Pemrograman Dasar
PYTHON

Apa itu Python?
Python adalah bahasa pemrograman yang populer.
Python dapat digunakan di server untuk membuat aplikasi web.
Python diciptakan oleh Guido van Rossum, dan dirilis pada tahun 1991.
Python digunakan untuk: pengembangan web (sisi server), pengembangan perangkat lunak, matematika, skrip sistem.

Apa yang dapat dilakukan Python?
- *Python dapat digunakan di server untuk membuat aplikasi web.*
- *Python dapat digunakan bersama-sama dengan perangkat lunak lain untuk membuat alur kerja.*
- *Python dapat terhubung ke sistem basis data.*
- *Python juga dapat membaca dan memodifikasi file.*
- *Python dapat digunakan untuk menangani data besar dan melakukan matematika kompleks.*
- *Python dapat digunakan untuk prototyping cepat, atau untuk pengembangan perangkat lunak siap produksi.*

1.Python Output / Print

Print Text :
Anda dapat menggunakan fungsi print() untuk menampilkan teks atau nilai output:

Praktik1
```
print("Hello World!")
print("I am learning Python.")
print("It is awesome!")
```

Teks dalam Python harus berada di dalam tanda kutip. Anda dapat menggunakan tanda kutip ganda " atau tanda kutip tunggal '

Praktik2
```
print("This will work!")
print('This will also work!')
```

Print Numbers

Praktik3
```
print(3)
print(358)
print(3 + 3)
print(2 * 5)
```

Mix Text and Numbers

Praktik4
```
print("I am", 35, "years old.")
```

Python Comments

Membuat Komentar
Komentar dimulai dengan tanda #, dan Python akan mengabaikannya.
Praktik5
#This is a comment
print("Hello, World!")

Komentar dapat ditempatkan di akhir sebuah baris, dan Python akan mengabaikan sisa baris tersebut.
Praktik6
print("Hello, World!") #This is a comment

1.1. Python User Input

User Input

Python memungkinkan adanya input dari pengguna.
Artinya, kita dapat meminta pengguna untuk memasukkan data.
Contoh berikut meminta nama Anda, dan ketika Anda memasukkan nama, nama tersebut akan ditampilkan di layar:

Example

Meminta input dari pengguna:
Praktik6a
```
print("Enter your name:")
name = input()
print(f"Hello {name}")
```

Python akan berhenti mengeksekusi program ketika mencapai fungsi input(), dan akan melanjutkan kembali setelah pengguna memberikan input.

Using prompt

Pada contoh di atas, pengguna harus memasukkan nama mereka pada baris baru. Fungsi input() pada Python memiliki parameter *prompt*, yang berfungsi sebagai pesan yang dapat ditampilkan sebelum input pengguna, pada baris yang sama:

Multiple Inputs

Anda dapat menambahkan input sebanyak yang Anda inginkan, Python akan berhenti mengeksekusi pada setiap input tersebut dan menunggu masukan dari pengguna.

Example

Multiple inputs:

Praktik6b

```
name = input("Enter your name:")
print(f"Hello {name}")
fav1 = input("What is your favorite animal:")
fav2 = input("What is your favorite color:")
fav3 = input("What is your favorite number:")
print(f"Do you want a {fav2} {fav1} with {fav3} legs?")
```

Input Number

Input dari pengguna diperlakukan sebagai sebuah string. Meskipun pada contoh di atas Anda dapat memasukkan angka, interpreter Python tetap akan memperlakukannya sebagai string.

Anda dapat mengonversi input tersebut menjadi angka dengan fungsi `float()`:

Example

Input dari pengguna akan diperlakukan sebagai sebuah string.
Meskipun pada contoh di atas Anda dapat memasukkan angka, interpreter Python tetap akan memperlakukannya sebagai string.

Praktik6c

```
x = input("Enter a number:")

#find the square root of the number:
y = math.sqrt(float(x))

print(f"The square root of {x} is {y}")
```

*Untuk mencari akar kuadrat, input harus dikonversi menjadi sebuah angka:

Validate Input

Merupakan praktik yang baik untuk memvalidasi setiap input dari pengguna.
Pada contoh di atas, akan terjadi kesalahan jika pengguna memasukkan sesuatu selain angka.

Untuk menghindari kesalahan, kita dapat memeriksa input tersebut. Jika input bukan angka, pengguna dapat menerima pesan seperti "Input salah, silakan coba lagi", dan diizinkan untuk memasukkan input baru.

Example

Terus minta input sampai Anda mendapatkan sebuah angka:

Praktik6d

```
y = True
while y == True:
  x = input("Enter a number:")
  try:
    x = float(x);
    y = False
  except:
    print("Wrong input, please try again.")
```

```
print("Thank you!")
```

2.Python Variables

Variabel adalah wadah untuk menyimpan nilai data.

Creating Variables

Python tidak memiliki perintah khusus untuk mendeklarasikan variabel.
Sebuah variabel akan dibuat pada saat Anda pertama kali memberikan nilai kepadanya.

Praktik7

x = 5
y = "John"
print(x)
print(y)

Variabel tidak perlu dideklarasikan dengan tipe tertentu, dan bahkan dapat mengubah tipe setelah diberi nilai.

Praktik8

```
x = 4       # x is of type int
x = "Sally" # x is now of type str
print(x)
```

Casting

Jika Anda ingin menentukan tipe data dari sebuah variabel, hal tersebut dapat dilakukan dengan casting.

