

Pertemuan 7

String & Bilangan

String

String adalah tipe data yang paling sering digunakan di Python. Kita bisa membuat string dengan meletakkan karakter di dalam tanda kutip. Tanda kutipnya bisa kutip tunggal maupun kutip ganda. Contohnya adalah sebagai berikut:

```
var1 = 'Hello Python'
var2 = 'Programming with Python'
```

Mengakses Nilai String

Untuk mengakses nilai atau substring dari string, kita menggunakan indeks dalam tanda [].

```
var1 = 'Hello Python!'
var2 = "I love Python"
print("var1[0]", var1[0])
print("var2[2:6] :", var2[2:6])
```

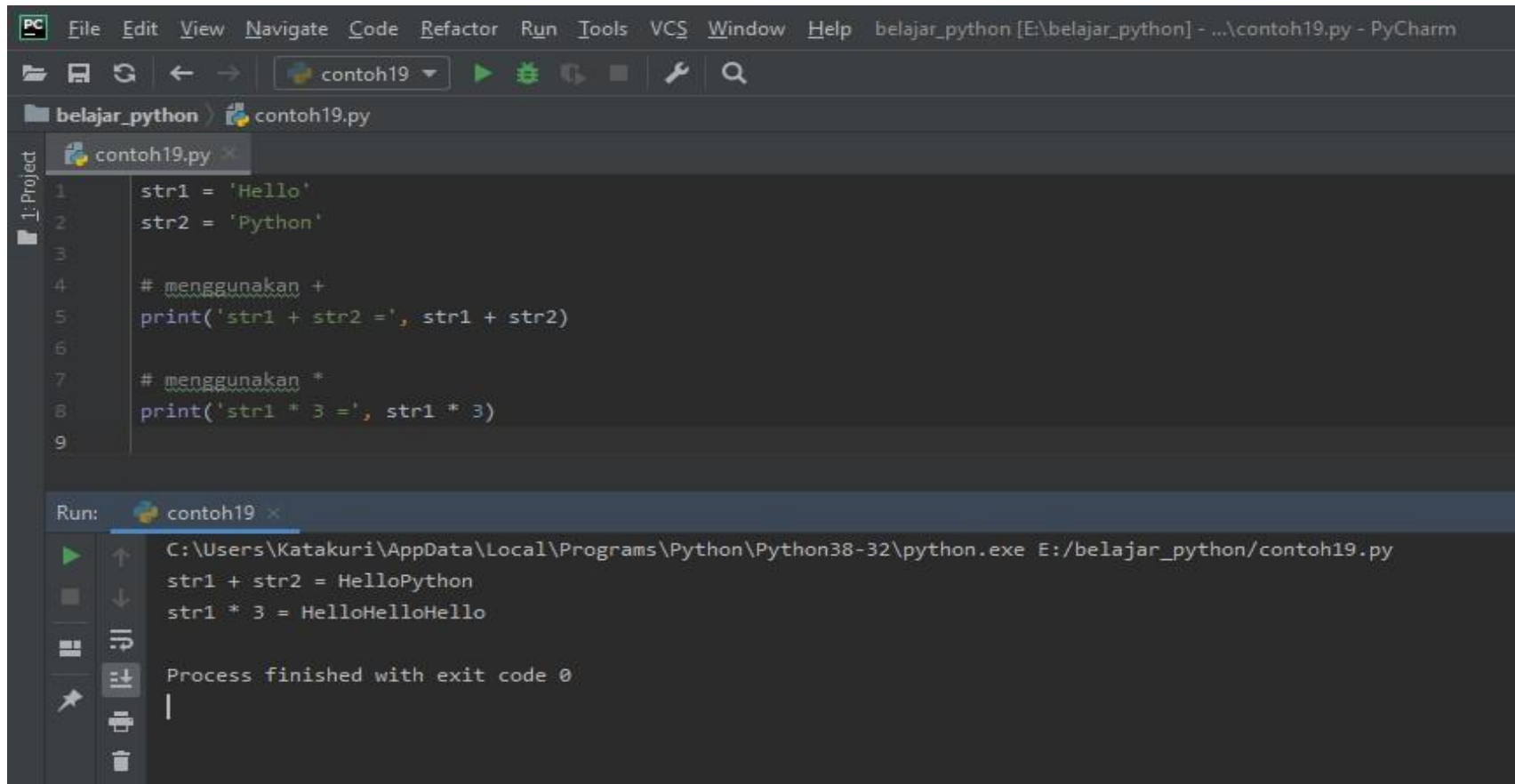
Mengupdate String

String adalah tipe data immutable, artinya tidak bisa diubah. Untuk mengupdate string, kita perlu memberikan nilai variabel string lama ke string yang baru. Nilai yang baru adalah nilai string lama yang sudah diupdate.

```
var1 = 'Hello Python!'  
var2 = var1[:6]  
print("String Update: - ", var1[:6] + 'World')
```

Menggabung String

Kita bisa menggabungkan dua atau lebih string menjadi satu dengan menggunakan operator `+`. Selain itu kita juga bisa melipatgandakan string menggunakan operator `*`.



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help belajar_python [E:\belajar_python] - ...\contoh19.py - PyCharm
contoh19
1 str1 = 'Hello'
2 str2 = 'Python'
3
4 # menggunakan +
5 print('str1 + str2 =', str1 + str2)
6
7 # menggunakan *
8 print('str1 * 3 =', str1 * 3)
9

Run: contoh19 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh19.py
str1 + str2 = HelloPython
str1 * 3 = HelloHelloHello
Process finished with exit code 0
```

Mengetahui Panjang String

Untuk mengetahui panjang dari string, kita bisa menggunakan fungsi `len()`.

