MODUL 4

OPERATORS

A. Tujuan

1. Mampu memahami konsep dari operator penugasan, aritmatika, relasional, logika, string dan bitwise
2. Mampu menerapkan penggunaan dari operator penugasan, aritmatika, relasional, logika, string dan bitwise

B. Operator Aritmetika

Operator aritmetika digunakan pada tipe data numerik, untuk melakukan operasi matematika sederhana yang terdiri atas:

Tabel 4.1 Pengunaan Operator Aritmetika

Simbol Operator	Keterangan	Contoh
+	Penambahan	3 + 2 menghasilkan output: 5
-	Pengurangan	4 – 2 menghasilkan output: 2
*	Perkalian	3 * 2 menghasilkan output: 6
/	Pembagian	3/2 menghasilkan output: 1.5
%	Modulo/ sisa pembagian	3 % 2 menghasilkan output: 1 8 % 2 menghasilkan output: 0
**	Perpangkatan	3**2 menghasilkan output 9
//	Pembagian dengan pembulatan ke bawah	3 // 2 menghasilkan output 1

Example 4.1

```
x = 15
y = 4

#Output: x + y = 19
print('x + y = ', x+y)

#Output: x - y = 11
print('x - y = ',x-y)

# Output: x * y = 60
print('x * y = ',x*y)

# Output: x / y = 3.75
print('x / y = ',x/y)

# Output: x // y = 3
print('x // y = ',x//y)

# Output: x ** y = 50625
print('x ** y = ',x**y)
```

C. Operator Relasional/*Comparison*

Operator comparison dapat digunakan untuk membandingkan dua buah nilai, berikut merupakan contoh-contoh operator komparasi.

Tabel 4.2 Penggunaan Operator Relasional/Komparasi

Simbol Operator	Keterangan	Contoh
= =	Persamaan	33 == 33 menghasilkan output: **True** 34 == 33 menghasilkan output: **False**
! =	Pertidaksamaan	34 != 33 menghasilkan output: **True** 33 != 33 menghasilkan output: **False**
>	Lebih besar dari	34 > 33 menghasilkan output: **True** 33 > 34 menghasilkan output: **False**
<	Lebih kecil dari	33 < 34 menghasilkan output: **True** 34 < 33 menghasilkan output: **False**
> =	Lebih besar atau sama dengan	34 >= 33 menghasilkan output: **True** 34 >= 34 menghasilkan output: **True** 33 >= 34 menghasilkan output: **False**
< =	Lebih kecil atau sama dengan	33 <= 34 menghasilkan output: **True** 33 <= 33 menghasilkan output: **True** 34 <= 33 menghasilkan output: **False**

Example 4.2

```
x = 15
x = 10
y = 12
# Output: x > y is False
print('x > y is',x>y)
# Output: x < y is True
print('x < y is',x<y)
# Output: x == y is False
print('x == y is',x==y)
# Output: x != y is True
print('x != y is',x!=y)
# Output: x >= y is False
print('x >= y is',x>=y)
# Output: x <= y is True
print('x <= y is',x<=y)
```

D. Operator Logika

Operator Logika digunakan untuk menggabungkan beberapa nilai kebenaran atas suatu statemen logika.

Tabel 4.3 Pengunaan Operator Logika

Simbol Operator	Keterangan	Contoh
and	'and' menerima dua nilai kebenaran dan mengembalikan nilai 'True' jika keduanya benar	x = 5 x >= 1 and x <= 10 mengembalikan nilai **True** x = 5 x >= 1 and x <= 4 mengembalikan nilai **False**
Or	'or' menerima dua nilai kebenaran dan mengembalikan nilai 'True' jika salah satu benar	x = 3 x >= 1 or x <= 2 mengembalikan nilai **True** karena statement logika pertama terpenuhi x = 3 x >= 5 or x <= 0 mengembalikan nilai **False** karena kedua statement logika tidak terpenuhi
not	'not' menerima sebuah nilai kebenaran dan mengembalikan komplemennya	x = 7 not(x == 7) mengembalikan nilai **False** not(x >= 10) mengembalikan nilai **True**

Example 4.3

```
x = True
y = False
print('x and y is',x and y)
print('x or y is',x or y)
print('not x is',not x)
```

E. Operator String

1. Concat Strings

Kita dapat menggabungkan dua nilai string menggunakan operator +. Dalam contoh berikut kita menggabungkan dua string untuk mendapatkan string ketiga.

Example 4.4

```
# strings
str1 = "Hello"
str2 = "World"
# concat
result = str1 + " " + str2
# output
print(result)
```

Program di atas akan menggabungkan (join) string pertama str1 diikuti dengan spasi dan kemudian string kedua str2. Jadi, kita akan mendapatkan output ‘Hello World’.

2. Mereplikasi Strings

Kita dapat mereplikasi string yang diberikan N kali menggunakan operator *. Dalam contoh berikut kami mereplikasi string HA 3 kali.

Example 4.5

```
# string
str = "HA"
# replicate
result = str * 3
# output
print(result)
```

Kode diatas akan memberikan string baru ‘HAHAHA’.

3. Pengecekan Membership - in

Kita dapat menggunakan operator in untuk memeriksa apakah string pencarian ada dalam string tertentu. Kita akan mendapatkan ‘True’ jika string pencarian ditemukan, ‘False’ jika sebaliknya.

