Week 2 Notes (Control Flow, List, Loops)

Control Flow

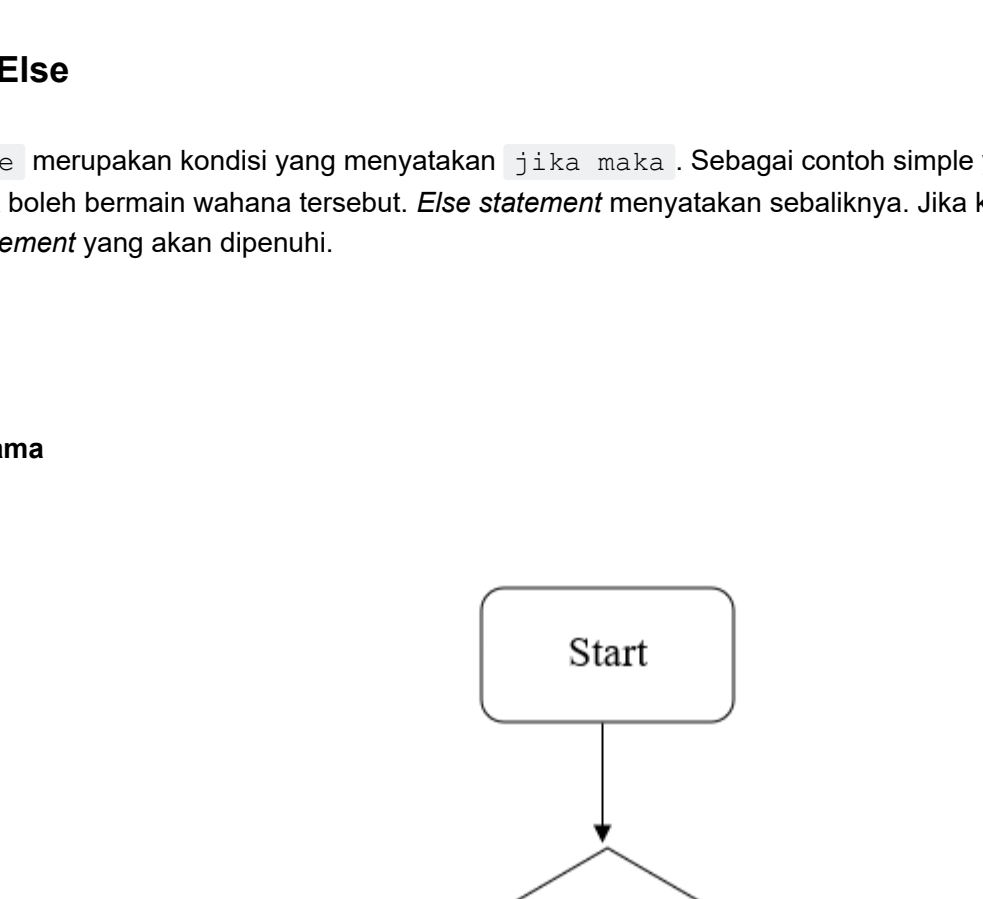
Fungsi dari control flow adalah untuk menentukan kondisi mana yang harus dieksekusi oleh komputer.

Control Flow: If Else

Control Flow `if else` merupakan kondisi yang menyatakan `jika maka`. Sebagai contoh simple yaitu *jika (if statement)* tinggi tidak mencukupi maka tidak boleh bermain wahana tersebut. *Else statement* menyatakan sebaliknya. Jika kondisi *if statement* tidak terpenuhi, maka kondisi *else statement* yang akan dipenuhi.

Contoh:

Diagram Kasus Pertama

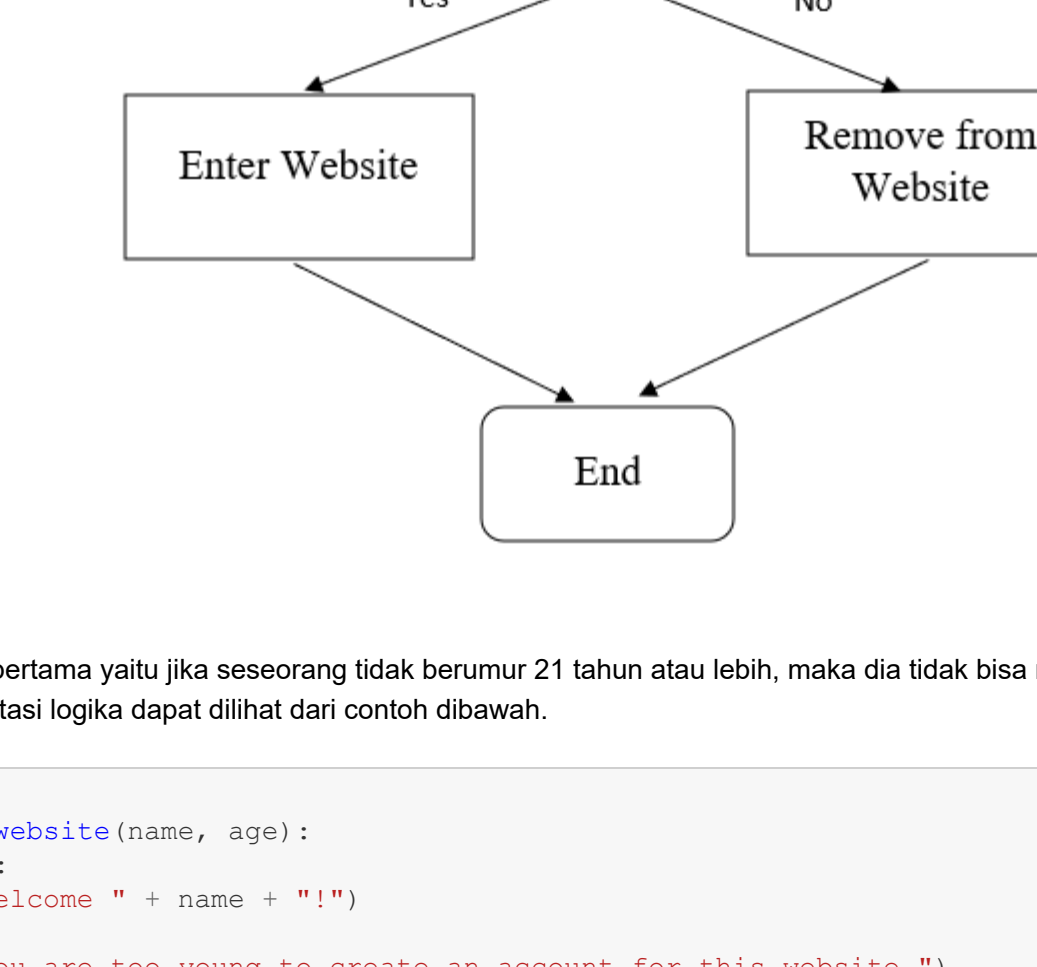


Disini terdapat kasus pertama yaitu jika seseorang tidak berumur 21 tahun atau lebih, maka dia tidak bisa memasuki website untuk membuat kartu kredit. Implementasi logika dapat dilihat dari contoh dibawah.

In [1]:
```python
# Contoh 1
def credit_card_website(name, age):
    if age >= 21:
        print("Welcome " + name + "!")
    else:
        print("You are too young to create an account for this website.")
        years_left = 21 - age
        print("Wait another " + str(years_left) + " year(s).")

credit_card_website("Randy", 21) # output: Welcome Randy! -> karena age > 18
credit_card_website("Susan", 17) # output: You are too young to create an account for this website. \n
# Wait another x year(s) -> karena age < 18
```

Diagram Kasus Kedua



Disini terdapat kasus kedua dimana seorang remaja harus berumur 17 tahun agar dapat memasuki bioskop sendiri. Jika anak remaja dibawah umur 17 tahun dan tidak bersama dengan orang tua, maka dia tidak bisa memasuki ruangan bioskop. Tetapi jika bersama orang tua, dia bisa memasuki bioskop. Implementasi logika dalam programming dapat dilihat dari contoh dibawah.