Praktik9

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

Get the Type

Anda dapat mengetahui tipe data dari sebuah variabel dengan menggunakan fungsi type().

Praktik10

x = 5
y = "John"
print(type(x))
print(type(y))

Single or Double Quotes?

Variabel string dapat dideklarasikan dengan menggunakan tanda kutip tunggal maupun tanda kutip ganda.

Praktik11

x = "John"
# is the same as
x = 'John'

Case-Sensitive

Nama variabel bersifat peka huruf besar-kecil (case-sensitive).

Praktik12

```
a = 4
A = "Sally"
print(a)
Print(A)
```

2a.Python - Variable Names

Variable Names

Sebuah variabel dapat memiliki nama yang pendek (seperti x dan y) atau nama yang lebih deskriptif (seperti age, carname, total_volume).

Aturan untuk variabel Python:

1. Nama variabel harus dimulai dengan huruf atau karakter garis bawah (_)
2. Nama variabel tidak boleh dimulai dengan angka
3. Nama variabel hanya boleh berisi karakter alfanumerik dan garis bawah (A–Z, a–z, 0–9, dan _)
4. Nama variabel bersifat case-sensitive (contoh: age, Age, dan AGE adalah tiga variabel yang berbeda)
5. Nama variabel tidak boleh menggunakan kata kunci (keywords) Python.

Example

Nama variabel yang diperbolehkan:

Praktik13

```
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"
```

Example

Nama variabel yang tidak diperbolehkan:

Praktik14

```
2myvar = "John"
my-var = "John"
my var = "John"
```

2b.Python Variables - Assign Multiple Values

Many Values to Multiple Variables

Python memungkinkan Anda untuk memberikan nilai kepada beberapa variabel dalam satu baris:

Praktik15

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

One Value to Multiple Variables

Dan Anda dapat memberikan nilai yang sama kepada beberapa variabel dalam satu baris:

Praktik 16

```
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

Unpack a Collection

Jika Anda memiliki sekumpulan nilai dalam sebuah list, tuple, dan sebagainya, Python memungkinkan Anda untuk mengekstrak nilai-nilai tersebut ke dalam variabel. Proses ini disebut unpacking.

Praktik 16 (Unpack a list:)

```
fruits = ["apple", "banana", "cherry"]
x, y, z = fruits
print(x)
print(y)
print(z)
```

2c.Python - Output Variables

Output Variables

Fungsi print() sering digunakan untuk menampilkan keluaran dari variabel.

Praktik17

```
x = "Python is awesome"
print(x)
```

Di dalam fungsi print(), Anda dapat menampilkan beberapa variabel yang dipisahkan dengan tanda koma:

Praktik18

```
x = "Python"
y = "is"
z = "awesome"
print(x, y, z)
```

Anda juga dapat menggunakan operator + untuk menampilkan beberapa variabel:

Praktik18

```
x = "Python "
y = "is "
z = "awesome"
print(x + y + z)
```

Perhatikan karakter spasi setelah "Python " dan "is ", tanpa spasi tersebut hasilnya akan menjadi "Pythonisawesome".

Untuk angka, karakter + berfungsi sebagai operator matematika.

Praktik19

```
x = 5
y = 10
print(x + y)
```

Di dalam fungsi print(), ketika Anda mencoba menggabungkan sebuah string dan angka menggunakan operator +, Python akan memberikan error:

Praktik20

```
x = 5
y = "John"
print(x + y)
```

Cara terbaik untuk menampilkan beberapa variabel dalam fungsi print() adalah dengan memisahkannya menggunakan tanda koma, yang bahkan mendukung tipe data yang berbeda:

Praktik21

```
x = 5
y = "John"
print(x, y)
```

2d.Python - Global Variables

Variabel Global

Variabel yang dibuat di luar sebuah fungsi (seperti pada semua contoh di halaman sebelumnya) disebut variabel global.
Variabel global dapat digunakan oleh siapa saja, baik di dalam fungsi maupun di luar fungsi.

Contoh

Buat sebuah variabel di luar fungsi, dan gunakan di dalam fungsi.

Praktik22

```
x = "awesome"

def myfunc():
  print("Python is " + x)

myfunc()
```

Jika Anda membuat sebuah variabel dengan nama yang sama di dalam sebuah fungsi, variabel tersebut akan menjadi variabel lokal, dan hanya dapat digunakan di dalam fungsi tersebut. Variabel global dengan nama yang sama akan tetap seperti semula, yaitu tetap menjadi variabel global dengan nilai aslinya.

Example

Buatlah sebuah variabel di dalam fungsi dengan nama yang sama seperti variabel global.

Praktik23

```
x = "awesome"

def myfunc():
  x = "fantastic"
  print("Python is " + x)

myfunc()

print("Python is " + x)
```

3.Python Data Types

Built-in Data Types

Dalam pemrograman, tipe data adalah konsep yang penting.
Variabel dapat menyimpan data dengan tipe yang berbeda, dan setiap tipe dapat melakukan hal yang berbeda.
Python memiliki tipe data bawaan berikut secara default, dalam kategori berikut:

Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`
None Type:	`NoneType`

Getting the Data Type

Anda dapat mengetahui tipe data dari objek apa pun dengan menggunakan fungsi type().

Example

Cetak tipe data dari variabel x:

Praktik23

