MODUL PRAKTIKUM

PROGRAMA KOMPUTER

PROGRAM STUDI TEKNIK INDUSTRI
UNIVERSITAS SEBELAS MARET

TIM ASISTEN LABORATORIUM PERANCANGAN DAN OPTIMASI SISTEM INDUSTRI 2020

MODUL III

OPERATOR

A. Tujuan

Berikut merupakan tujuan Praktikum Programa Komputer Modul III.

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

B. Materi

Operator adalah simbol yang digunakan untuk melakukan operasi tertentu seperti operasi aritmatika (penjumlahan, pengurangan, pembagian dan perkalian) dan operasi lainnya. Operator pada Pemrograman Python dapat disimbolkan dengan tanda atau karakter seperti **+, -, *, /, **, %** dan sebagainya.

Bahasa Pemrograman Python mendukung berbagai macam jenis operator. Secara garis besar, Python memiliki **delapan jenis operator** seperti operator aritmatika, pembanding, penugasan, logical, keanggotaan, identitas, string dan bitwise. Terdapat jenis-jenis operator dalam pemrograman python yaitu :

1. Aritmatika

Operator Aritmatika merupakan operator yang sering digunakan, terutama saat memecahkan sebuah kasus yang berhubungan dengan hitung menghitung (tipe data numerik). Operator Aritmatika terdiri dari tujuh operator aritmatika seperti berikut:

Tabel 1 Contoh Penulisan Operator Aritmatika

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

```python
a = 10
b = 3

# Operasi Penjumlahan (+)
hasil = a + b
print(a,'+',b,'=',hasil)

# Operasi Pengurangan (-)
hasil = a - b
print(a,'-',b,'=',hasil)

# operasi perkalian (*)
hasil = a * b
print(a,'*',b,'=',hasil)

# operasi pembagian /
hasil = a / b
print(a,'/',b,'=',hasil)

# operasi eksponen (pangkat) **
hasil = a ** b
print(a,'**',b,'=',hasil)

# operasi modulus %
hasil = a % b
print(a,'%',b,'=',hasil)

# operasi floor division //
hasil = a // b
print(a,'//',b,'=',hasil)
```

```
10 + 3 = 13
10 - 3 = 7
10 * 3 = 30
10 / 3 = 3.3333333333333335
10 ** 3 = 1000
10 % 3 = 1
10 // 3 = 3
```

Gambar 1 Contoh Penulisan Operator Aritmatika

Seperti pada operasi aritmatika pada Matematika Dasar, operasi aritmatika pada Python memiliki prioritas untuk dikerjakan terlebih dahulu. Berikut urutan prioritas dari operasi aritmatika.

1. Kurung ()
2. Exponen atau pangkat **
3. Perkalian, pembagian, modulus, *floor division* * / ** % //
4. Penjumlahan dan pengurangan + -

```
x = 3
y = 2
z = 4

hasil = x ** y * (z + x) / y - y % z // x
print(x,'**',y,'*',z,'+',x,'/',y,'-',y,'%',z,'//',x,'=',hasil)
```

✓ 0.0s

```
3 ** 2 * 4 + 3 / 2 - 2 % 4 // 3 = 31.5
```

```
hasil = x + y * z
print(x,'+',y,'*',z,'=',hasil)

# Contoh kurung akan menjadi prioritas
hasil = (x + y) * z
print('(',x,'+',y,') *',z,'=',hasil)
```

✓ 0.0s

```
3 + 2 * 4 = 11
( 3 + 2 ) * 4 = 20
```

Gambar 2 Contoh Prioritas Operator Aritmatika

2. Perbandingan atau Relasi

Operator perbandingan/relasi adalah operator yang bertugas untuk membandingkan antara dua buah nilai. Hasil dari operasi perbandingan adalah Boolean.

Jika hasil perbandingan benar, maka akan menghasilkan nilai **True**, dan sebaliknya jika salah maka akan menghasilkan nilai **False**. Berikut ini adalah enam jenis operator Perbandingan pada Python.