In [2]:
```python
# Contoh 2
def can_watch_movie(age, with_parents):
    if age >= 17:
        return True
    else:
        if with_parents:
            return True
        else:
            return False

print(can_watch_movie(18, False))
print(can_watch_movie(16, False))
print(can_watch_movie(15, True))
```

```
True
False
True
```

If Else Continuation: Elif (else if) Statement

Terdapat satu statement lagi di control flow `if else statement`, yaitu `elif` (else if). `Elif` merupakan sebuah instruksi yang diberikan kepada komputer untuk memeriksa apakah terdapat kondisi lain selain `if statement`. Jika `if statement` tidak terpenuhi, maka program akan berahlih ke kondisi elif. Jika kondisi elif dan if tidak terpenuhi, barulah else statement akan dieksekusi. Kita akan mengambil contoh lagi dari kasus kedua.

In [3]:
```python
# Contoh 3
def can_watch_movie(age, with_parents):
    if age >= 17:
        return True
    elif age < 17 and with_parents == True:
        return True
    else:
        return False

print(can_watch_movie(18, False))
print(can_watch_movie(16, False))
print(can_watch_movie(2, True))
```

```
True
False
True
```

Contoh 2 dan Contoh 3 mempunyai logika yang sama, tetapi hanya cara penulisannya saja yang beda. Contoh kedua menggunakan apa yang disebut `nested if else statement` (nested artinya terdapat statement yang sama di dalam statement tersebut). Contoh 3 menggunakan `elif`. Apakah ada logika lain yang lebih pendek untuk menyelesaikan masalah pada contoh 2 dan 3? Ada. Lihat contoh 4.

In [4]:
```python
# Contoh 4
def can_watch_movie(age, with_parents):
    if age >= 17 or with_parents == True:
        return True
    else:
        return False

print(can_watch_movie(18, False))
print(can_watch_movie(16, False))
print(can_watch_movie(2, True))
```

```
True
False
True
```

Control Flow: While loop

`While loop` disini juga merupakan kondisi yang sering digunakan dalam pemrograman. While loop menyatakan `selagi` kondisi masih terpenuhi, maka dia akan dieksekusi terus menerus.

Diagram Kasus

Mari kita masuk ke contoh kasus memakan apel. Kondisi yang kita miliki yaitu selagi masih terdapat apel, maka seseorang akan memakannya dan mengucapkan terima kasih. Implementasi logika dapat dilihat dari contoh dibawah.

In [5]:
```python
def eat_apples(num_of_apples):
    apples_remaining = num_of_apples

    while apples_remaining > 0:
        apples_remaining -= 1
        print("Thank you!")

    print("Done")

eat_apples(5)
```

```
Thank you!
Thank you!
Thank you!
Thank you!
Thank you!
Done
```

Control Flow: Try Exception

`try except` merupakan suatu control flow di python yang digunakan untuk memberitahu error yang terjadi kepada user (pengguna). Sebagai contoh, program tidak bisa mengkalkulasi sesuatu yang tidak terbatas jumlahnya yaitu `5 / 0` sehingga akan menghasilkan error yang tidak dimengerti oleh pengguna. Alangkah baiknya jika kita dapat memberitahu pengguna dengan menggunakan bahasa manusia yaitu dengan bantuan control flow `try except`.

Contoh:

In [6]:
```python
# Program tanpa try except
# print(5 / 0) <- dikomen karena akan menghasilkan error (coba sendiri)
```

In [7]:
```python
# Program dengan try except
try:
    print(5 / 0)
except:
    print("Program akan error jika integer dibagi dengan nol.")
```

```
Program akan error jika integer dibagi dengan nol.
```

Lists (Data Structure)

Kegunaan List

Untuk menyimpan `n` nilai. `n` disini berarti jumlah nilai tersebut lebih dari satu.

Cara Mendeklarasikan List

`nama_list = list()` atau `nama_list = []`. List yang tidak bernilai biasa disebut sebagai `empty list`.

Note: `nama_list = list()` hanya berlaku untuk mendeklarasikan empty list.

Contoh:

In [8]:
```python
my_list = [] # empty list
lst = list() # empty list

print(my_list) # []
print(lst) # []
```

```
[]
[]
```

In [9]:
```python
# Menyimpan nilai lebih dari satu, pisahkan dengan koma
my_list = [1, 2, 3, 4, 5, 6]

print(type(my_list)) # untuk mengetahui tipe data
print(my_list) # [1, 2, 3, 4, 5, 6]
```

```
<class 'list'>
[1, 2, 3, 4, 5, 6]
```

Menghitung Panjang Sebuah List

Kita dapat mengetahui panjang sebuah list dengan kata kunci `len()`. `len()` disini berarti length dalam bahasa Inggris.