```
x = 5
print(type(x))
```

Setting the Data Type

Dalam Python, tipe data ditentukan ketika Anda memberikan sebuah nilai kepada variabel.

Example	Data Type
`x = "Hello World"`	str
`x = 20`	int
`x = 20.5`	float
`x = 1j`	complex
`x = ["apple", "banana", "cherry"]`	list
`x = ("apple", "banana", "cherry")`	tuple
`x = range(6)`	range
`x = {"name" : "John", "age" : 36}`	dict
`x = {"apple", "banana", "cherry"}`	set
`x = frozenset({"apple", "banana", "cherry"})`	frozenset
`x = True`	bool
`x = b"Hello"`	bytes
`x = bytearray(5)`	bytearray
`x = memoryview(bytes(5))`	memoryview
`x = None`	NoneType

Setting the Specific Data Type

Jika Anda ingin menentukan tipe data secara khusus, Anda dapat menggunakan fungsi-fungsi konstruktor berikut:

Example

`x = str("Hello World")`	str
`x = int(20)`	int
`x = float(20.5)`	float
`x = complex(1j)`	complex

`x = list(("apple", "banana", "cherry"))`	list
`x = tuple(("apple", "banana", "cherry"))`	tuple
`x = range(6)`	range
`x = dict(name="John", age=36)`	dict
`x = set(("apple", "banana", "cherry"))`	set
`x = frozenset(("apple", "banana", "cherry"))`	frozenset
`x = bool(5)`	bool
`x = bytes(5)`	bytes
`x = bytearray(5)`	bytearray
`x = memoryview(bytes(5))`	memoryview

3a.Python Numbers

Python Numbers

Ada tiga jenis tipe numerik dalam Python:

- `int`
- `float`
- `complex`

Variabel dengan tipe numerik dibuat ketika Anda memberikan sebuah nilai kepadanya.

Praktik24

```
x = 1    # int
y = 2.8  # float
z = 1j   # complex
```

To verify the type of any object in Python, use the `type()` function:

Praktik25

```
print(type(x))
print(type(y))
print(type(z))
```

Int

Int, atau integer, adalah bilangan bulat, positif atau negatif, tanpa desimal, dengan panjang yang tidak terbatas.

Praktik25

```
x = 1
y = 35656222554887711
z = -3255522
```

```
print(type(x))
print(type(y))
print(type(z))
```

Float

Float, atau "floating point number", adalah bilangan positif atau negatif yang mengandung satu atau lebih angka desimal.

Praktik26

```
x = 1.10
y = 1.0
z = -35.59

print(type(x))
print(type(y))
print(type(z))
```

Float juga dapat berupa angka ilmiah dengan huruf "e" untuk menunjukkan pangkat dari 10.

Praktik27

```
x = 35e3
y = 12E4
z = -87.7e100

print(type(x))
print(type(y))
print(type(z))
```

3b.Python Casting

Specify a Variable Type

Ada kalanya Anda ingin menentukan tipe pada sebuah variabel. Hal ini dapat dilakukan dengan casting. Python adalah bahasa berorientasi objek, sehingga menggunakan kelas untuk mendefinisikan tipe data, termasuk tipe primitifnya.

Casting dalam Python dilakukan menggunakan fungsi-fungsi konstruktor berikut:

int() – membentuk bilangan integer dari literal integer, literal float (dengan menghapus semua angka desimal), atau literal string (asalkan string tersebut merepresentasikan bilangan bulat).

float() – membentuk bilangan float dari literal integer, literal float, atau literal string (asalkan string tersebut merepresentasikan bilangan float atau bilangan bulat).

str() – membentuk string dari berbagai jenis tipe data, termasuk string, literal integer, dan literal float.

Example

`Integers`

Praktik28

```
x = int(1)   # x will be 1
y = int(2.8) # y will be 2
z = int("3") # z will be 3
```

Example

Floats:

Praktik29

```
x = float(1)     # x will be 1.0
y = float(2.8)   # y will be 2.8
z = float("3")   # z will be 3.0
w = float("4.2") # w will be 4.2
```

Example

Strings:

Praktik30

```
x = str("s1") # x will be 's1'
y = str(2)    # y will be '2'
z = str(3.0)  # z will be '3.0'
```

4.Python Strings

Strings

String dalam Python dikelilingi oleh tanda kutip tunggal atau tanda kutip ganda.
'hello' sama dengan "hello".
Anda dapat menampilkan string literal dengan fungsi print():

Praktik 30

```
print("Hello")
print('Hello')
```

Quotes Inside Quotes

Anda dapat menggunakan tanda kutip di dalam sebuah string, selama tanda kutip tersebut tidak sama dengan tanda kutip yang digunakan untuk mengapit string tersebut.

Praktik31

```
print("It's alright")
print("He is called 'Johnny'")
print('He is called "Johnny"')
```

Assign String to a Variable

Memberikan sebuah string ke dalam variabel dilakukan dengan menuliskan nama variabel diikuti tanda sama dengan, lalu string-nya.

Praktik32

```
a = "Hello"
print(a)
```

Multiline Strings

Anda dapat memberikan sebuah string multibaris ke dalam variabel dengan menggunakan tiga tanda kutip.

Example

Anda dapat menggunakan tiga tanda kutip ganda.

Praktik33

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

Atau tiga tanda kutip tunggal.

Praktik34

```
a = '''Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.'''
print(a)
```

4b.Python - Slicing Strings

Slicing

Anda dapat mengembalikan rentang karakter dengan menggunakan sintaks slice. Tentukan indeks awal dan indeks akhir yang dipisahkan oleh tanda titik dua untuk mengembalikan sebagian dari string.

Example

Ambil karakter dari posisi 2 hingga posisi 5 (posisi 5 tidak termasuk):

Praktik35