```
>>> string = 'I love Python'
```

```
>>> len(string)
```

```
18
```

Karakter Escape

Kalau kita hendak mencetak string: He said, "What's there?" kita tidak bisa menggunakan tanda kutip tunggal maupun ganda. Bila kita melakukannya, akan muncul pesan error `SyntaxError` karena teks berisi kutip tunggal dan ganda.

```
>>> print("He said, "What's there?")
```

```
...
```

```
SyntaxError: invalid syntax
```

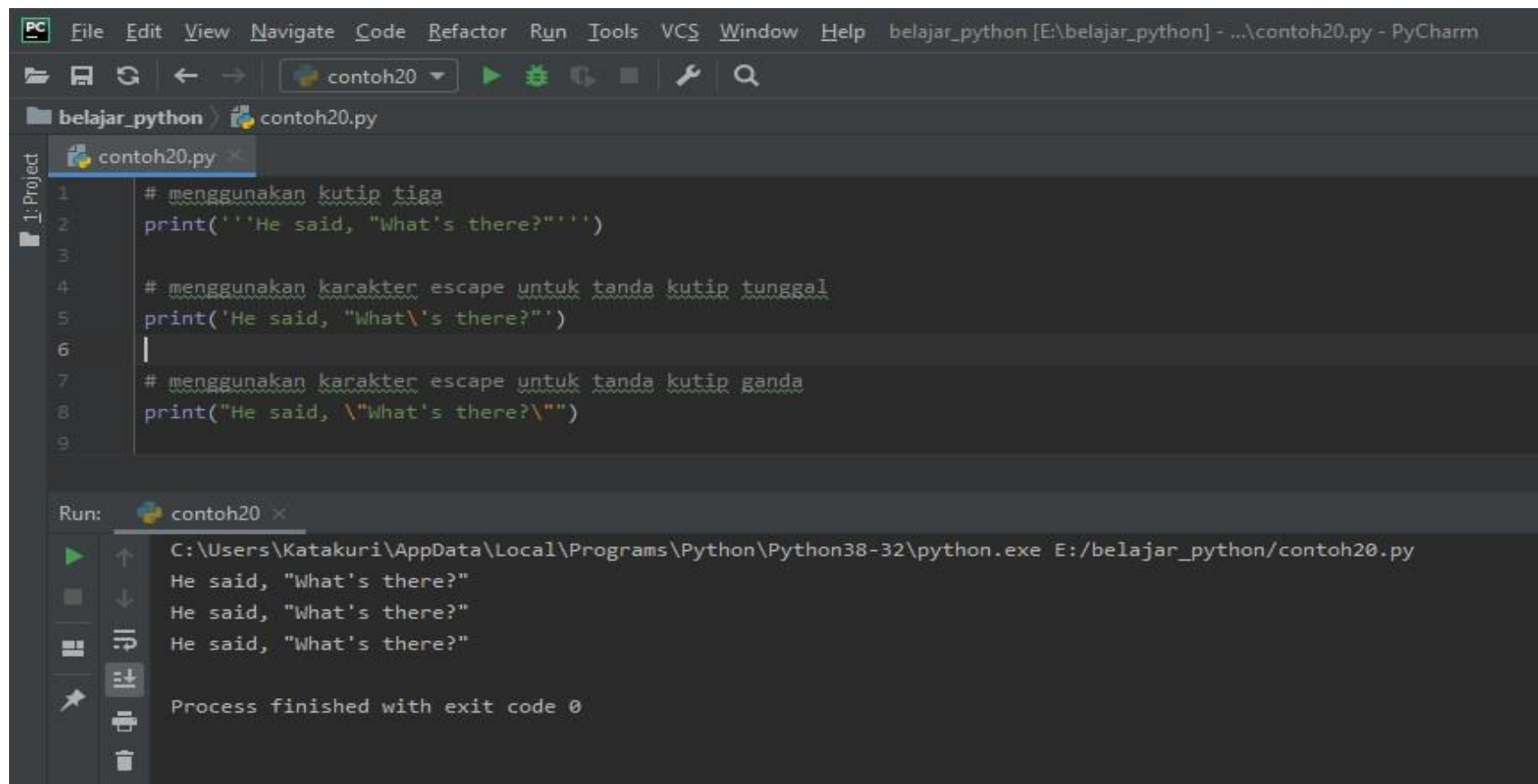
```
>>> print('He said, "What's there?")
```

```
...
```

```
SyntaxError: invalid syntax
```

Untuk hal seperti ini kita bisa menggunakan tanda kutip tiga atau menggunakan karakter escape.

Karakter escape dimulai dengan tanda backslash \. Interpreter akan menerjemahkannya dengan cara berbeda dengan string biasa. Solusi untuk error di atas adalah sebagai berikut:



```
1 # menggunakan kutip tiga
2 print('He said, "What's there?")')
3
4 # menggunakan karakter escape untuk tanda kutip tunggal
5 print('He said, "What\'s there?")')
6
7 # menggunakan karakter escape untuk tanda kutip ganda
8 print("He said, \"What's there?\"")
9
```

Run: contoh20

C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh20.py

He said, "What's there?"

He said, "What's there?"

He said, "What's there?"

Process finished with exit code 0

Berikut adalah daftar karakter escape yang didukung oleh Python.

Karakter Escape	Deskripsi
<code>\newline</code>	Backslash dan newline diabaikan
<code>\\</code>	Backslash
<code>\'</code>	Kutip tunggal
<code>\"</code>	Kutip ganda
<code>\a</code>	ASCII bel

<code>\b</code>	ASCII backspace
<code>\f</code>	ASCII formfeed
<code>\n</code>	ASCII linefeed
<code>\r</code>	ASCII carriage return
<code>\t</code>	ASCII tab horizontal
<code>\v</code>	ASCII tab horizontal
<code>\ooo</code>	karakter dengan nilai oktal oo
<code>\xHH</code>	karakter dengan nilai heksadesimal HH

Berikut ini adalah beberapa contohnya:

