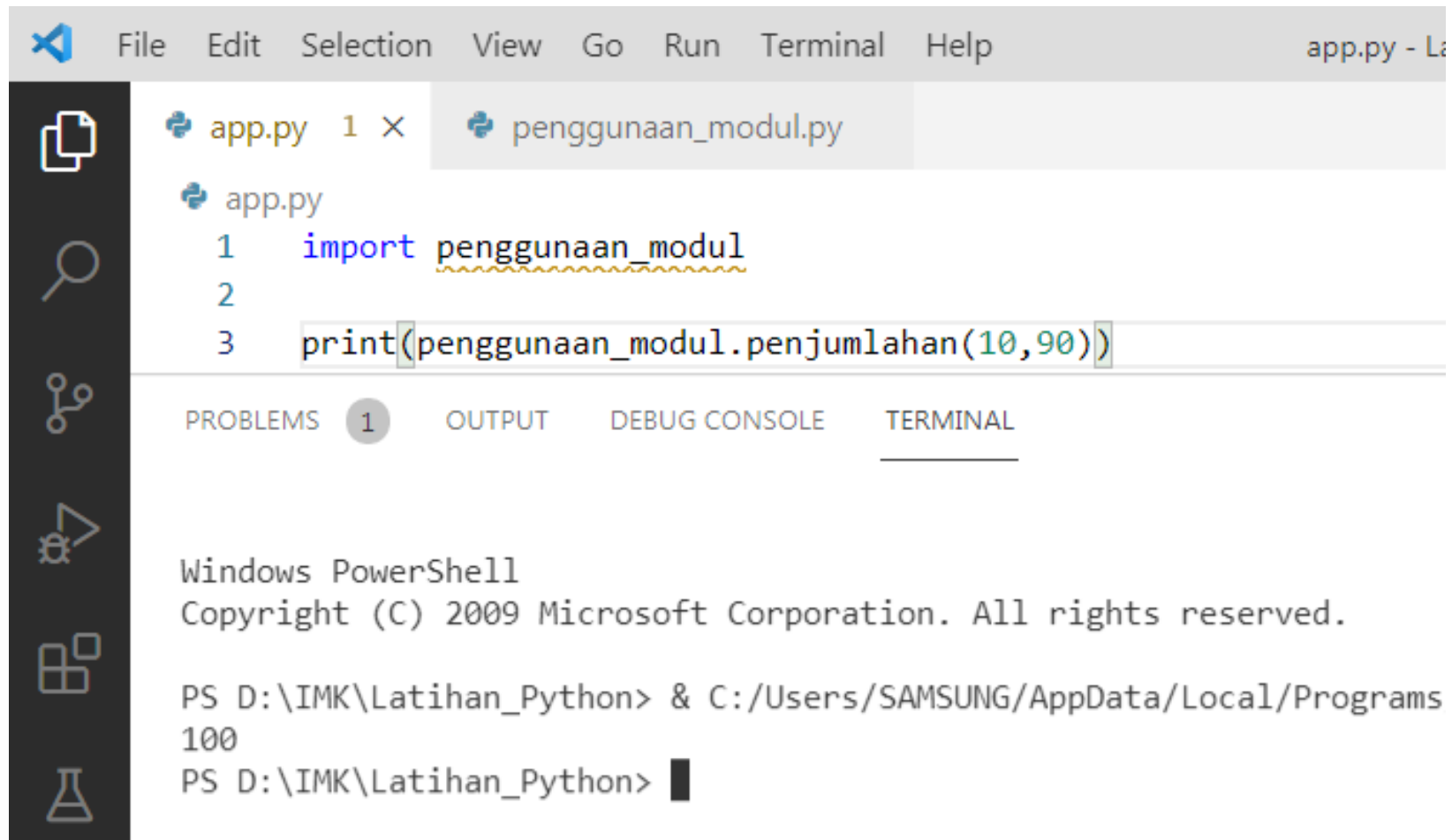
Menggunakan Perintah Import

- Kita bisa mengimpor modul python ke dalam program yang kita buat. Dengan mengimpor modul, maka definisi, variabel, fungsi dan yang lainnya yang ada di dalam modul itu bisa kita pergunakan kapan saja jika diperlukan. Kita mengimpor modul dengan menggunakan kata kunci **import**. Misalnya, kita akan mengimpor modul **contoh_penggunaan_modul** yang sudah kita buat di atas, maka kita bisa mengetikkan perintah berikut :