```
b = "Hello, World!"
print(b[2:5])
```

Slice From the Start

Dengan menghilangkan indeks awal, rentang (range) akan dimulai dari karakter pertama.

Example

Dengan menghilangkan indeks awal, rentangnya akan dimulai dari karakter pertama.

Praktik36

```
b = "Hello, World!"
print(b[:5])
```

4c.Python - Modify Strings

Python memiliki sekumpulan metode bawaan yang dapat Anda gunakan pada string.

Upper Case

Example

Metode upper() mengembalikan string dalam huruf kapital.
Praktik37

```
a = "Hello, World!"
print(a.upper())
```

Lower Case

Example

Metode lower() mengembalikan string dalam huruf kecil.
Praktik38

```
a = "Hello, World!"
print(a.lower())
```

Remove Whitespace

Whitespace adalah spasi sebelum dan/atau setelah teks sebenarnya, dan sering kali Anda ingin menghapus spasi tersebut.

Example

Metode strip() menghapus semua whitespace dari awal atau akhir string.
Praktik39

```
a = " Hello, World! "
print(a.strip())
```

Replace String

Example

Metode replace() mengganti sebuah string dengan string lainnya.
Praktik40

```
a = "Hello, World!"
print(a.replace("H", "J"))
```

Split String

Metode split() mengembalikan sebuah list di mana teks di antara pemisah yang ditentukan menjadi item-item dalam list tersebut.

Example

Praktik41

```
a = "Hello, World!"
print(a.split(","))
```

4d.Python - String Concatenation

String Concatenation

Untuk menggabungkan dua string, Anda dapat menggunakan operator +.

Example

Gabungkan variabel a dengan variabel b ke dalam variabel c:

Praktik42

```
a = "Hello"
b = "World"
c = a + b
print(c)
```

Example

Untuk menambahkan spasi di antara keduanya, tambahkan " ":

Praktik43

```
a = "Hello"
b = "World"
c = a + " " + b
print(c)
```

4e.Python - Format - Strings

String Format

Seperti yang telah kita pelajari pada bab Variabel Python, kita tidak dapat menggabungkan string dan angka seperti ini:

Praktik 44

```
age = 36
#This will produce an error:
txt = "My name is John, I am " + age
print(txt)
```

F-Strings

F-String diperkenalkan di Python 3.6, dan sekarang menjadi cara yang disarankan untuk memformat string.

Untuk menentukan sebuah string sebagai f-string, cukup letakkan huruf f di depan literal string, dan tambahkan kurung kurawal {} sebagai penampung untuk variabel atau operasi lainnya.

Example

Create an f-string:

Praktik45

```
age = 36
txt = f"My name is John, I am {age}"
print(txt)
```

Placeholders and Modifiers

Penampung (placeholder) dapat berisi variabel, operasi, fungsi, dan modifier untuk memformat nilainya.

Example

Tambahkan penampung (placeholder) untuk variabel price:

Praktik46

```
price = 59
txt = f"The price is {price} dollars"
print(txt)
```

Penampung (placeholder) dapat menyertakan modifier untuk memformat nilai.

Modifier ditambahkan dengan menambahkan titik dua : diikuti oleh tipe format yang sah, seperti .2f yang berarti bilangan dengan titik tetap dan 2 angka desimal.

Example

Tampilkan harga dengan 2 angka desimal:

Praktik47

```
price = 59
txt = f"The price is {price:.2f} dollars"
print(txt)
```

Penampung (placeholder) dapat berisi kode Python, seperti operasi matematika:

Example

Lakukan operasi matematika di dalam penampung (placeholder), dan kembalikan hasilnya:
Praktik48

```
txt = f"The price is {20 * 59} dollars"
print(txt)
```

4f.Python - Escape Characters

Escape Character

Untuk memasukkan karakter yang tidak diperbolehkan dalam sebuah string, gunakan karakter escape.

Karakter escape adalah tanda backslash \ diikuti dengan karakter yang ingin Anda masukkan.

Contoh karakter yang tidak diperbolehkan adalah tanda kutip ganda di dalam string yang diapit oleh tanda kutip ganda:

Example

Anda akan mendapatkan error jika menggunakan tanda kutip ganda di dalam string yang diapit oleh tanda kutip ganda:

Praktik49

```
txt = "We are the so-called "Vikings" from the north."
```

```
Print(txt)
```

Untuk memperbaiki masalah ini, gunakan karakter escape \":

Example

Karakter escape memungkinkan Anda menggunakan tanda kutip ganda ketika biasanya tidak diperbolehkan:
Praktik50

```
txt = "We are the so-called \"Vikings\" from the north."