Contoh:

In [10]:
```python
my_list = [1, 2, 3, 4, 5, 6]

print(len(my_list)) # 6
```

```
6
```

Mengetahui Nilai Pada Indeks ke - n pada Sebuah List

List tentunya mempunyai apa yang kita namakan indexing. Indexing berfungsi untuk mengetahui nilai pada posisi ke - n suatu list. **Ingat, indexing di dalam programming dimulai dari 0**.

Berikut adalah cara penindeksan di dalam programming:

In [11]:
```python
my_list = ["a", 1, 5, 6, "b", True, False]
```

In [12]:
```python
# Dari list diatas

from prettytable import PrettyTable

table = PrettyTable()

table.field_names = ["Nilai", "Indeks ke -"]
table.add_rows([["a", "0"],
               ["1", "1"],
               ["5", "2"],
               ["6", "3"],
               ["b", "4"],
               ["True", "5"],
               ["False", "6"]])

print(table)
```

```
+-------+-------------+
| Nilai | Indeks ke - |
+-------+-------------+
|   a   |      0      |
|   1   |      1      |
|   5   |      2      |
|   6   |      3      |
|   b   |      4      |
|  True |      5      |
| False |      6      |
+-------+-------------+
```

Jika kita ingin mengakses nilai `b` pada list diatas, maka kita dapat melakukannya dengan `nama_list[indeks]`

In [13]:
```python
print(my_list[4])
```

```
b
```

Jika ingin mengakses nilai `5` ?

In [14]:
```python
print(my_list[2])
```

```
5
```

Negative Indexing

Indexing tidak hanya dilakukan dengan angka bulat positif, tetapi dapat dilakukan juga dengan angka bulat negatif. Apa perbedaannya? *positive indexing dimulai dari depan, sedangkan negative indexing dimulai dari belakang*.

In [15]:
```python
table = PrettyTable()

table.field_names = ["Nilai", "Negative Indexing"]
table.add_rows([["a", "-7"],
               ["1", "-6"],
               ["5", "-5"],
               ["6", "-4"],
               ["b", "-3"],
               ["True", "-2"],
               ["False", "-1"]])

print(table)
```

```
+-------+-------------------+
| Nilai | Negative Indexing |
+-------+-------------------+
|   a   |         -7        |
|   1   |         -6        |
|   5   |         -5        |
|   6   |         -4        |
|   b   |         -3        |
|  True |         -2        |
| False |         -1        |
+-------+-------------------+
```

In [16]:
```python
print(my_list[-1])
print(my_list[-5])
print(my_list[-7])
```

```
False
5
a
```

Slicing a List

Slicing disini berarti mengakses nilai list hanya pada interval yang ditentukan. Cara melakukan slicing pada list yaitu dengan `nama_list[starting_index:ending_index + 1]`. Jika kita tidak menentukan `starting index` ataupun `ending index`, maka Python akan menganggap bahwa kita mengambil semua indeks setelah indeks yang ditentukan.

Contohnya jika ingin mengakses nilai list pada indeks ke 1 sampai ke 3, maka kita akan mengetikkannya dengan:

In [17]:
```python
print(my_list[1:4])
```

```
[1, 5, 6]
```

In [18]:
```python
print(my_list[-3:])
```

```
['b', True, False]
```

In [19]:
```python
print(my_list[2:])
```

```
[5, 6, 'b', True, False]
```

In [20]:
```python
print(my_list[:3])
```

```
['a', 1, 5]
```

Documentation and Advise

Dalam belajar coding ataupun di dunia nyata, sering kali kita akan menemui suatu masalah tertentu dan sangkut pada masalah tersebut. Disini saya **secara sengaja** tidak mengajarkan semua bahan mengenai list kepada kalian karena saya ingin mengasah kemampuan problem solving masing - masing dari kalian. **Apa yang akan kalian lakukan adalah membaca dokumentasi mengenai list**. Kalian hanya perlu membaca bagian `append()`, `insert()`, `remove()`, `pop()`, `count()`, dan `reverse()`.