```
>>> print("C:\\Python34\\Lib")
```

C:\Python34\Lib

```
>>> print("Ini adalah baris pertama\nDan ini
```

baris dua") Ini adalah baris pertama

Dan ini baris dua

```
>>> print("Ini adalah \x48\x45\x58")
```

Ini adalah HEX

Raw String untuk Mengabaikan Karakter Escape

Kadang kala kita perlu untuk mengabaikan karakter escape yang ada dalam string. Kita bisa melakukannya dengan meletakkan huruf r atau R sebelum tanda kutip string.

```
>>> print("This is \x61 \ngood example")
```

```
This is a  good example
```

```
>>> print(r"This is \x61 \ngood example")  This
```

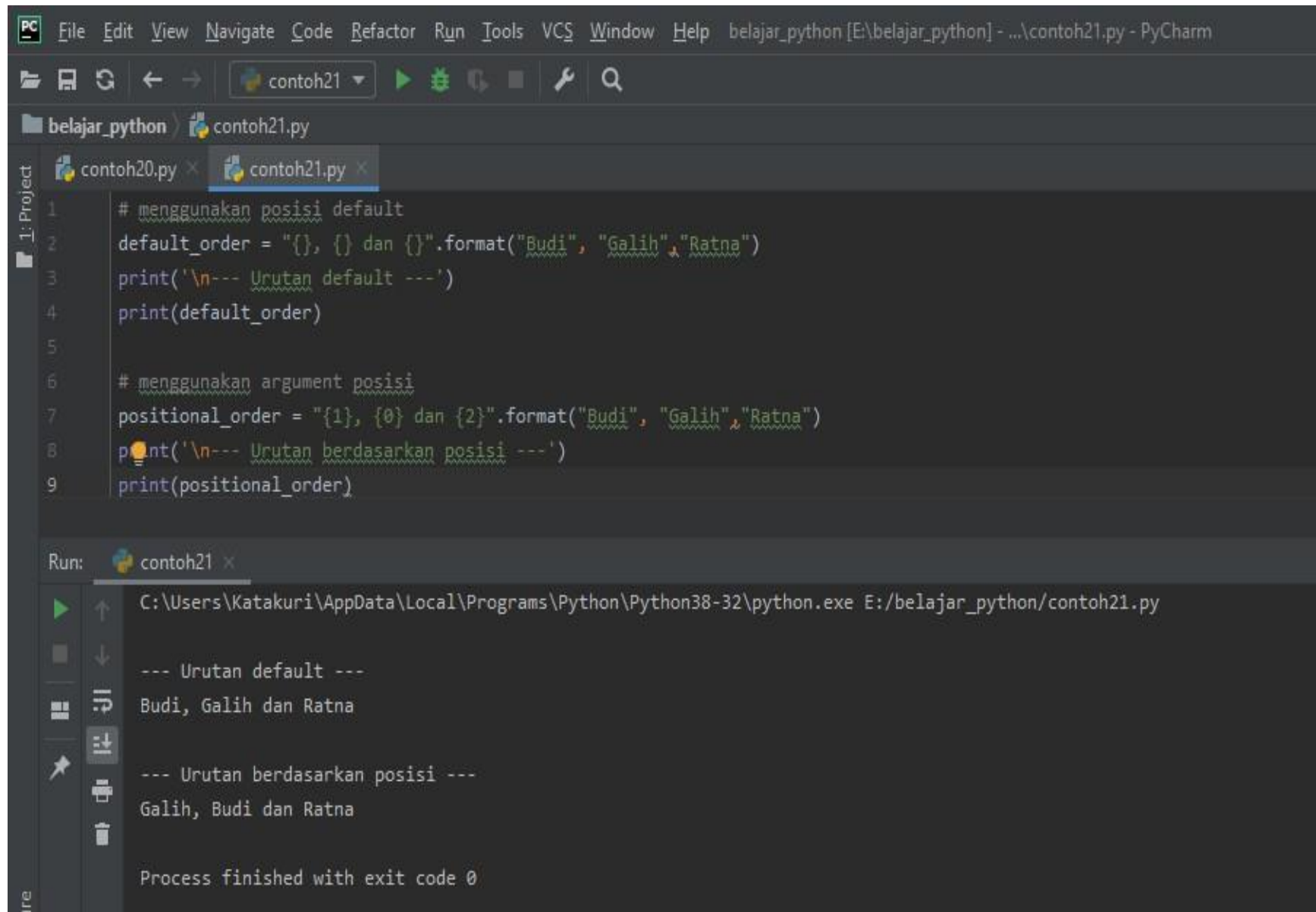
```
is \x61 \ngood example
```

Mengatur Format String

Ada dua cara melakukan format pada string. Pertama dengan menggunakan fungsi `format()`, dan kedua dengan menggunakan cara lama (menggunakan `%`).

Metode `format()`

Memformat string dengan fungsi `format()` dibuat dengan menggunakan tanda `{ }` sebagai placeholder atau posisi substring yang akan digantikan. Kita biasa menggunakan argumen posisi atau kata kunci untuk menunjukkan urutan dari substring.



The screenshot displays the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The toolbar shows icons for file operations and running code. The project explorer on the left shows the 'belajar_python' project with two files: 'contoh20.py' and 'contoh21.py'. The editor window shows the code for 'contoh21.py'.