Print(txt)
```

Escape Characters

Karakter escape lain yang digunakan dalam Python:

Code	Result
\'	Single Quote
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Octal value
\xhh	Hex value

4g.Python - String Methods

String Methods

Python memiliki sekumpulan metode bawaan yang dapat Anda gunakan pada string.
Note: Semua metode string mengembalikan nilai baru. Mereka tidak mengubah string aslinya.

Method	Description

capitalize()	Mengubah karakter pertama menjadi huruf kapital.
casefold()	Mengubah seluruh huruf dalam string menjadi huruf kecil.
center()	Mengembalikan string yang diposisikan di tengah.
count()	Mengembalikan jumlah kemunculan suatu nilai tertentu dalam sebuah string.
encode()	Mengembalikan versi string yang telah dienkodeg
endswith()	Mengembalikan True jika string diakhiri dengan nilai tertentu.
expandtabs()	Menetapkan ukuran tab pada string.
find()	Mencari nilai tertentu dalam string dan mengembalikan posisi di mana nilai tersebut ditemukan.
format()	Memformat nilai-nilai yang ditentukan dalam sebuah string
format_map()	Memformat nilai-nilai tertentu di dalam sebuah string
index()	Mencari nilai tertentu dalam string dan mengembalikan posisi di mana nilai tersebut ditemukan
isalnum()	Mengembalikan True jika semua karakter dalam string adalah alfanumerik
isalpha()	Mengembalikan True jika semua karakter dalam string merupakan huruf alfabet
isascii()	Mengembalikan True jika semua karakter dalam string adalah karakter ASCII
isdecimal()	Mengembalikan True jika semua karakter dalam string adalah angka desimal
isdigit()	Mengembalikan True jika semua karakter dalam string adalah digit
isidentifier()	Mengembalikan True jika string merupakan sebuah identifier
islower()	Mengembalikan True jika semua karakter dalam string adalah huruf kecil
isnumeric()	Mengembalikan True jika semua karakter dalam string adalah numerik
isprintable()	Mengembalikan True jika semua karakter dalam string dapat dicetak
isspace()	Mengembalikan True jika semua karakter dalam string adalah spasi (whitespace)
istitle()	Mengembalikan True jika string mengikuti aturan

	penulisan judul (title case)
isupper()	Mengembalikan True jika semua karakter dalam string adalah huruf kapital
join()	Menggabungkan elemen-elemen dari iterable ke akhir string
ljust()	Mengembalikan versi string yang diratakan ke kiri
lower()	Mengubah string menjadi huruf kecil
lstrip()	Mengembalikan versi string dengan penghapusan spasi di sebelah kiri
maketrans()	Mengembalikan tabel translasi yang dapat digunakan dalam proses translasi
partition()	Mengembalikan sebuah tuple di mana string dibagi menjadi tiga bagian
replace()	Mengembalikan string di mana nilai tertentu diganti dengan nilai yang ditentukan
rfind()	Mencari nilai tertentu dalam string dan mengembalikan posisi terakhir di mana nilai tersebut ditemukan
rindex()	Mencari nilai tertentu dalam string dan mengembalikan posisi terakhir tempat nilai tersebut ditemukan
rjust()	**Mengembalikan versi string yang diratakan ke kanan**
rpartition()	Mengembalikan sebuah tuple di mana string dibagi menjadi tiga bagian
rsplit()	Memecah string pada pemisah yang ditentukan, dan mengembalikan sebuah list
rstrip()	Mengembalikan versi string dengan penghapusan spasi di sebelah kanan
split()	Memecah string pada pemisah yang ditentukan, dan mengembalikan sebuah daftar (list)
splitlines()	Memecah string pada pemisah baris (line break) dan mengembalikan sebuah daftar (list)
startswith()	Mengembalikan True jika string diawali dengan nilai yang ditentukan
strip()	Mengembalikan versi string yang telah dipangkas (trimmed)
swapcase()	Menukar huruf, huruf kecil menjadi huruf besar dan sebaliknya
title()	Mengubah karakter pertama dari setiap kata menjadi huruf kapital

translate()	Mengembalikan string yang telah diterjemahkan
upper()	Mengubah sebuah string menjadi huruf kapital
zfill()	Mengisi string dengan sejumlah nilai 0 tertentu di awal

5.Python Booleans

Tipe data boolean merepresentasikan salah satu dari dua nilai: True atau False.

Boolean Values

"Dalam pemrograman, Anda sering perlu mengetahui apakah sebuah ekspresi bernilai True atau False."

"Anda dapat mengevaluasi ekspresi apa pun di Python, dan mendapatkan salah satu dari dua jawaban, True atau False."

"Ketika Anda membandingkan dua nilai, ekspresi tersebut dievaluasi dan Python mengembalikan jawaban Boolean:"

Example

Praktik51

```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

Ketika Anda menjalankan sebuah kondisi dalam pernyataan if, Python mengembalikan True atau False:

Example

Cetak pesan berdasarkan apakah kondisinya adalah True atau False:

Praktik52

```
a = 200
b = 33

if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

Evaluate Values and Variables

Fungsi bool() memungkinkan Anda mengevaluasi nilai apa pun, dan memberikan hasil True atau False sebagai balasannya,

Example

Evaluasi sebuah string dan sebuah angka:

Praktik53

```
print(bool("Hello"))
print(bool(15))
```

Example

Evaluate two variables:

Praktik54

```
x = "Hello"
y = 15

print(bool(x))
print(bool(y))
```

Most Values are True

Hampir semua nilai akan dievaluasi sebagai True jika memiliki konten tertentu.

Setiap string bernilai True, kecuali string kosong.
Setiap angka bernilai True, kecuali 0.
Setiap list, tuple, set, dan dictionary bernilai True, kecuali yang kosong.

Example

Berikut ini akan mengembalikan True:

Praktik55

```
bool("abc")
bool(123)
bool(["apple", "cherry", "banana"])
```

Some Values are False

Bahkan, tidak banyak nilai yang dievaluasi sebagai False, kecuali nilai-nilai kosong, seperti (), [], {}, "", angka 0, dan nilai None. Dan tentu saja, nilai False itu sendiri dievaluasi sebagai False.

Example

Berikut ini akan mengembalikan False:

Praktik56