import penggunaan_modul

- Setelah kita import, maka kita bisa mengakses isi dari modul **penggunaan_modul**. Kita bisa mengakses fungsi maupun variabel global di dalam modul tersebut dengan menggunakan operasi titik (.). Misalnya sebagai berikut:

Contoh Menggunakan Perintah Import



```
File Edit Selection View Go Run Terminal Help app.py - La

app.py 1 x penggunaan_modul.py

app.py
1 import penggunaan_modul
2
3 print(penggunaan_modul.penjumlahan(10,90))

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

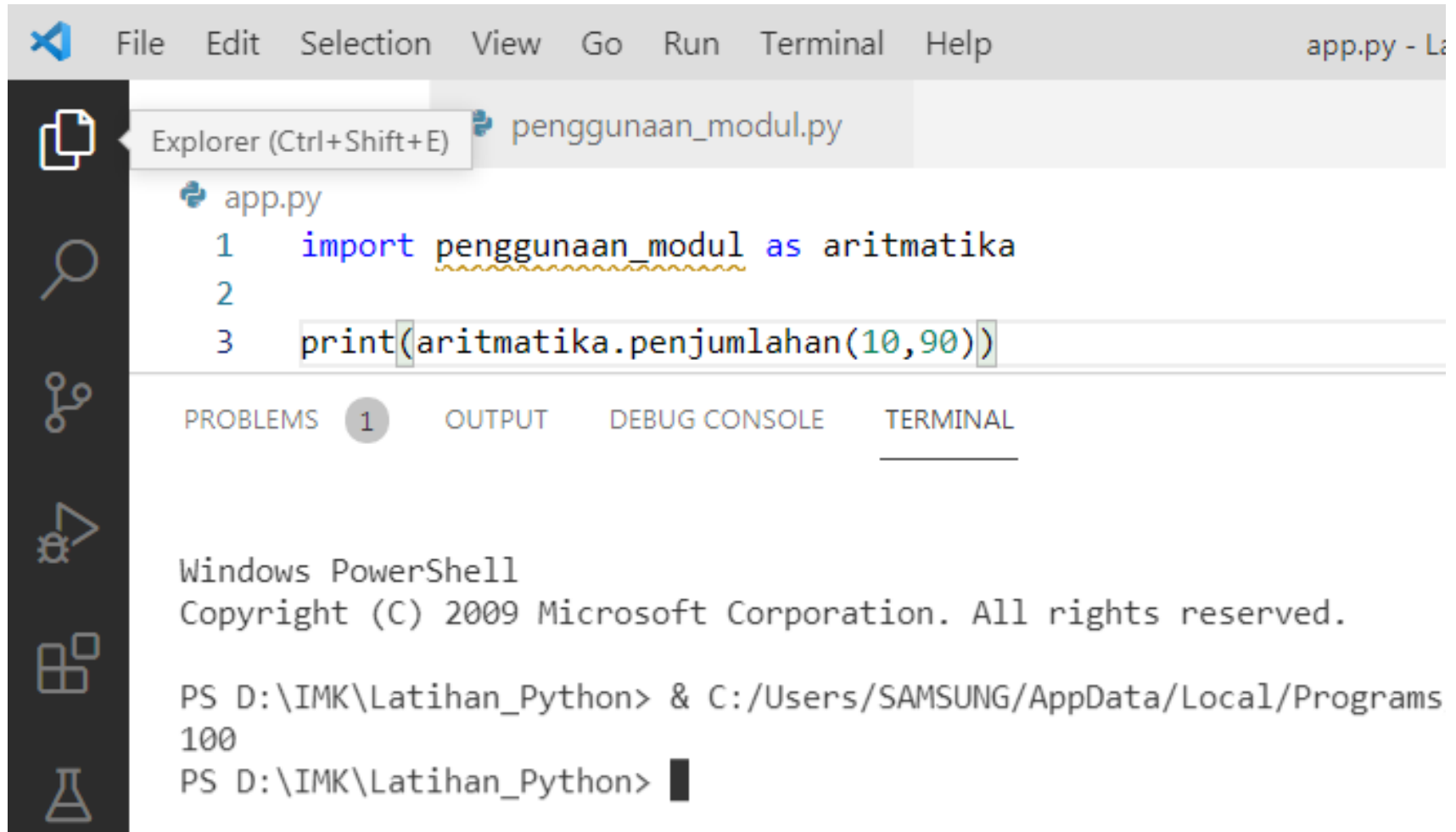
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS D:\IMK\Latihan_Python> & C:/Users/SAMSUNG/AppData/Local/Programs
100
PS D:\IMK\Latihan_Python> 
```

