Algoritma

Perkuliahan Algoritma Pemrograman pada Semester Ganjil 2020

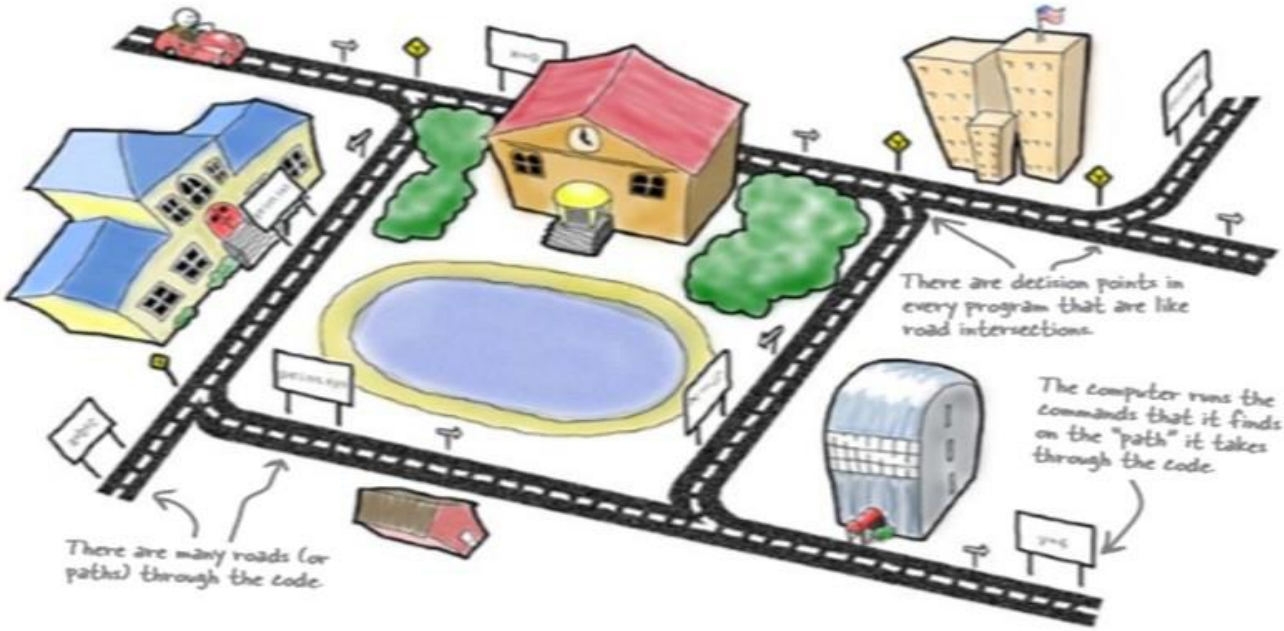
Terdapat tiga jenis algoritma :

1. *Sequence*
2. *Branches* (*Selection*)
3. *Iteration* (*Loop*)

Sequence

Contoh algoritma sebelumnya adalah algoritma sequence, yang dikerjakan berurutan dari awal sampai dengan akhir.

Akan tetapi, program tidak selamanya berbentuk sequence seperti yang sudah dikerjakan, akan tetapi lebih dari itu, seperti yang dilihat pada Gambar berikut :



Ekspressi Boolean

Ekspressi boolean adalah suatu ekspressi atau suatu kalimat yang memiliki dua buah nilai yang bertipe data **boolean**, yaitu **True** atau **False**.

Ekspressi boolean ini biasanya menggunakan *relational operator*, antara lain :

- x==y, *x is equal to y*
- x!=y, *x is not equal to y*
- x > y, *x is greater than y*
- x < y, *x is less than y*
- x>=y, *x is greater or equal to y*
- x<=y, *x is less than or equal to y*

Selain dengan menggunakan *relational operator*, ekspressi boolean ini juga dapat menggunakan *logical operator*, seperti :

- and
- or
- not

Cobalah syntax berikut di dalam python :

In [ ]:

```
x=5
y=6
print(x,y)
print(x==y)
x=y
print(x,y)
print(x==y)
```

In [ ]:

```
x=4
y=5
print(x)
print(y)
print(x==y)
x=y
print(x)
print(y)
```

In [ ]:

```
x==y
```

In [ ]:

```
5==4
```

In [ ]:

```
A=False
type(A)
```

In [ ]:

```
A=4>=3
print(A)
```

In [ ]:

```
B=False
print(B)
```

In [ ]:

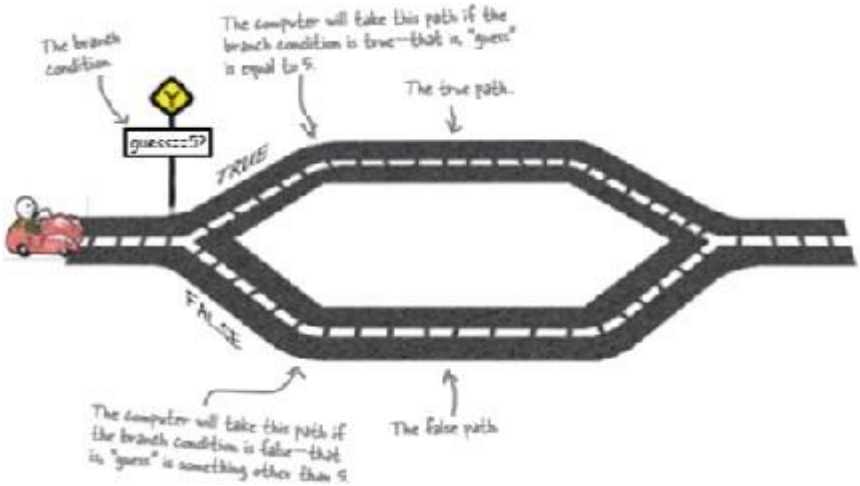
```
not(8>=9) and ('a'=='b')
```

In [ ]:

```
A=4
B=(A==0) or ((A%4)==0)
print(B)
```

Branches (Selection)

Adakalanya, suatu program terdapat pilihan, apakah mengerjakan bagian code A ataukah mengerjakan bagian yang lain?. Pilihan inilah yang disebut dengan **branches atau selection**. Bagian instruksi yang memuat branches ini adalah **branch condition**. Pada *branch condition* ini terdapat kondisi **True** atau **False**. Jika memenuhi kondisi **True** maka syntax pada percabangan **True** yang akan diekseskusi, begitu juga sebaliknya.



Problem 1 : Pencarian nilai terbesar dari dua buah angka

Input : dua buah angka (a,b)

Output : angka terbesar

Algoritma

1. Masukkan nilai a dan b
2. Jika a lebih besar dari b, maka hasil=a
3. Jika b lebih besar dari a, maka hasil=b

4. Tampilkan hasil

Implementasi

In [2]:

```
a=int(input('masukkan nilai a='))
b=int(input('masukkan nilai b='))
if a>b:
    #print('1')
    hasil=a

if a<=b:
    #print('2')
    hasil=b
    print('Nilai terbesar adalah=',hasil)
```

```
masukkan nilai a=27
masukkan nilai b=4
```

In [3]:

```
a=int(input('masukkan nilai a='))
b=int(input('masukkan nilai b='))
if a>b:
    hasil=a

else:
    hasil=b
    print('bilangan terbesar = ',hasil)
```

```
masukkan nilai a=7
masukkan nilai b=5
```

Problem 2 : Identifikasi jenis suatu bilangan, apakah bilangan genap ataukah ganjil

Input : angka

Output : Jenis bilangan

Algoritma

1.
2.
3.
4.

Problem 3 : Identifikasi tahun kabisat

Input : tahun

Output : tahun kabisat ataukah bukan tahun kabisat

Algoritma

1.
2.
3.
4.

Problem 4 : Tebak angka rahasia (10)

Input : angka

Output : Benar atau salah

Algoritma

1.
2.
3.
4.

Syntax if

if condition_is_True: statements

if condition_is_True: statements else: statements

if condition_is_true : statements elif condition_is_true : statements elif condition_is_true : statements elif condition_is_true : statements else: statements

Problem 5: Tebak angka rahasia (10)

Input : angka

Output :

- jika benar tulis 'right'
- jika lebih kecil dari 10, maka tulis 'higher please'
- jika lebih besar dari 10, maka tulis 'lower please'

Algoritma

1.
2.
3.
4.