```
bool(False)
bool(None)
bool(0)
bool("")
bool(())
bool([])
bool({})
```

Ada satu nilai lagi, atau objek dalam konteks ini, yang dievaluasi sebagai False, yaitu jika Anda memiliki objek yang dibuat dari kelas dengan fungsi __len__ yang mengembalikan 0 atau False:

Example

Praktik57

```
class myclass():
  def __len__(self):
    return 0

myobj = myclass()
print(bool(myobj))
```

Functions can Return a Boolean

Anda dapat membuat fungsi yang mengembalikan nilai Boolean:

Example

Cetak jawaban dari sebuah fungsi:

Praktik58

```
def myFunction() :
  return True

print(myFunction())
```

Anda dapat mengeksekusi kode berdasarkan jawaban Boolean dari suatu fungsi:

Example

Cetak "YES!" jika fungsinya mengembalikan True, jika tidak cetak "NO!":

Praktik59

```
def myFunction() :
  return True

if myFunction():
  print("YES!")
else:
  print("NO!")
```

Python juga memiliki banyak fungsi bawaan yang mengembalikan nilai boolean, seperti fungsi isinstance(), yang dapat digunakan untuk menentukan apakah suatu objek bertipe data tertentu:

Example

Periksa apakah suatu objek merupakan integer atau bukan:

Praktik60

```
x = 200
print(isinstance(x, int))
```

6.Python Operators

Python Operators

Operator digunakan untuk melakukan operasi pada variabel dan nilai.

Pada contoh di bawah, kita menggunakan operator + untuk menjumlahkan dua nilai:

Example

Praktik61

```
print(10 + 5)
```

Meskipun operator + sering digunakan untuk menjumlahkan dua nilai, seperti pada contoh di atas, operator ini juga dapat digunakan untuk menjumlahkan sebuah variabel dengan sebuah nilai, atau dua variabel:

Example

Praktik62

```
sum1 = 100 + 50       # 150 (100 + 50)
sum2 = sum1 + 250     # 400 (150 + 250)
sum3 = sum2 + sum2    # 800 (400 + 400)
```

Python membagi operator ke dalam kelompok berikut:

Operator aritmatika

Operator penugasan (assignment)

Operator perbandingan (comparison)

Operator logika (logical)

Operator identitas (identity)

Operator keanggotaan (membership)

Operator bitwise

6a.Python Arithmetic Operators

Arithmetic Operators

Operator aritmatika digunakan dengan nilai numerik untuk melakukan operasi matematika umum:

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Examples

Berikut adalah contoh penggunaan berbagai operator aritmatika:

Praktik63

```
x = 15
y = 4

print(x + y)
print(x - y)
print(x * y)
print(x / y)
print(x % y)
print(x ** y)
print(x // y)
```

Division in Python

Python memiliki dua operator pembagian:

/ – Pembagian (mengembalikan nilai float)

// – Pembagian bulat (floor division, mengembalikan nilai integer)

Example

Pembagian (/) selalu mengembalikan nilai float:

Praktik64

```
x = 12
y = 5

print(x / y)
```

Example

Pembagian bulat (//) selalu mengembalikan bilangan bulat.

Hasilnya dibulatkan KE BAWAH ke bilangan bulat terdekat:

Praktik65

```
x = 12
y = 5

print(x // y)
```

6b.Python Assignment Operators

Assignment Operators

Assignment operators are used to assign values to variables:

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3
:=	print(x := 3)	x = 3 print(x)

6c.The Walrus Operator

Python 3.8 memperkenalkan operator :=, yang dikenal sebagai "walrus operator". Operator ini memungkinkan kita untuk **menetapkan nilai ke variabel sebagai bagian dari ekspresi yang lebih besar.**

Example

Praktik66

```
numbers = [1, 2, 3, 4, 5]
count = len(numbers)
if count > 3:
    print(f"List has {count} elements")

if (count := len(numbers)) > 3:
    print(f"List has {count} elements")
```

6d.Python Comparison Operators

Comparison Operators

Operator perbandingan digunakan untuk **membandingkan dua nilai**. Hasil dari perbandingan ini selalu berupa **Boolean** (`True` atau `False`).

Berikut adalah operator perbandingan di Python:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Examples

Operator perbandingan mengembalikan True atau False tergantung hasil perbandingannya.

Praktik67

```
x = 5
y = 3

print(x == y)
print(x != y)
print(x > y)
print(x < y)
print(x >= y)
print(x <= y)
```

Chaining Comparison Operators

Python memungkinkan Anda untuk menggabungkan operator perbandingan dalam satu ekspresi, yang disebut chained comparison. Hal ini membuat kode lebih ringkas dan mudah dibaca.

Example

Praktik68

```
x = 5

print(1 < x < 10)

print(1 < x and x < 10)
```

6e.Python Logical Operators

Logical Operators

Operator logika digunakan untuk **menggabungkan beberapa pernyataan kondisi**. Hasil dari operasi ini selalu berupa **Boolean** (`True` atau `False`).

Python memiliki tiga operator logika utama:

Operator	Description	Example
and	Mengembalikan True jika **kedua kondisi** bernilai True	x < 5 and x < 10
or	Mengembalikan True jika **salah satu kondisi** bernilai True	x < 5 or x < 4
not	Membalik nilai Boolean dari kondisi	not(x < 5 and x < 10)

Example

Untuk menguji apakah sebuah angka **lebih besar dari 0 dan lebih kecil dari 10**

Praktik69

```
x = 5

print(x > 0 and x < 10)
```

Example

Untuk menguji apakah sebuah angka **kurang dari 5 atau lebih besar dari 10**

Praktik70