```
1 # menggunakan posisi default
2 default_order = "{} , {} dan {}".format("Budi", "Galih", "Ratna")
3 print('\n--- Urutan default ---')
4 print(default_order)
5
6 # menggunakan argument posisi
7 positional_order = "{1}, {0} dan {2}".format("Budi", "Galih", "Ratna")
8 print('\n--- Urutan berdasarkan posisi ---')
9 print(positional_order)
```

The Run window at the bottom shows the execution of 'contoh21.py' using the Python 3.8.32 interpreter. The output is as follows:

```
--- Urutan default ---
Budi, Galih dan Ratna

--- Urutan berdasarkan posisi ---
Galih, Budi dan Ratna

Process finished with exit code 0
```

Metode `format()` dapat memiliki spesifikasi format opsional. Misalnya, kita bisa menggunakan tanda `<` untuk rata kiri, `>` untuk rata kanan, `^` untuk rata tengah, dan sebagainya.

```
>>> # format integer
```

```
>>> "{0} bila diubah jadi biner menjadi {0:b}".format(12)  '12 bila diubah jadi biner menjadi 1100'
```

```
>>> # format float
```

```
>>> "Format eksponensial: {0:e}".format(1566.345)  'Format eksponensial: 1566345e+03'
```

```
>>> # pembulatan
```

```
>>> "Sepertiga sama dengan: {0:.3f}".format(1/3)  'Sepertiga sama dengan: 0.333'
```

```
>>> # Meratakan string
>>> "|{:<10}|{: ^10}|{:>10}|".format('beras', 'gula', 'garam')
'|beras      |      gula      |      garam|'
```

```
>>> nama = 'Budi'
```

```
>>> print('Nama saya %s' %s)  Nama saya Budi
```

```
>>> x = 12.3456789
```

```
>>> print('Nilai x = %3.2f' %x)  Nilai x = 12.35
```

```
>>> print('Nilai x = %3.4f' %x)  Nilai x = 12.3456
```

Metode / Fungsi Bawaan String

String memiliki banyak fungsi bawaan. `format()` yang kita bahas di atas hanya salah satu darinya. Fungsi atau metode lainnya yang sering dipakai adalah `join()`, `lower()`, `upper()`, `split()`, `startswith()`, `endswith()`, `replace()` dan lain sebagainya.

```
>>> "Universitas Bina Sarana Informatika".lower()
```

```
'universitas bina sarana informatika'
```

```
>>> " Universitas Bina Sarana Informatika ".upper()
```

```
'UNIVERSITAS BINA SARANA INFORMATIKA'
```



```
>>> "I love programming in Python".split()
['I', 'love', 'programming', 'in', 'Python']

>>> "I love Python".startswith("I")  True

>>> "Saya belajar Python".endswith("on")  True

>>> ' - '.join(['I', 'love', 'you'])  I - love - you

>>> "Belajar Java di BSI".replace('Java', 'Python')  'Belajar Python di BSI'
```

Bilangan (Number)

Bilangan (number) adalah salah satu tipe data dasar di Python. Python mendukung bilangan bulat (integer), bilangan pecahan (float), dan bilangan kompleks (complex). Masing – masing diwakili oleh kelas int, float, dan complex. Integer adalah bilangan bulat, yaitu bilangan yang tidak mempunyai koma. Contohnya 1, 2, 100, -30, -5, 99999, dan lain sebagainya. Panjang integer di python tidak dibatasi jumlah digitnya. Selama memori masih cukup, maka sepanjang itulah jumlah digit yang akan ditampilkan.

Float adalah bilangan pecahan atau bilangan berkoma. Contohnya adalah 5.5, 3.9, 72.8, -1.5, -0.7878999, dan lain sebagainya. Panjang angka di belakang koma untuk float ini adalah 15 digit. Bilangan kompleks (complex) adalah bilangan yang terdiri dari dua bagian, yaitu bagian yang real dan bagian yang imajiner. Contohnya adalah $3 + 2j$, $9 - 5j$, dan lain sebagainya.

Konversi Jenis Bilangan

Kita bisa mengubah jenis bilangan dari int ke float, atau sebaliknya. Mengubah bilangan integer ke float bisa menggunakan fungsi `int(num)` dimana `num` adalah bilangan float.

```
>>> int(2.5)
```

```
2
```

```
>>> int(3.8)    3
```

```
>>> float(5)    5.0
```

Pada saat kita mengubah float ke integer, bilangan dibulatkan ke bawah. Sebaliknya saat kita mengubah integer ke float, maka bilangan bulat akan menjadi bilangan berkoma.

Python Decimal

Ada kalanya perhitungan menggunakan float di Python membuat kita terkejut. Kita tahu bahwa 1.1 ditambah 2.2 hasilnya adalah 3.3. Tapi pada saat kita lakukan dengan Python, maka hasilnya berbeda.