Tabel 2 Contoh Perbandingan dan Relasi

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**

```python
a = 4
b = 2

# Lebih besar dari >
print('========== lebih besar dari (>)')
hasil = a > 3
print(a,'>',3,'=',hasil)
hasil = b > 3
print(b,'>',3,'=',hasil)
hasil = b > 2
print(b,'>',2,'=',hasil)

# kurang dari <
print('========== kurang dari (<)')
hasil = a < 3
print(a,'<',3,'=',hasil)
hasil = b < 3
print(b,'<',3,'=',hasil)
hasil = b < 2
print(b,'<',2,'=',hasil)

# Lebih dari sama dengan >=
print('========== lebih dari sama dengan(>=)')
hasil = a >= 3
print(a,'>=',3,'=',hasil)
hasil = b >= 3
print(b,'>=',3,'=',hasil)
hasil = b >= 2
print(b,'>=',2,'=',hasil)

# kurang dari sama dengan <=
print('======== kurang dari sama dengan(<=)')
hasil = a <= 3
print(a,'<=',3,'=',hasil)
hasil = b <= 3
print(b,'<=',3,'=',hasil)
hasil = b <= 2
print(b,'<=',2,'=',hasil)

# sama dengan (==)
print('======== sama dengan(==)')
hasil = a == 4
print(a,'==',4,'=',hasil)
hasil = b == 4
print(b,'==',4,'=',hasil)

# tidak sama dengan (!=)
print('======== sama dengan(!=)')
hasil = a != 4
print(a,'!=',4,'=',hasil)
hasil = b != 4
print(b,'!=',4,'=',hasil)
```

```
========== lebih besar dari (>)
4 > 3 = True
2 > 3 = False
2 > 2 = False
========== kurang dari (<)
4 < 3 = False
2 < 3 = True
2 < 2 = False
========== lebih dari sama dengan(>=)
4 >= 3 = True
2 >= 3 = False
2 >= 2 = True
======== kurang dari sama dengan(<=)
4 <= 3 = False
2 <= 3 = True
2 <= 2 = True
======== sama dengan(==)
4 == 4 = True
2 == 4 = False
======== sama dengan(!=)
4 != 4 = False
2 != 4 = True
```

Gambar 3 Contoh Perbandingan dan Relasi

3. Penugasan (*Assignment)*

Operator *Assignment* adalah operator untuk memasukkan suatu nilai ke dalam variabel. Operator ini akan mempersingkat operasi.

Dalam Bahasa Pemrograman Python, Operator *Assignment* menggunakan tanda sama dengan (=). Misal nilai = 29, artinya nilai telah diberi tugas untuk menyimpan angka 29. Berikut operator penugasan lainnya:

Tabel 3 Penjelasan Operator Penugasan

Simbol	Keterangan	Contoh
=	Sama dengan	a = 1 Memberikan nilai di kanan ke dalam variabel yang berada di sebelah kiri.
+=	Tambah sama dengan	a += 2 Memberikan nilai variabel dengan nilai variabel itu sendiri ditambah dengan nilai di sebelah kanan.
-=	Kurang sama dengan	a -= 2 Memberikan nilai variabel dengan nilai variabel itu sendiri dikurangi dengan nilai di sebelah kanan.
*=	Kali sama dengan	a *= 2 Memberikan nilai variabel dengan nilai variabel itu sendiri dikali dengan nilai di sebelah kanan.
/=	Bagi sama dengan	a /= 4 Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan.
%=	Sisa bagi sama dengan	a %= 3 Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan. Yang diambil nantinya adalah sisa baginya.
**=	Pangkat sama dengan	a **= 3 Memberikan nilai variabel dengan nilai variabel itu sendiri dipangkatkan dengan nilai di sebelah kanan.
//=	Pembagian bulat sama dengan	a //= 3 Membagi bulat nilai variabel sebelah kiri operator dengan nilai sebelah kanan operator kemudian hasilnya diisikan ke operan sebelah kiri.

```
a = 5 # adalah assignment
print("nilai a =",a)

a += 1 # artinya adalah a = a + 1
print("nilai a += 1, nilai a menjadi",a)

a -= 2 # artinya adalah a = a - 2
print("nilai a -= 2, nilai a menjadi",a)

a *= 5 # artinya adalah a = a * 5
print("nilai a *= 5, nilai a menjadi",a)

a /= 2 # artinya adalah a = a / 2
print("nilai a /= 2, nilai a menjadi",a)

# pangkat atau eksponen
a **= 3
print("nilai a **= 3, nilai a menjadi",a)
```