```
x = 5

print(x < 5 or x > 10)
```

Example

Anda bisa membalik hasil **Boolean** menggunakan operator logika not.

Praktik71

```
x = 5

print(not(x > 3 and x < 10))
```

6f.Python Identity Operators

Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example
is	Mengembalikan True jika **kedua variabel menunjuk ke objek yang sama**	x is y
is not	Mengembalikan True jika **kedua variabel**	x is not y

Example

Operator is mengembalikan True jika **kedua variabel menunjuk ke objek yang sama** di memori, bukan hanya memiliki nilai yang sama.

Praktik72

```
x = ["apple", "banana"]
y = ["apple", "banana"]
z = x

print(x is z)
print(x is y)
print(x == y)
```

Example

Operator is not mengembalikan True jika **kedua variabel tidak menunjuk ke objek yang sama** di memori.

Praktik73

```
x = ["apple", "banana"]
y = ["apple", "banana"]

print(x is not y)
```

Difference Between `is` and `==`

- **is** - Memeriksa apakah **kedua variabel menunjuk ke objek yang sama di memori**.
- **==** - Untuk **memeriksa apakah nilai kedua variabel sama**

Example

Praktik74

```
x = [1, 2, 3]
y = [1, 2, 3]

print(x == y)
print(x is y)
```

7.Python If Statement

Python Conditions and If statements

- Python mendukung kondisi logika yang umum digunakan dalam matematika :
- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

Kondisi logika ini dapat digunakan dalam berbagai cara, paling umum dalam pernyataan if dan perulangan (loops).

Pernyataan if ditulis dengan menggunakan kata kunci if di Python. Pernyataan ini digunakan untuk menjalankan blok kode hanya jika kondisi tertentu bernilai True.

Example

If statement:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

Dalam contoh ini, kita menggunakan dua variabel, a dan b, sebagai bagian dari pernyataan if untuk menguji apakah b lebih besar dari a.

Nilai a adalah 33

Nilai b adalah 200

Karena 200 memang lebih besar dari 33, kondisi b > a bernilai True, sehingga Python akan menjalankan blok kode di dalam if dan menampilkan pesan:

How If Statements Work

Pernyataan if mengevaluasi sebuah kondisi (ekspresi yang menghasilkan True atau False).

Jika kondisi bernilai True, blok kode di dalam if akan dijalankan.

Jika kondisi bernilai False, blok kode tersebut dilewati dan tidak dijalankan.

Example

Checking if a number is positive:

Praktik75

```
number = 15
if number > 0:
  print("The number is positive")
```

7a.Python Elif Statement

The Elif Keyword

Kata kunci `elif` di Python digunakan untuk mengatakan:

"Jika kondisi sebelumnya **tidak True***, maka periksa kondisi ini."*

Dengan `elif`, kita bisa memeriksa **beberapa kondisi secara berurutan** dan mengeksekusi blok kode **pada kondisi pertama yang bernilai True**.

Example

Praktik76

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

7b.Python Else Statement

The Else Keyword

Kata kunci `else` menangkap semua kondisi yang **tidak terpenuhi oleh pernyataan** `if` **atau** `elif` **sebelumnya**.

Blok kode di dalam `else` **dijalankan hanya jika semua kondisi sebelumnya bernilai False**.

Example

Praktik77

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

Dalam contoh ini, a lebih besar dari b, sehingga kondisi pertama pada if tidak True.

Kondisi elif juga tidak True,

Maka Python mengeksekusi blok kode else dan menampilkan pesan:

7c.Python Shorthand If

Short Hand If

Jika hanya ada satu pernyataan yang ingin dijalankan, kita bisa menulisnya di baris yang sama dengan if.

Example

Pernyataan if satu baris (one-line if statement)

Praktik88

```
a = 5
b = 2
if a > b: print("a is greater than b")
```

7d.Python Logical Operators

Python Logical Operators

Operator logika digunakan untuk menggabungkan beberapa pernyataan kondisi sehingga kita bisa membuat ekspresi yang lebih kompleks. Python memiliki tiga operator logika utama:

- and - Mengembalikan True jika kedua kondisi True
- or - Mengembalikan True jika salah satu kondisi True
- not - Membalik nilai Boolean dari kondisi

The and Operator

Kata kunci and adalah operator logika yang digunakan untuk **menggabungkan beberapa kondisi**.

> **Kedua kondisi harus True** agar keseluruhan ekspresi bernilai True.
>
> Jika salah satu kondisi False, maka hasilnya False.

Example

Untuk menguji apakah a lebih besar dari b dan c lebih besar dari a, kita bisa menggunakan operator logika and.:

Praktik89

```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

7e.Python Nested If

Nested If Statements

Anda bisa menempatkan if **di dalam** if. Ini disebut **nested if statements** (pernyataan if bersarang).

Example

Praktik90

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

Dalam contoh ini, if **bagian dalam (inner if)** hanya dijalankan jika **kondisi luar (outer if)** x > 10 bernilai True..

8.Python Match

Pernyataan match digunakan untuk melakukan **aksi yang berbeda berdasarkan kondisi yang berbeda.**

The Python Match Statement

Alih-alih menulis banyak pernyataan `if..elif..else`, Anda bisa menggunakan `match`.

> Pernyataan `match` memilih **salah satu blok kode** untuk dijalankan berdasarkan nilai dari sebuah variabel.
>
> Ini membuat kode lebih **rapi dan mudah dibaca** ketika ada banyak kemungkinan kondisi.