Menggunakan Perintah Alias pada Modul

Untuk memanggil suatu modul pada python, kita juga bisa memanggil modul tersebut dengan merubah namanya dengan perintah **as**, misalkan pada contoh dibawah ini :

Contoh Menggunakan Perintah Alias pada Modul



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a file named `app.py`. The main editor area shows the following Python code:

```
1 import penggunaan_modul as aritmatika
2
3 print(aritmatika.penjumlahan(10,90))
```

Below the editor, the TERMINAL tab is active, showing a Windows PowerShell prompt. The output of the command is 100.

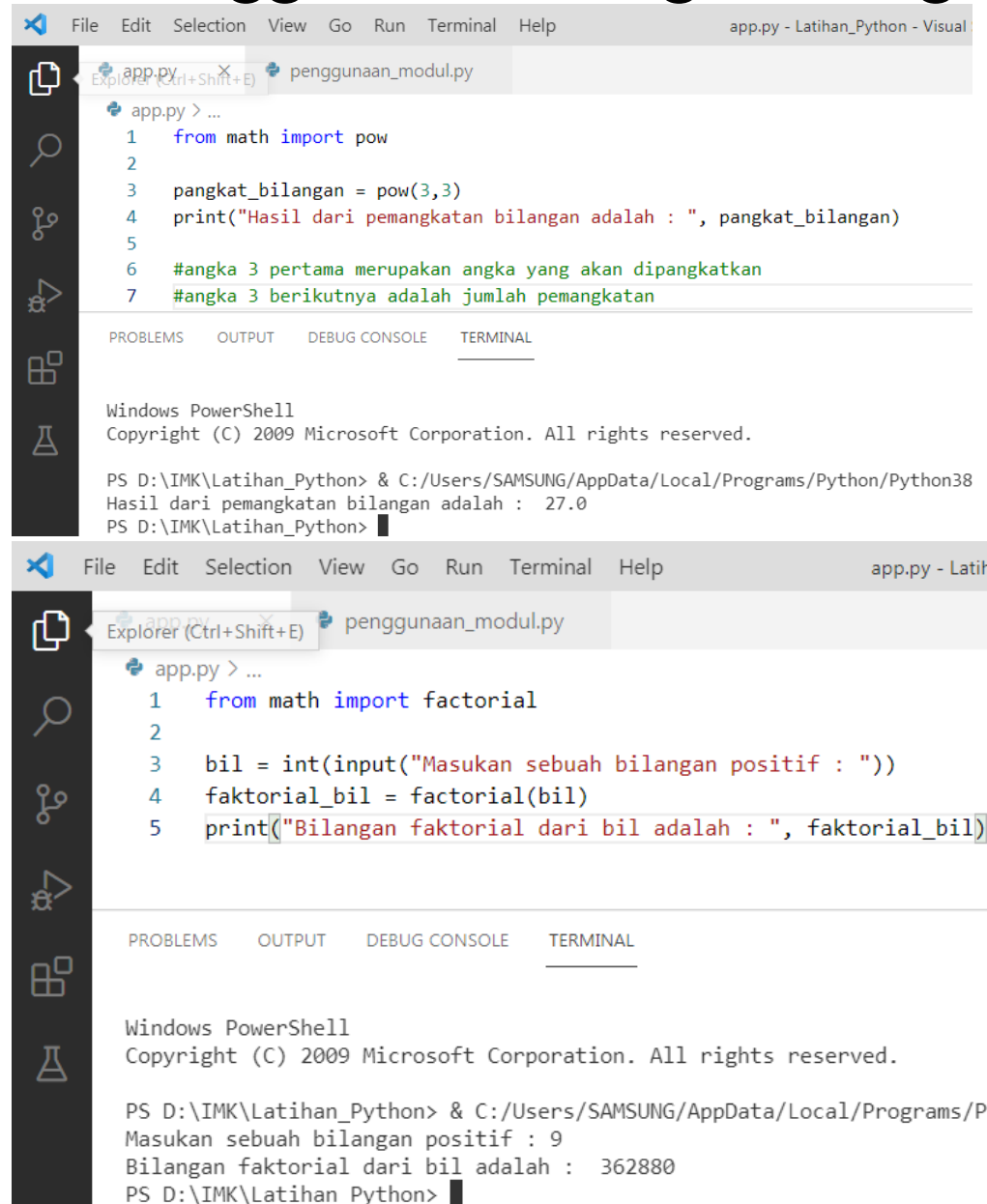
```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS D:\IMK\Latihan_Python> & C:/Users/SAMSUNG/AppData/Local/Programs
100
PS D:\IMK\Latihan_Python> 
```