```
>>> (1.1 + 2.2 ) == 3.3
```

```
False
```

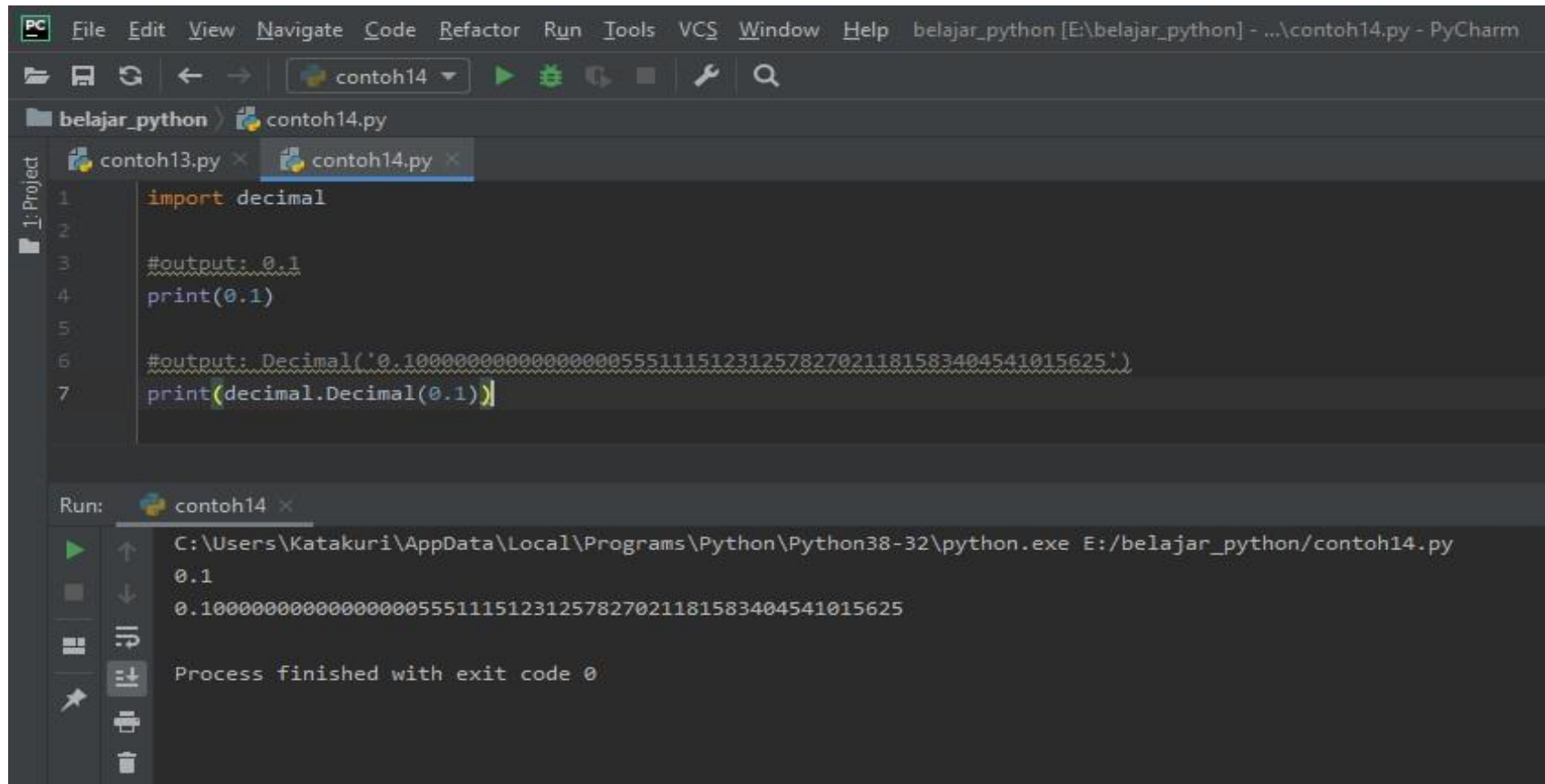
```
>>> 1.1 + 2.2
```

```
3.3000000000000003
```

Mengapa terjadi demikian?

Hal ini terjadi karena bilangan dalam komputer disimpan dalam bentuk digit 0 atau 1. Bila padanan digitnya tidak sesuai, maka bilangan float seperti 0.1 dalam bilangan biner akan menjadi pecahan yang sangat panjang yaitu 0.000110011001100110011... dan komputer kita hanya akan menyimpan panjang yang terbatas. Hal inilah yang menyebabkan terjadinya masalah seperti pada contoh di atas.

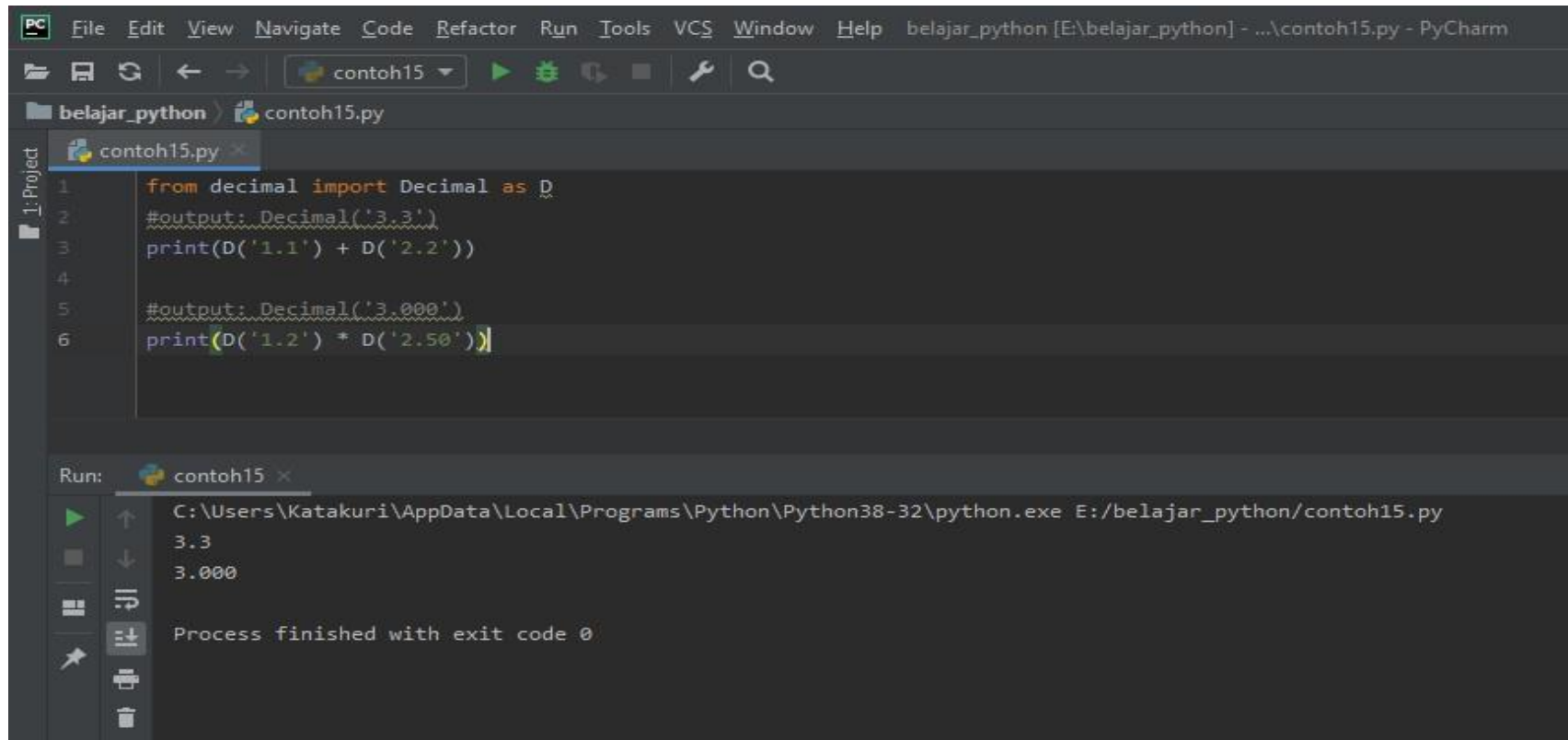
Untuk menangani hal seperti itu, kita bisa menggunakan modul bawaan Python yaitu modul decimal. Float hanya memiliki presisi sampai 15 digit di belakang koma, sementara dengan modul decimal kita bisa mengatur presisi jumlah digit di belakang koma.



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help belajar_python [E:\belajar_python] - ...\contoh14.py - PyCharm