✓ 0.0s

```
nilai a = 5
nilai a += 1, nilai a menjadi 6
nilai a -= 2, nilai a menjadi 4
nilai a *= 5, nilai a menjadi 20
nilai a /= 2, nilai a menjadi 10.0
nilai a **= 3, nilai a menjadi 1000.0
```

```
b = 10
print("\nnilai b =",b)

# modulus dan floor division
b %= 3
print("nilai b %= 3, nilai b menjadi",b)

b = 10
print("\nnilai b =",b)

b //= 3
print("nilai b //= 3, nilai b menjadi",b)
```

✓ 0.0s

```
nilai b = 10
nilai b %= 3, nilai b menjadi 1

nilai b = 10
nilai b //= 3, nilai b menjadi 3
```

Gambar 4 Contoh Operator Penugasan

4. Logika atau Boolean

Operator Logika digunakan untuk membandingkan dua nilai yang bertipe Boolean dan akan menghasilkan nilai *Boolean* yaitu **TRUE** atau **FALSE**. Berikut beberapa jenis operatornya:

Tabel 4 Logika atau Boolean

Simbol	Keterangan
NOT	Menegasikan hasil. True menjadi False dan sebaliknya
OR	Jika salah satu True, maka hasilnya adalah True
AND	Jika dua buah nilai True, maka hasil True

```
# NOT
a = False
c = not a
print('data a =',a)
print('data c =',c)
```
✓ 0.0s
```
data a = False
data c = True
```

```
# OR
a = False
b = False
c = a or b
print(a,'OR',b,'=',c)
a = False
b = True
c = a or b
print(a,'OR',b,' =',c)
a = True
b = False
c = a or b
print(a,' OR',b,'=',c)
a = True
b = True
c = a or b
print(a,' OR',b,' =',c)
```
✓ 0.0s
```
False OR False = False
False OR True  = True
True  OR False = True
True  OR True  = True
```

```
# AND
a = False
b = False
c = a and b
print(a,'AND',b,'=',c)
a = False
b = True
c = a and b
print(a,'AND',b,' =',c)
a = True
b = False
c = a and b
print(a,' AND',b,'=',c)
a = True
b = True
c = a and b
print(a,' AND',b,' =',c)
```
✓ 0.0s
```
False AND False = False
False AND True  = False
True  AND False = False
True  AND True  = True
```

Gambar 6 Logika atau Boolean

5. Bitwise

Operator Bitwise hampir sama seperti Operator Logika, akan tetapi operator ini melakukan operasi berdasarkan bilangan bit/biner dengan angka 1 yang berarti **TRUE** dan angka 0 yang berarti **FALSE.**

Jika nilai asal yang dipakai bukan bilangan biner, akan dikonversi secara otomatis oleh Python menjadi bilangan biner. Operator Bitwise di antaranya:

Tabel 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)

true, true = false

```
a = 9
b = 5

# bitwise OR (|)
c = a | b
print('nilai :',a,' , binary :',format(a,'08b'))
print('nilai :',b,' , binary :',format(b,'08b'))
print('---------------------------- (|)')
print('nilai :',c,' , binary :',format(c,'08b'))
```

```
✓ 0.0s
nilai : 9  , binary : 00001001
nilai : 5  , binary : 00000101
---------------------------- (|)
nilai : 13  , binary : 00001101
```

```
# bitwise AND (&)
c = a & b
print('nilai :',a,' , binary :',format(a,'08b'))
print('nilai :',b,' , binary :',format(b,'08b'))
print('---------------------------- (&)')
print('nilai :',c,' , binary :',format(c,'08b'))
```

```
✓ 0.0s
nilai : 9  , binary : 00001001
nilai : 5  , binary : 00000101
---------------------------- (&)
nilai : 1  , binary : 00000001
```

```
# bitwise XOR (^)
c = a ^ b
print('nilai :',a,' , binary :',format(a,'08b'))
print('nilai :',b,' , binary :',format(b,'08b'))
print('---------------------------- (^)')
print('nilai :',c,' , binary :',format(c,'08b'))
```

```
✓ 0.0s
nilai : 9  , binary : 00001001
nilai : 5  , binary : 00000101
---------------------------- (^)
nilai : 12  , binary : 00001100
```

```
# bitwise NOT (~)
d = 10
c = ~d
print('nilai :',d,' , binary :',format(d,'08b'))
print('---------------------------- (~)')
print('nilai :',c,' , binary :',format(c,'08b'))
```

```
✓ 0.0s
nilai : 10  , binary : 00001010
---------------------------- (~)
nilai : -11  , binary : -0001011
```

```
# shift right (>>)
c = a >> 2
print('nilai :',a,' , binary :',format(a,'08b'))
print('---------------------------- (>>)')
print('nilai :',c,' , binary :',format(c,'08b'))
```

```
✓ 0.0s
nilai : 9  , binary : 00001001
---------------------------- (>>)
nilai : 2  , binary : 00000010
```

```
# shift Left (<<)
c = a << 2
print('nilai :',a,' , binary :',format(a,'08b'))
print('---------------------------- (<<)')
print('nilai :',c,' , binary :',format(c,'08b'))
```

```
✓ 0.0s
nilai : 9  , binary : 00001001
---------------------------- (<<)
nilai : 36  , binary : 00100100
```

