MODUL IV

OPERATORS

PRAKTIKUM PROGRAMA KOMPUTER TI 20 - UNS

A. OPERATOR ARITMETIKA

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

Simbol Operator	Keterangan	Contoh
+	Penambahan	**3 + 2** akan menghasilkan output: 5
-	Pengurangan	**4 - 2** akan menghasilkan output: 2
*	Perkalian	**3 * 2** akan menghasilkan output: 6
/	Pembagian	**3 / 2** akan menghasilkan output: 1.5
%	Modulo/sisa bagi	**3 % 2** akan menghasilkan output: 1 dikarenakan 3 tidak habis dibagi 2 dan menyisakan 1 **8 % 2** akan menghasilkan output: 0 dikarenakan 8 habis dibagi 2
**	Pangkat	**3 ** 2** akan menghasilkan output: 9
//	Pembagian dengan pembulatan ke bawah	**3 / 2** akan menghasilkan output: 1 dikarenakan 1.5 akan menjadi 1 saat dibulatkan ke bawah.

EXERCISE 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)
```

B. OPERATOR RELASIONAL/COMPARISON

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

Simbol Operator	Keterangan	Contoh
==	Persamaan	**33 == 33** akan menghasilkan output: **True** dikarenakan benar 33 sama dengan 33 **34 == 33** akan menghasilkan output: **False** dikarenakan 34 tidak sama dengan 33
!=	Pertidaksamaan	**34 != 33** akan menghasilkan output: **True** dikarenakan benar bahwa 34 tidak sama dengan 33 **33 != 33** akan menghasilkan output: **False** dikarenakan 33 sama dengan 33
>	Lebih besar dari	**34 > 33** akan menghasilkan output: **True** dikarenakan 34 lebih besar dari 33 **33 > 34** akan menghasilkan output **False** dikarenakan tidak benar 33 lebih besar dari 34
<	Lebih kecil dari	**33 < 34** akan menghasilkan output **True** dikarenakan benar 33 lebih kecil dari 34 **34 < 33** akan menghasilkan output: **False** dikarenakan tidak benar 34 lebih kecil dari 33
>=	Lebih besar atau sama dengan	**34 >= 33** akan menghasilkan output **True** dikarenakan 34 lebih besar dari 33 **34 >= 34** akan menghasilkan output **True** dikarenakan 34 sama dengan 34 **33 >= 34** akan menghasilkan output **False** dikarenakan 33 tidak lebih besar dari 34 dan tidak sama dengan 34
<=	Lebih kecil atau sama dengan	**33 <= 34** akan menghasilkan output **True** dikarenakan 33 lebih kecil dari 34 **33 <= 33** akan menghasilkan output **True** dikarenakan 34 sama dengan 33 **34 <= 33** akan menghasilkan output **False** dikarenakan 34 tidak lebih kecil dari 33 dan tidak sama dengan 34

EXERCICE 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)
```

C. OPERATOR LOGIKA

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

Simbol Operator	Keterangan	Contoh
and	dan - menerima dua nilai kebenaran dan mengembalikan nilai benar jika keduanya benar	x = 5 x >= 1 and x <= 10 akan mengembalikan nilai **True** x = 5 x >= 1 and x >= 4 akan mengembalikan nilai **False**
or	atau - menerima dua nilai kebenaran dan mengembalikan nilai benar jika salah satu benar	x = 3 x >= 1 or x <= 2 akan mengembalikan nilai **True** dikarenakan statemen logika pertama terpenuhi x = 3 x >= 5 or x <= 0 akan mengembalikan nilai **False** dikarenakan kedua statemen logika tidak terpenuhi (bernilai False)
not	negasi - menerima sebuah nilai kebenaran dan mengembalikan komplemennya	x = 7 not(x == 7) akan mengembalikan nilai **False** not(x >= 10) akan mengembalikan nilai **True**

EXERCISE 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)
```

D. OPERATOR STRING

1. Concat strings

Kita dapat menggabungkan dua nilai string menggunakan operator +.

Dalam contoh berikut kami menggabungkan dua string untuk mendapatkan string ketiga.

EXERCISE 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.

   EXERCISE 4.5

```
# string

str = "HA"



# replicate

result = str * 3



# output

print(result)
```

The above code will give us a new string HAHAHA.

3. **Pengecekan membership– in**

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

   Dalam program Python berikut kami memeriksa apakah string pencarian lo ada dalam string yang diberikan Hello World.

   EXERCISE 4.6

```
# strings

needle = "HA"

haystack = "Hello World"



# check

if needle in haystack:

  print(needle, "is present in the string", haystack)

else:

  print("Not found")
```

We will get the following output for the above program.

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

4. **Checking membership - not in**

   Kita dapat menggunakan operator not in untuk memeriksa apakah string pencarian tidak ada dalam string tertentu. Kami mendapatkan True jika string pencarian tidak ditemukan, False sebaliknya.

   Dalam program Python berikut kami memeriksa apakah string pencarian HA ada dalam string yang diberikan Hello World.

EXERCISE 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")
```

Kami akan mendapatkan Not Found sebagai output dari kode di atas.

5. **Mengakses karakter dalam string**

   Kami 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 kami mengekstrak karakter kedua (indeks 1) dari string "Jane Doe".

   EXERCISE 4.8

```
# string

str = "Jane Doe"



# character

ch = str[1]
```

```
# output

print(ch)      # a
```

6. **Substring**

   Kami menggunakan str [start: end] untuk mendapatkan substring dari string tertentu. Kami mulai dari indeks awal dan mengekstrak substring hingga indeks akhir tanpa menyertakannya.

   Dalam contoh berikut kami mengekstrak substring lo dari string Hello World.

   EXERCISE 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 kami melewatkan 1 karakter untuk string yang diberikan "Hello World".

   String "Hello World" memiliki 11 karakter dan kami ingin melewatkan karakter yang diindeks ganjil jadi, langkahnya adalah 2.

   Karena kita mempertimbangkan seluruh string jadi, kita bisa mengabaikan end. Kami akan menggunakan str [start :: step].

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

EXERCISE 4.10

```
string:    Hello World

skipping:   x x x x x    # characters marked with x

final str: 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 kami membalikkan string "Hello World".

   EXERCISE 4.11

```
# string

str = "Hello World"



# reverse

result = str[::-1]



# output

print(result)
```

Kode di atas akan memberi kita 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.

Description	Escape sequence notation
Alert or Bell	`\a`
Backspace	`\b`
Escape	`\e`
Form feed	`\f`
New line	`\n`
Carriage return	`\r`
Space	`\s`
Tab	`\t`

E. OPERATOR BITWISE

Operator bitwise bertindak atas operan seolah-olah itu adalah string digit biner. Mereka beroperasi sedikit demi sedikit, itulah namanya.

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)

Operator	Meaning	Example
&	Bitwise AND	x & y = 0 (0000 0000)
|	Bitwise OR	x | y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x >> 2 = 2 (0000 0010)
<<	Bitwise left shift	x << 2 = 40 (0010 1000)

EXERCISE 4.12

```
#!/usr/bin/python


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
```

```
c = ~a;           # -61 = 1100 0011

print "Line 4 - Value of c is ", c


c = a << 2;       # 240 = 1111 0000

print "Line 5 - Value of c is ", c


c = a >> 2;       # 15 = 0000 1111

print "Line 6 - Value of c is ", c
```

SUMBER :

www.dqlab.com

www.faceprep.in

www.programiz.com