contoh14
1 import decimal
2
3 #output: 0.1
4 print(0.1)
5
6 #output: Decimal('0.100000000000000055511151231257827021181583404541015625')
7 print(decimal.Decimal(0.1))

Run: contoh14 x
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh14.py
0.1
0.100000000000000055511151231257827021181583404541015625
Process finished with exit code 0
```

Modul ini juga membuat kita bisa melakukan perhitungan seperti di sekolah.



```
PC File Edit View Navigate Code Refactor Run Tools VCS Window Help belajar_python [E:\belajar_python] - ...\contoh15.py - PyCharm
contoh15
belajar_python
contoh15.py
1 from decimal import Decimal as D
2 #output: Decimal('3.3')
3 print(D('1.1') + D('2.2'))
4
5 #output: Decimal('3.000')
6 print(D('1.2') * D('2.50'))

Run: contoh15
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh15.py
3.3
3.000
Process finished with exit code 0
```

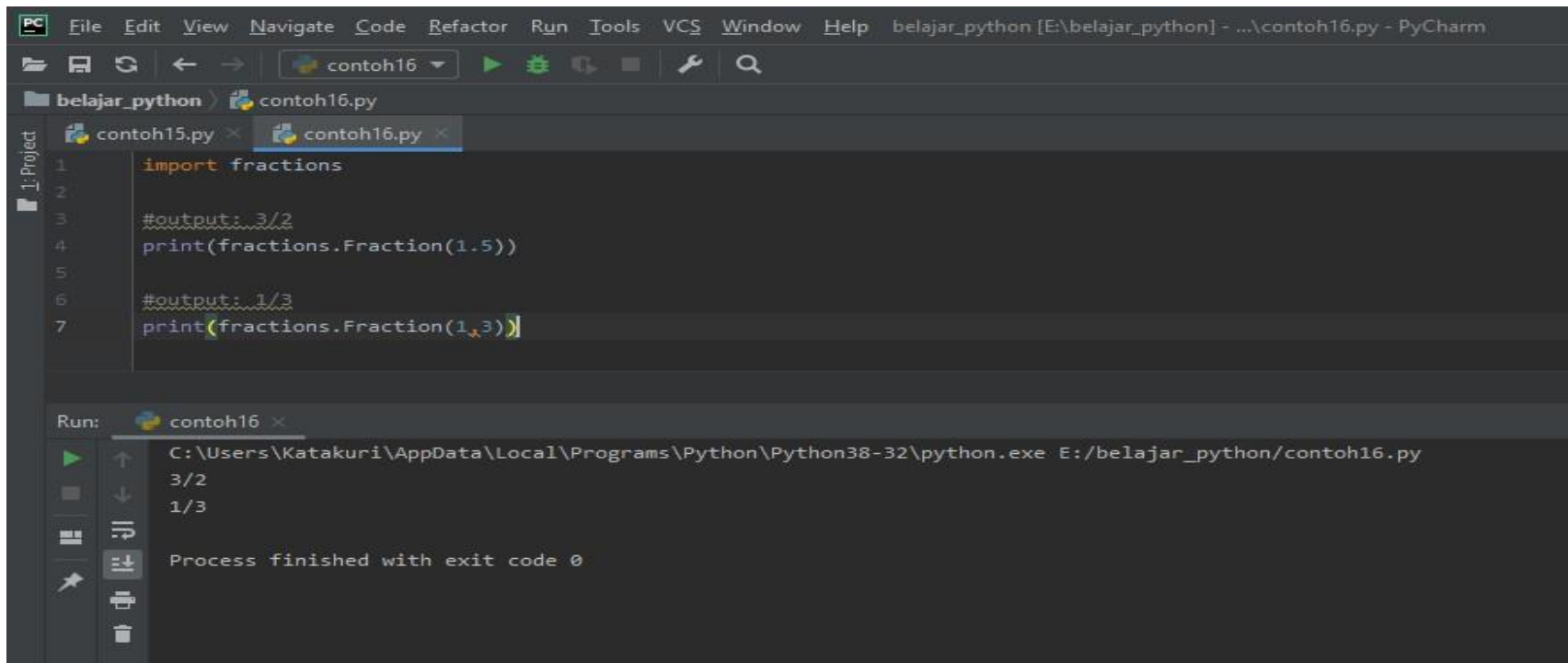
Kapan Saatnya Menggunakan Decimal Dibanding Float?