Mengimpor sebagian fungsi modul pada Python

Pada python kita juga bisa menggunakan sebagian fungsi dari suatu modul dengan perintah `from... import...` seperti pada contoh dibawah ini :

Contoh Penggunaan Sebagian fungsi Modul



```
File Edit Selection View Go Run Terminal Help app.py - Latihan_Python - Visual
```

app.py > ...

```
1 from math import pow
2
3 pangkat_bilangan = pow(3,3)
4 print("Hasil dari pemangkatan bilangan adalah : ", pangkat_bilangan)
5
6 #angka 3 pertama merupakan angka yang akan dipangkatkan
7 #angka 3 berikutnya adalah jumlah pemangkatan
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS D:\IMK\Latihan_Python> & C:/Users/SAMSUNG/AppData/Local/Programs/Python/Python38
Hasil dari pemangkatan bilangan adalah : 27.0
PS D:\IMK\Latihan_Python>

```
File Edit Selection View Go Run Terminal Help app.py - Latif
```

app.py > ...

```
1 from math import factorial
2
3 bil = int(input("Masukan sebuah bilangan positif : "))
4 faktorial_bil = factorial(bil)
5 print("Bilangan faktorial dari bil adalah : ", faktorial_bil)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

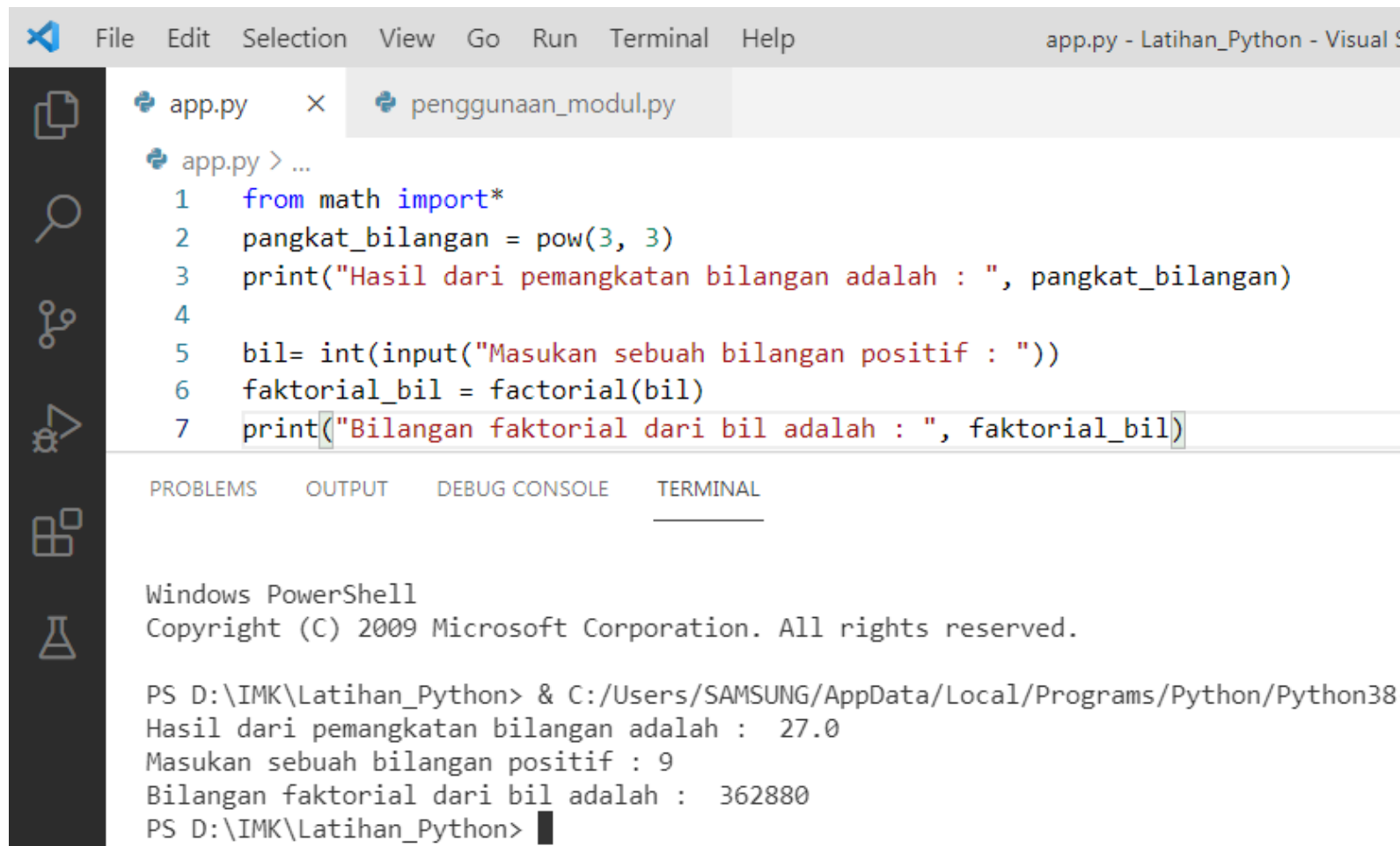
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS D:\IMK\Latihan_Python> & C:/Users/SAMSUNG/AppData/Local/Programs/P
Masukan sebuah bilangan positif : 9
Bilangan faktorial dari bil adalah : 362880
PS D:\IMK\Latihan_Python>

Mengimpor semua fungsi modul pada Python