Dalam program Python berikut kita memeriksa apakah string pencarian ‘lo’ ada dalam string ‘Hello World’.

Example 4.6

```
# strings
needle = "lo"
haystack = "Hello World"
# check
if needle in haystack:
    print(needle, "is present in the string", haystack)
else:
    print("Not found")
```

Kita akan mendapatkan output sebagai berikut dari program diatas.

```
lo is present in the string Hello World
```

4. Pengecekan Membership – not in

Kita dapat menggunakan operator not in untuk memeriksa apakah string pencarian tidak ada dalam string tertentu. Kita akan mendapatkan ‘True’ jika string pencarian tidak ditemukan dan ‘False’ jika sebaliknya.

Dalam program Python berikut kita akan memeriksa apakah string pencarian ‘HA’ ada dalam string yang diberikan, ‘Hello World’.

Example 4.7

```
# strings
needle = "HA"
haystack = "Hello World"
# check
if needle in haystack:
    print(needle, "is present in the string", haystack)
else:
    print("Not found")
```

Kita akan mendapatkan output ‘Not Found’ dari program diatas.

5. Mengakses Karakter dalam String

Kita menggunakan str [i] untuk mendapatkan karakter pada indeks i dalam string yang diberikan str. Indeks dimulai dari 0 jadi, karakter pertama di indeks 0, karakter kedua di 1 dan seterusnya. Dalam contoh berikut kita akan mengekstrak karakter kedua (indeks 1) dari string "Jane Doe".

Example 4.8

```
# string
str = "Jane Doe"
# character
ch = str[1]
# output
print(ch) # a
```

6. Substring

Kita menggunakan str [start: end] untuk mendapatkan substring dari string tertentu. Kita mulai dari indeks awal dan mengekstrak substring hingga indeks akhir tanpa menyertakannya. Dalam contoh berikut kita akan mengekstrak substring lo dari string Hello World.

Example 4.9

```
# string
str = "Hello World"
# substring
substr = str[3:5]
# output
print(substr) # lo
```

7. Skipping Characters

Kita dapat melewati karakter dari sebuah string menggunakan str [start: end: step]. Dimana, start adalah indeks awal. end mewakili indeks terakhir (tidak termasuk) hingga string diekstraksi dan langkah adalah jumlah langkah yang harus diambil. Dalam contoh berikut kita melewatkan 1 karakter untuk string yang diberikan "Hello World".

String "Hello World" memiliki 11 karakter dan kita ingin melewatkan karakter yang diindeks ganjil jadi, langkahnya adalah 2. Karena kita mempertimbangkan seluruh string jadi, kita bisa mengabaikan end. Kita akan menggunakan str [start :: step].

Jadi, kita akan mempertimbangkan karakter berikut: 0-> 2-> 4-> 6-> 8-> 10

Example 4.10

```
string:  Hello World
skipping: x x x x x
final string: HloWrd
```

```
# string
str = "Hello World"
# skip
new_str = str[0::2]
```

8. Reverse String

Cara termudah untuk membalikkan string dengan Python adalah dengan menulis str [:: - 1]. Dalam program Python berikut kita akan membalikkan string "Hello World".

Example 4.11

```
# string
str = "Hello World"
# reverse
result = str[::-1]
# output
print(result)
```

Program di atas akan memberi output ‘dlroW olleH’.

9. Escape Sequence

Urutan escape sequence mewakili karakter yang tidak dapat dicetak. Mereka mulai dengan garis miring terbalik. Berikut adalah beberapa urutan escape.

Tabel 4.4 Escape Sequence

Description	Escape Sequence Notation
Alert or Bell	\a
Backspace	\b
Form feed	\f
New line	\n
Carriage return	\r
Tab	\t

F. Operator Bitwise

Operator bitwise bertindak atas operan seolah-olah itu adalah string digit biner. Mereka beroperasi sedikit demi sedikit. Misalnya, 2 adalah 10 dalam biner dan 7 adalah 111. Pada tabel di bawah ini: Misalkan x = 10 (0000 1010 dalam biner) dan y = 4 (0000 0100 dalam biner).

Tabel 4.5 Operator Bitwise

Operator	Arti	Contoh
&	Bitwise AND	x & y = 0 (0000 0000)
|	Bitwise OR	x y =14 (0000 1110)|
-	Bitwise NOT	x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = (0000 1110)
>>	Bitwise right shift	x >> 2 = (0000 0010)
<<	Bitwise left shift	x << 2 = (0010 1000)

Example 4.12

```
a = 60 # 60 = 0011 1100
b = 13 # 13 = 0000 1101
c = 0
c = a & b; # 12 = 0000 1100
print ("Line 1 - Value of c is ", c)
c = a | b; # 61 = 0011 1101
print ("Line 2 - Value of c is ", c)
c = a ^ b; # 49 = 0011 0001
print ("Line 3 - Value of c is ", c)
a = 60 # 60 = 0011 1100
b = 13 # 13 = 0000 1101
c = 0
c = a & b; # 12 = 0000 1100
print ("Line 1 - Value of c is ", c)
c = a | b; # 61 = 0011 1101
print ("Line 2 - Value of c is ", c)
c = a ^ b; # 49 = 0011 0001
print ("Line 3 - Value of c is ", c)
```