In [ ]:

```
angka=10
data=int(input('Masukkan angka rahasia = '))
while data!=angka:

    if data==angka:
        print('Right')
    elif data>angka:
        print('Lower pls')
    else:
        print('higher pls')
    data=int(input('Masukkan angka rahasia = '))
```

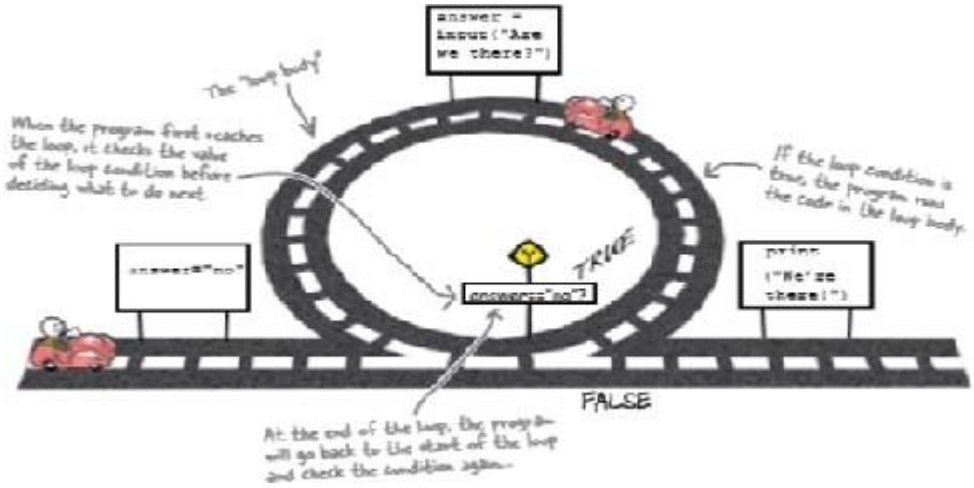
Iteration (Loop)

Pada beberapa permasalahan, terkadang dibutuhkan suatu perintah yang berulang kali, misalkan :

1. Pada contoh tebak angka rahasia, user hanya diberi kesempatan satu kali saja, jika terjadi kesalahan dalam menebak angka, maka user tidak dapat mengulangi proses kembali. Oleh karena itu dibutuhkan suatu algoritma untuk mengulang perintah yang sama.
2. Perhitungan deret matematika (aritmatika atau geometri)
3. dll

Perintah yang berulang seperti contoh diatas, tidak mungkin menuliskan perintah atau syntax secara manual (diulang secara manual), oleh karena itu bahasa pemrograman menyediakan iteration atau loop untuk menyelesaikan permasalahan ini.

Pada iterasi ini, syntax akan dieksekusi secara berulang-ulang selama kondisi bernilai **True**. Jika kondisi bernilai **False** maka proses iterasi akan berhenti.



syntax -while-

while condition_is_True:
statements

syntax -for-

for variable in range(number): statements

In [ ]:

```
for num in range(5):
    print(num)
```

Problem 1 : Deret aritmatika

Input : suku awal dan pembeda

Output : lima bilangan pada deret aritmatika

Algoritma

1. Masukkan suku awal
2. Masukkan pembeda
3. temp=suku awal
4. Lakukan perulangan berikut sampai dengan lima kali:
    D. tampilkan temp
    E. temp=temp+pembeda

Implementasi

In [ ]:

```
n=int(input('jumlah suku = '))
a=int(input('suku awal='))
b=int(input('pembeda='))
deret=a
for i  in range(n):
    print('suku  ke-',i+1,'=',deret)
    deret=deret+b
```

In [ ]:

```
a=int(input('suku awal='))
b=int(input('pembeda'))
temp=a
for i  in range(5):
    print('suku  ke-',i,'=',temp)
    temp=temp+i
```

Problem 2 : Bilangan Genap antara 0 sampai dengan 100

Input : -

Output : Bilangan genap antara 0 sampai dengan 100

Algoritma

1.
2.

3.
4.

In [ ]:

```
i=10
for a in range(101):
    if (i%2==0):
        print(i)
```

Problem 6: Tebak angka rahasia (10)

Input : angka

Output :

- jika benar tulis 'right'
- jika lebih kecil dari 10, maka tulis 'higher please'
- jika lebih besar dari 10, maka tulis 'lower please'
- Ulangi game ini sampai user menebak angka yang tepat

Algoritma

1.
2.
3.
4.

Problem 7: Hitung Faktorial suatu bilangan

Input : bilangan

Output : hasil faktorial

Algoritma

1.
2.
3.
4.