Jika pada contoh sebelumnya menggunakan sebagian fungsi dari suatu modul pada python, maka contoh kali ini kita dapat menggunakan semua fungsi pada suatu modul, seperti pada contoh dibawah ini :

Contoh Penggunaan Semua Fungsi pada Suatu Modul



```
File Edit Selection View Go Run Terminal Help app.py - Latihan_Python - Visual S
app.py x penggunaan_modul.py
app.py > ...
1 from math import*
2 pangkat_bilangan = pow(3, 3)
3 print("Hasil dari pemangkatan bilangan adalah : ", pangkat_bilangan)
4
5 bil= int(input("Masukan sebuah bilangan positif : "))
6 faktorial_bil = factorial(bil)
7 print("Bilangan faktorial dari bil adalah : ", faktorial_bil)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS D:\IMK\Latihan_Python> & C:/Users/SAMSUNG/AppData/Local/Programs/Python/Python38
Hasil dari pemangkatan bilangan adalah : 27.0
Masukan sebuah bilangan positif : 9
Bilangan faktorial dari bil adalah : 362880
PS D:\IMK\Latihan_Python> 
```

Berikut ini merupakan modul-modul yang tersedia dalam python dan fungsinya :

Nama Modul	Fungsi
<code>__main__</code>	Tempat skrip level teratas dijalankan.
<code>array</code>	Array dengan nilai numerik.
<code>binascii</code>	Alat untuk mengonversi antara representasi biner berencode ASCII dan biner.
<code>bisect</code>	Algoritma pembagian dua untuk pencarian biner.
<code>calendar</code>	Fungsi untuk kalender, termasuk beberapa emulasi program Unix <code>cal</code> .
<code>cmath</code>	Fungsi matematika untuk bilangan kompleks.
<code>csv</code>	Menulis dan membaca data tabel dan dari file yang dibatasi.