Syntax

```
match expression:
  case x:
    code block
  case y:
    code block
  case z:
    code block
```

Begini cara kerja match:

- Ekspresi pada match **dievaluasi satu kali.**

- Nilai dari ekspresi tersebut dibandingkan dengan nilai pada setiap case.
- Jika ada **cocok**, blok kode yang terkait dengan case itu akan dijalankan.

Contoh: menggunakan nomor hari untuk mencetak nama hari:

Praktik91

```
day = 4
match day:
  case 1:
    print("Monday")
  case 2:
    print("Tuesday")
  case 3:
    print("Wednesday")
  case 4:
    print("Thursday")
  case 5:
    print("Friday")
  case 6:
    print("Saturday")
  case 7:
    print("Sunday")
```

9.Python While Loops

Python Loops

Python memiliki **dua perintah loop primitif**, yaitu:

- while loops
- for loops

The while Loop

Dengan while **loop**, kita dapat mengeksekusi sekumpulan pernyataan **selama kondisi tertentu bernilai True**.

Example

Berikut contoh menggunakan while loop untuk mencetak i selama i kurang dari 6::

Praktik93

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Note: Ya, penting untuk meningkatkan nilai i di dalam while loop, jika tidak, kondisi akan selalu True dan loop akan berjalan tanpa henti (infinite loop).

Loop `while` membutuhkan variabel yang relevan **siap digunakan sebelum loop dimulai**.

> Dalam contoh ini, kita perlu **mendefinisikan variabel indeks** `i` terlebih dahulu, yang akan digunakan untuk memeriksa kondisi.
>
> Kita menetapkan `i = 1` sebelum memulai loop.

Example:

Praktik93B

```
i = 1  # variabel indeks siap digunakan

while i < 6:

  print(i)

  i += 1
```

The break Statement

With the `break` statement we can stop the loop even if the while condition is true:

Example

Exit the loop when i is 3:

Praktik94

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

9a.Python For Loops

Python For Loops

for loop digunakan untuk mengulangi setiap item dalam sebuah urutan (seperti list, tuple, dictionary, set, atau string).

Berbeda dengan for di beberapa bahasa pemrograman lain, for di Python bekerja seperti iterator, yang berjalan satu per satu pada setiap item dalam urutan.

Dengan for loop, kita dapat mengeksekusi blok kode untuk setiap item dalam urutan tersebut..

Example

mencetak setiap buah dalam daftar:

Praktik95

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

Untuk loop` di Python tidak memerlukan variabel indeks yang harus diinisialisasi sebelumnya.

Variabel loop secara otomatis **mengambil setiap nilai dari urutan** yang diiterasi.

Kita tidak perlu mendefinisikan atau mengatur nilai awal seperti pada `while` loop.

Looping Through a String

Bahkan **string** merupakan objek yang dapat diiterasi (**iterable object**), karena mereka terdiri dari **urutan karakter**.

Example

Berikut contoh menggunakan for **loop** untuk mengulang setiap huruf dalam kata "banana":

Praktik96

```
for x in "banana":
  print(x)
```

The break Statement

Dengan break **statement**, kita dapat **menghentikan loop sebelum semua item diiterasi**.

Example

Berikut contoh menggunakan break untuk keluar dari loop saat x bernilai "banana"

Praktik97

```
ruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

Example

Berikut contoh di mana break ditempatkan **sebelum** print, sehingga loop akan berhenti sebelum mencetak "banana":

Praktik98

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

10.Python Functions

Python Functions

Fungsi (**function**) adalah **blok kode** yang hanya dijalankan ketika **dipanggil**.

Beberapa hal penting tentang fungsi:

> **Hanya berjalan saat dipanggil** – kode di dalam fungsi tidak otomatis dijalankan saat program dimulai.
>
> **Dapat mengembalikan nilai** – fungsi bisa menggunakan `return` untuk memberikan hasil.
>
> **Mencegah pengulangan kode** – dengan fungsi, kita bisa menulis kode sekali dan memanggilnya berkali-kali.

Creating a Function

Di Python, sebuah fungsi didefinisikan menggunakan kata kunci def, diikuti dengan **nama fungsi** dan **tanda kurung.**:

Example

Praktik99

```
def my_function():
  print("Hello from a function")
```

This creates a function named `my_function` that prints "Hello from a function" when called.

Kode di dalam fungsi **harus diberi indentasi**.

Python menggunakan **indentasi** untuk menentukan **blok kode**, bukan tanda kurung atau kata kunci khusus seperti di beberapa bahasa lain.

Semua perintah yang termasuk dalam fungsi harus **dalam satu level indentasi** yang sama.

Calling a Function

Untuk memanggil fungsi, tulis nama fungsi diikuti dengan tanda kurung ().

Example

Praktik100

```
def my_function():
  print("Hello from a function")

my_function()
```

Example

Praktik101

```
def my_function():
  print("Hello from a function")

my_function()
my_function()
my_function()
```

Python Function Arguments

Arguments

Informasi dapat **dikirim ke fungsi sebagai argumen**.

Argumen dituliskan **setelah nama fungsi**, di dalam tanda kurung.

Bisa menambahkan **beberapa argumen**, dipisahkan dengan koma.

Argumen memungkinkan fungsi menggunakan data yang diberikan saat dipanggil.

Example

Praktik102

```
def my_function(name): # name is a parameter
  print("Hello", name)
```

```
my_function("Emil") # "Emil" is an argument
```