Gambar 7 Contoh Bitwise

6. Keanggotaan

Operator Keanggotaan hanya bisa digunakan pada variable jenis *sequence* (**string, list, tuple, set dan *dictionary***) yang dapat menampung banyak nilai. Fungsi dari operator ini adalah untuk memeriksa apakah suatu nilai merupakan salah satu anggota dari variabel berjenis *sequence* atau tidak. Kemudian akan menghasilkan keluaran **TRUE** atau **FALSE**. Berikut beberapa Operator Keanggotaan:

Tabel 6 Penjelasan Keanggotaan

Operator	Penjelasan
in	Bernilai True jika nilai yang dicari ada di dalam *sequence*
not in	Bernilai True jika nilai yang dicari tidak ada dalam *sequence*

```
perusahaan = 'McKinsey'
empat_besar = ['Arsenal', 'City', 'MU', 'Tottenham'
mahasiswa = {
  'nama': 'POSI',
  'tahun': '2020'
}

print(
  "Huruf 'c' ada di variabel perusahaan?",
  'c' in perusahaan
)
```

✓ 0.0s

```
Huruf 'c' ada di variabel perusahaan? True
```

```
print(
  "Huruf 'z' tidak ada di variabel perusahaan?",
  'z' not in perusahaan
)
```

✓ 0.0s

```
Huruf 'z' tidak ada di variabel perusahaan? True
```

```
print(
  "Liverpool ada di empat besar klasemen?",
  'Liverpool' in empat_besar, "lah. Yakali ada"
)
```

✓ 0.0s

```
Liverpool ada di empat besar klasemen? False lah. Yakali ada
```

```
print(
  "Atribut 'nama' ada di dictionary mahasiswa?",
  'nama' in mahasiswa
)
```

✓ 0.0s

```
Atribut 'nama' ada di dictionary mahasiswa? True
```

Gambar 8 Contoh Keanggotaan

7. Identitas

Selain Operator Keanggotaan, Python masih memiliki operator spesial lainnya yaitu Operator Identitas. Operator ini didefinisikan dengan **is** dan **is not**. Tugasnya adalah untuk mengetahui apakah dua buah variabel merupakan objek yang sama (berhubungan dengan penempatan lokasi di memori) dan atau memiliki nilai yang sama /tidak. Jika sama akan menghasilkan nilai **TRUE** dan sebaliknya, jika berbeda akan menghasilkan nilai **FALSE**.