Modul-modul yang tersedia dalam python dan fungsinya (lanjutan)

datetime	Fungsi tanggal dan waktu.
decimal	Penerapan Spesifikasi Aritmatika Desimal.
email	Paket yang mendukung parsing, manipulasi, dan pembuatan pesan email.
enum	Implementasi kelas pencacahan.
getpass	Pembacaan password portabel dan pengambilan userid.
html	Fungsi untuk memanipulasi HTML.
http	Kode dan pesan status HTTP
json	Encode dan decode format JSON.
math	Fungsi matematika (sin () dll.).
numbers	Kelas dasar abstrak numerik (Kompleks, Real, Integral, dll.).

Modul-modul yang tersedia dalam python dan fungsinya (lanjutan)

random	Hasilkan bilangan pseudo-random dengan berbagai distribusi umum.
statistics	Fungsi statistik matematika
sys	Akses parameter dan fungsi khusus sistem.
time	Akses dan konversi waktu.
warnings	Keluarkan pesan peringatan dan kendalikan disposisi mereka.
zipfile	Membaca dan menulis file arsip berformat ZIP.
zipimport	Dukungan untuk mengimpor modul Python dari arsip ZIP.

Eksepsi

Pada saat menulis dan menjalankan program, kita sering dihadapkan pada munculnya kesalahan atau error. Seringkali error menyebabkan program berhenti sendiri.

Error dapat terjadi akibat kesalahan struktur (sintaks) program. Hal ini disebut syntax error. Contohnya adalah seperti berikut:

```
>>> if x < 5   File "<stdin>", line 1
if x < 5
SyntaxError: invalid syntax
```

Kita bisa melihat bahwa penyebabnya adalah lupa titik dua pada pernyataan if. Error juga dapat terjadi pada saat runtime (saat program berjalan). Error seperti ini disebut eksepsi. Misalnya, bila kita membuka file yang tidak ada, maka akan muncul pesan kesalahan FileNotFoundError. Bila kita membagi bilangan dengan nol akan muncul ZeroDivisionError, dan lain sebagainya.

Pada saat terjadi eksepsi, Python akan menampilkan traceback dan detail dimana kesalahan terjadi.

```
>>> 1/0
```

```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in
```

```
<module>
```

```
ZeroDivisionError: division by zero
```

Eksepsi	Penyebab Error
AssertionError	Muncul pada saat pernyataan assert gagal
AttributeError	Muncul pada saat penugasan terhadap attribute atau referensi gagal
EOFError	Muncul saat fungsi input() mendapatkan kondisi akhir file (end-of-file)
FloatingPointError	Muncul saat operasi terhadap bilangan float gagal
GeneratorExit	Muncul saat metode close() generator dipanggil
ImportError	Muncul saat modul yang hendak diimpor tidak ditemukan
IndexError	Muncul saat indeks dari sequence berada di luar range
KeyError	Muncul saat suatu key tidak ditemukan di dalam dictionary
KeyboardInterrupt	Muncul saat user menekan tombol interupsi (Ctrl + C)
MemoryError	Muncul saat operasi kehabisan memori
NameError	Muncul saat variabel tidak ditemukan