Kita lebih baik menggunakan Decimal dalam kasus:

- Saat kita ingin membuat aplikasi keuangan yang membutuhkan presisi desimal yang pasti
- Saat kita ingin mengontrol tingkat presisi yang diperlukan
- Saat kita ingin menerapkan perkiraan berapa digit decimal yang signifikan
- Saat kita ingin melakukan operasi perhitungan sama persis dengan yang kita lakukan di sekolah

Bilangan Pecahan

Python menyediakan modul fractions untuk mengoperasikan bilangan pecahan. Pecahan adalah bilangan yang memiliki pembilang dan penyebut, misalnya $3/2$. Perhatikan contoh berikut:



```
1 import fractions
2
3 #output: 3/2
4 print(fractions.Fraction(1.5))
5
6 #output: 1/3
7 print(fractions.Fraction(1,3))
```

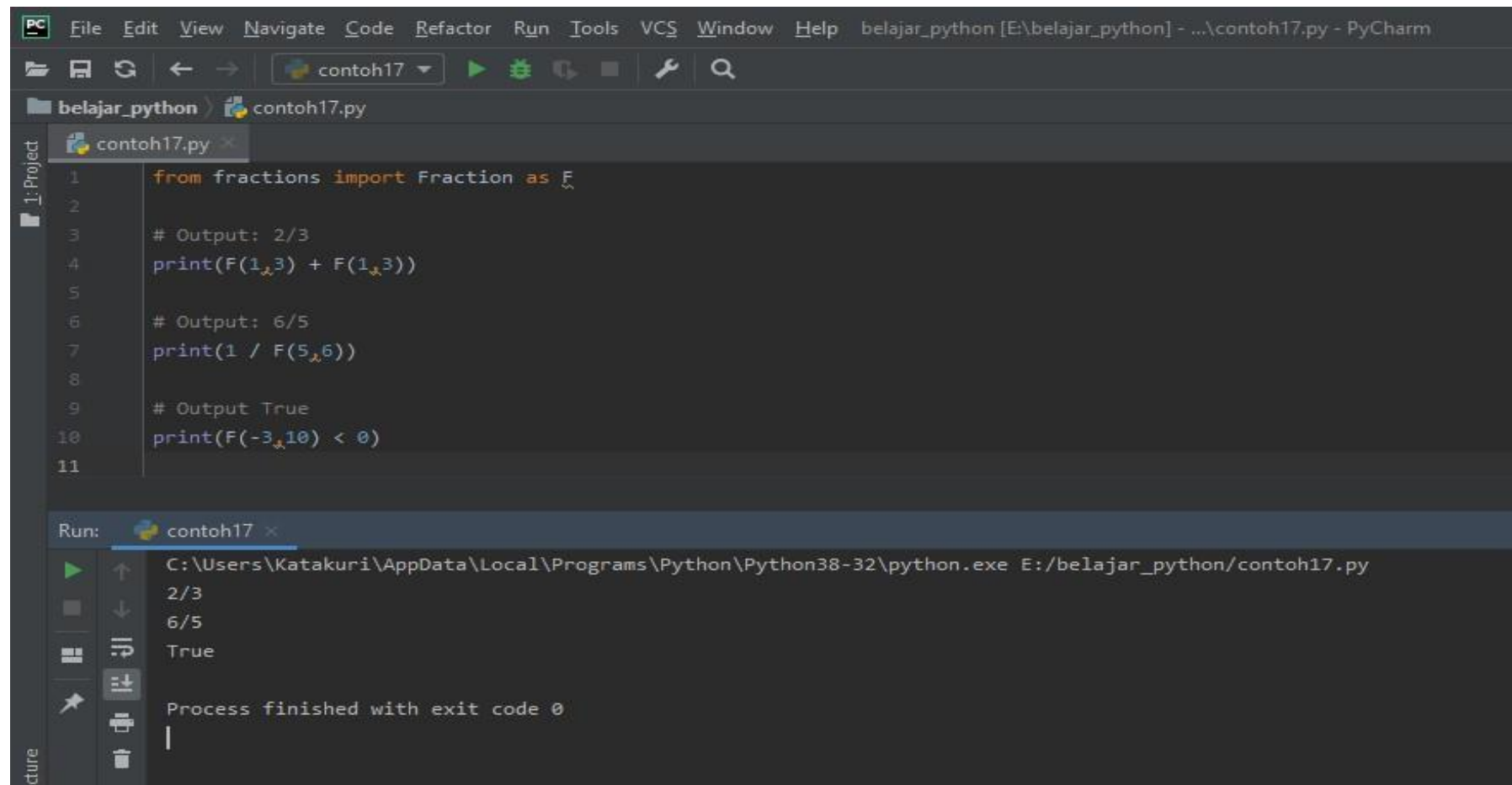
Run: contoh16 x

C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh16.py

3/2
1/3

Process finished with exit code 0

Operasi dasar seperti penjumlahan atau pembagian pecahan juga bisa dilakukan dengan modul fractions ini



```
1 from fractions import Fraction as F
2
3 # Output: 2/3
4 print(F(1,3) + F(1,3))
5
6 # Output: 6/5
7 print(1 / F(5,6))
8
9 # Output True
10 print(F(-3,10) < 0)
11
```