Tabel 7 Penjelasan Identitas

Operator	Penjelasan
is	Bernilai **TRUE** jika dua variabel bersifat identik baik dari segi nilai mau pun penempatan lokasi di memory
is not	Bernilai **FALSE** jika dua variabel tidak identik baik dari segi nilai mau pun penempatan lokasi di memory

```
x = 5
y = 6
print('nilai x =',x,', id =',hex(id(x)))
print('nilai y =',y,', id =',hex(id(y)))
hasil = x is y
print('x is y =',hasil)
```
✓ 0.0s
```
nilai x = 5 , id = 0x26818aa69b0
nilai y = 6 , id = 0x26818aa69d0
x is y = False
```

```
# 'is'
x = 5
y = 5
print('nilai x =',x,', id =',hex(id(x)))
print('nilai y =',y,', id =',hex(id(y)))
hasil = x is y
print('x is y =',hasil)
```
✓ 0.0s
```
nilai x = 5 , id = 0x26818aa69b0
nilai y = 5 , id = 0x26818aa69b0
x is y = True
```

```
x = 5
y = 6
print('nilai x =',x,',id = ',hex(id(x)))
print('nilai y =',y,',id = ',hex(id(y)))
hasil = x is not y
print('x is not y =',hasil)
```
✓ 0.0s
```
nilai x = 5 ,id =  0x26818aa69b0
nilai y = 6 ,id =  0x26818aa69d0
x is not y = True
```

```
# is not
x = 5
y = 5
print('nilai x =',x,', id =',hex(id(x)))
print('nilai y =',y,', id =',hex(id(y)))
hasil = x is not y
print('x is not y =',hasil)
```
✓ 0.0s
```
nilai x = 5 , id = 0x26818aa69b0
nilai y = 5 , id = 0x26818aa69b0
x is not y = False
```

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

print(x, hex(id(x)))
print(y, hex(id(y)))

print(x is z)
print(x is y)
print(x == y)
```
✓ 0.0s
```
['apple', 'banana'] 0x1dc35c7eec0
['apple', 'banana'] 0x1dc35c6f080
True
False
True
```

Gambar 9 Contoh Operasi Identitas

8. Operator String

a. Concat String

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

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

```
# string
str1 = "Hello"
str2 = "World"

# concat
result = str1 + " " + str2
print(result)
```

✓ 0.0s

```
Hello World
```

Gambar 10 Contoh Concat String

b. Menghitung Panjang String

Kita dapat menghitung panjang dari string yang ada dengan menggunakan operator (**len**).

Dalam contoh berikut, kita akan menghitung panjang dari string "Perancangan dan Optimasi Sistem Industri"

```
nama_lengkap = "Perancangan dan Optimasi Sistem Industri"
panjang = len(nama_lengkap)
print("panjang string " + nama_lengkap + " adalah ",(panjang))
```

✓ 0.0s

```
panjang string Perancangan dan Optimasi Sistem Industri adalah  40
```

Gambar 11 Contoh Panjang String

c. Mereplikasi String

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

```
# string
str = "HA"

# replicate
result = str * 3
print(result, "lucu bangetttt")
```

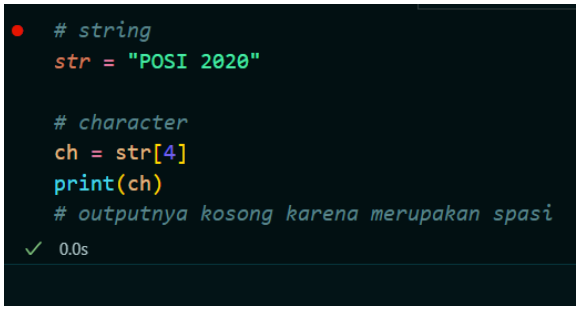
✓ 0.1s

```
HAHAHA lucu bangetttt
```

Gambar 11 Contoh Mereplikasi String

d. Mengakses Karakter dalam String

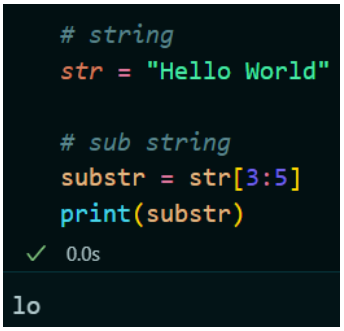
Kita menggunakan str [**i**] untuk mendapatkan karakter pada indeks i dalam string yang diberikan str. Indeks dimulai dari 0 jadi, karakter pertama di indeks 0, karakter kedua di 1 dan seterusnya. Dalam contoh berikut kita akan mengekstrak karakter kedua (indeks 1) dari string "POSI 2020".



Gambar 12 Contoh Mengakses karakter dalam string

e. 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.



Gambar 13 Contoh Substring

f. 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

Gambar 14 Contoh Skipping characters

g. Reverse String

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

Gambar 14 Contoh Reverse String

h. Escape Sequence

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

Tabel 8 Penjelasan Escape Sequence

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