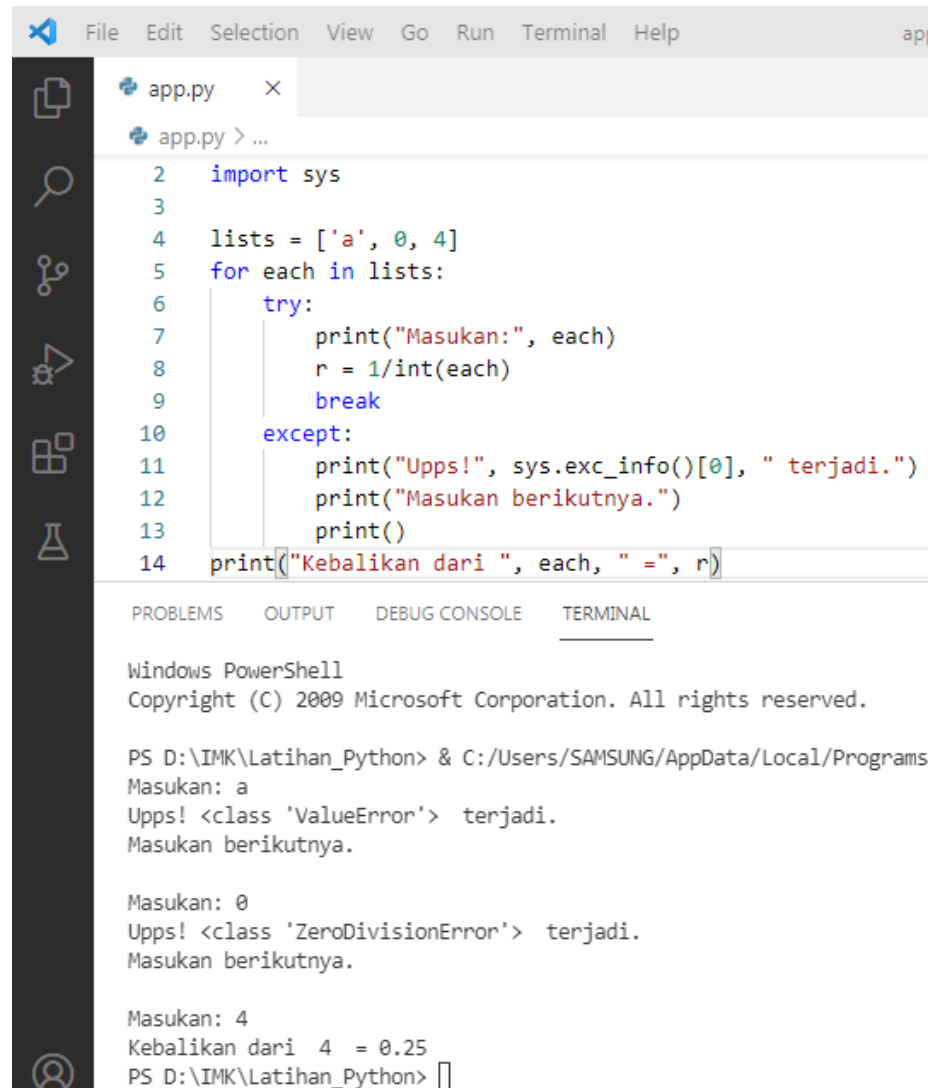
NotImplementedError	Muncul oleh metode abstrak
OSError	Muncul saat sistem operasi bersangkutan mengalami error
OverflowError	Muncul saat hasil operasi perhitungan terlalu besar untuk direpresentasikan
ReferenceError	Muncul saat <i>weak reference</i> digunakan untuk mengakses referensi sampah program
RuntimeError	Muncul saat error yang terjadi di luar semua kategori eksepsi lain
StopIteration	Muncul oleh fungsi <code>next()</code> untuk menunjukkan bahwa tidak ada lagi item yang tersisa pada iterator
SyntaxError	Muncul oleh parser saat terjadi kesalahan sintaks
IndentationError	Muncul saat ada indentasi yang salah
TabError	Muncul saat indentasi memiliki jumlah spasi atau tab yang tidak konsisten
SystemError	Muncul saat interpreter mendeteksi kesalahan internal
SystemExit	Muncul oleh fungsi <code>sys.exit()</code>