Run: contoh17 x

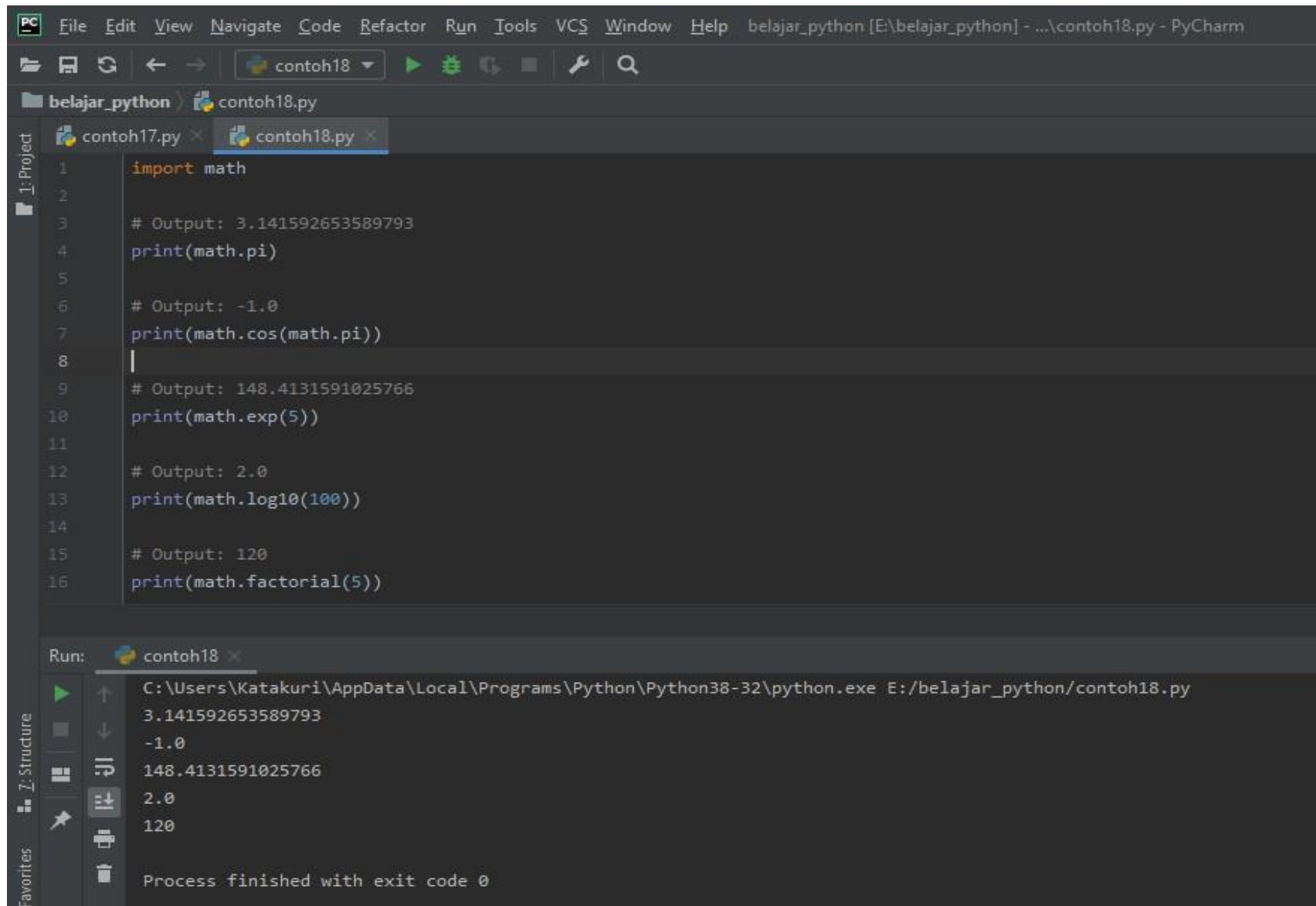
C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh17.py

2/3
6/5
True

Process finished with exit code 0

Matematika dengan Python

Python menyediakan modul math melakukan hal yang berbau matematis seperti trigonometri, logaritma, probabilitas, statistik, dan lain – lain.



The screenshot displays the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The title bar shows the project name 'belajar_python' and the file 'contoh18.py'. The left sidebar shows the project structure with '1: Project' and 'I: Structure' views. The main editor window shows the code for 'contoh18.py' with line numbers 1 through 16. The code imports the 'math' module and prints several mathematical constants and functions. The bottom panel shows the 'Run' output for 'contoh18', displaying the execution path, the output values, and the exit code.

```
1 import math
2
3 # Output: 3.141592653589793
4 print(math.pi)
5
6 # Output: -1.0
7 print(math.cos(math.pi))
8
9 # Output: 148.4131591025766
10 print(math.exp(5))
11
12 # Output: 2.0
13 print(math.log10(100))
14
15 # Output: 120
16 print(math.factorial(5))
```

Run: contoh18 x

C:\Users\Katakuri\AppData\Local\Programs\Python\Python38-32\python.exe E:/belajar_python/contoh18.py

3.141592653589793

-1.0

148.4131591025766

2.0

120

Process finished with exit code 0