TypeError	Muncul saat melakukan operasi pada tipe data yang tidak sesuai
UnboundLocalError	Muncul saat referensi dibuat untuk variabel lokal dari fungsi, tapi tidak ada nilainya.
UnicodeError	Muncul saat terjadi kesalahan berkenaan dengan encoding dan decoding unicode
UnicodeEncodeError	Muncul saat terjadi kesalahan pada proses encoding
UnicodeDecodeError	Muncul saat terjadi kesalahan pada proses decoding
UnicodeTranslateError	Muncul saat terjadi kesalahan berkenaan dengan penerjemahan unicode
ValueError	Muncul saat fungsi menerima argumen yang tipe datanya salah
ZeroDivisionError	Muncul saat terjadi operasi pembagian bilangan dengan nol

Menangani Eksepsi Dengan Try, Except, dan Finally

Terjadinya eksepsi pada program dapat menyebabkan program terhenti. Untuk mencegah hal tersebut, kita harus mengantisipasi hal tersebut. Python menyediakan metode penanganan eksepsi dengan menggunakan pernyataan try dan except.

Di dalam blok try kita meletakkan baris program yang kemungkinan akan terjadi error. Bila terjadi error, maka penanganannya diserahkan kepada blok except. Berikut adalah contoh penanganan eksepsi pada operasi pembagian bilangan.



```
File Edit Selection View Go Run Terminal Help
app.py
app.py > ...
2 import sys
3
4 lists = ['a', 0, 4]
5 for each in lists:
6     try:
7         print("Masukan:", each)
8         r = 1/int(each)
9         break
10    except:
11        print("Upps!", sys.exc_info()[0], " terjadi.")
12        print("Masukan berikutnya.")
13        print()
14 print("Kebalikan dari ", each, " =", r)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS D:\IMK\Latihan_Python> & C:/Users/SAMSUNG/AppData/Local/Programs
Masukan: a
Upps! <class 'ValueError'> terjadi.
Masukan berikutnya.

Masukan: 0
Upps! <class 'ZeroDivisionError'> terjadi.
Masukan berikutnya.

Masukan: 4
Kebalikan dari 4 = 0.25
PS D:\IMK\Latihan_Python>

Pada program di atas kita mencari kebalikan dari bilangan, misalnya 4, maka kebalikannya adalah $1/4 = 0.25$.

Pembagian dengan huruf 'a', dan juga dengan 0 tidak bisa dilakukan, sehingga muncul error. Bila tidak dilakukan penanganan eksepsi, maka program akan langsung terhenti pada saat terjadi error.

Menangani Eksepsi Tertentu

Pada contoh di atas kita hanya menangani error secara umum. Tidak dikelompokkan, apakah dia adalah `TypeError`, `ValueError`, `SyntaxError`, dan lain sebagainya. Sebuah pernyataan `try`, bisa memiliki sejumlah pernyataan `except` untuk menangani jenis – jenis eksepsi secara lebih spesifik. Kita juga bisa mendefinisikan beberapa error sekaligus menggunakan tuple. Contohnya adalah seperti berikut:

```
try:
    # lakukan sesuatu pass
except ValueError:
    # tangani eksepsi ValueError pass

except (TypeError, ZeroDivisionError): #
    menangani multi eksepsi
    # TypeError dan ZeroDivisionError pass
except:

    # menangani eksepsi lainnya
    pass
```


Pernyataan `pass` adalah pernyataan yang tidak melakukan apa-apa. Istilahnya adalah statemen kosong. `pass` sering digunakan untuk mengisi blok fungsi atau kelas yang masih kosong.

Memunculkan Eksepsi

Eksepsi muncul bila terjadi error pada saat runtime atau saat program berjalan. Akan tetapi, kita juga bisa memunculkan eksepsi dengan sengaja untuk maksud tertentu dengan menggunakan kata kunci `raise`. Contohnya adalah seperti berikut:

```
>>> raise KeyboardInterrupt
Traceback (most recent call last):
...
KeyboardInterrupt

>>> try:
    a = int(input("Masukkan sebuah bilangan positif: "))
    if a <= 0:
        raise ValueError("Itu bukan bilangan positif!")
except ValueError as ve: print(ve)

Masukkan sebuah bilangan positif: -3  Itu
bukan bilangan positif!
```