List documentation: https://docs.python.org/3/tutorial/datastructures.html

List Continuation: Iterating through a List

In [21]:
```python
# Soal 1
# Buatlah / tuliskan function dimana function akan mengoutput semua
# element yang berada di dalam list
def print_list(lst):
    i = 0
    while i < len(lst):
        print(lst[i])
        i += 1
    print("done")

print_list([1, 2, 3, 4])
```

```
1
2
3
4
done
```

In [22]:
```python
# Soal 2
# Buatlah / tuliskan function yang mengoutput list dari angka
# Misalkan jika kita memiliki angka 1, 2, 3, 4 maka semua akan dioutput dalam
# bentuk list [1, 2, 3, 4]
def list_numbers(n):
    i = 0
    new_list = []

    while i < n:
        i += 1
        new_list.append(i)

    return new_list

list_numbers(5)
```

Out[22]:
```
[1, 2, 3, 4, 5]
```

List Continuation: Modifying Values in a List

Kita bisa mengganti nilai di dalam list dengan cara: `nama_list[indeks] = nilai`

In [23]:
```python
# Misalkan kita mempunyai sebuah list
a = [1, 2, 3, 4]

# Kita mengganti nilai pada indeks ke 1
a[1] = 10

# Maka pada indeks ke 1 nilainya akan berubah menjadi 10
print(a) # [1, 10, 3, 4]
```

```
[1, 10, 3, 4]
```

In [24]:
```python
# Contoh Soal
# Misalkan ada sebuah list. Kita disuruh menuliskan fungsi untuk
# mengoutput hasil kuadrat dari masing - masing nilai di dalam list.
# Jika input list [1, 3, 5], maka output dari list tersebut adalah
# [1, 9, 25]

def list_squared(lst):
    new_list = []
    i = 0

    while i < len(lst):
        a = lst[i] ** 2
        new_list.append(a) # cara cepat: new_list.append(lst[i] ** 2)
        i += 1

    return new_list

print(list_squared([1, 3, 5]))
```

```
[1, 9, 25]
```

Addition to List

Selain mengganti nilai, kita juga bisa menggabungkan list dengan simbol `+` sama seperti `string`

In [25]:
```python
# Contoh
a = [1, 3, 5, 7]
print(a + [1, 2, 3]) # [1, 3, 5, 7, 1, 2, 3]
```

```
[1, 3, 5, 7, 1, 2, 3]
```

In [26]:
```python
a = [1, 3, 5, 7]
b = [2, 4, 6, 8]
print(a + b) # [1, 3, 5, 7, 2, 4, 6, 8]
```

```
[1, 3, 5, 7, 2, 4, 6, 8]
```

Loops: For Loops

Fungsi `for loops` sama seperti `while loop`, hanya saja berbeda cara penulisan

In [27]:
```python
# Misalkan kita mempunyai sebuah list
flavors = ["vanilla", "strawberry", "chocolate", "mint", "banana", "rubber"]

# Dengan cara while loop
i = 0
while i < len(flavors):
    print(flavors[i])
    i += 1
```

```
vanilla
strawberry
chocolate
mint
banana
rubber
```

Cara menulis for loop: `for [item] in [list]: [action]`

In [28]:
```python
# Dengan cara for loop
for flavor in flavors:
    print(flavor)
```

```
vanilla
strawberry
chocolate
mint
banana
rubber
```

Range

`range([start], [end], [steps])` merupakan sebuah command spesial di dalam bahasa pemrograman Python karena `range()` membantu mempermudah pekerjaan kita. Jika kita ingin menuliskan sebuah list secara berurutan dari 1 - 20, kita hanya perlu melakukannya dengan mengetikkan `list(range(1, 21))`, maka terbentuklah list yang berisi nilai 1 - 20. Kita tidak perlu lagi menuliskannya secara manual.

In [29]:
```python
# Contoh
my_list = list(range(1, 21))
print(my_list)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

Kita juga bisa menggunakan `range()` dalam for loop.

In [30]:
```python
# Contoh
for i in range(4):
    print("hello")
```

```
hello
hello
hello
hello
```

Loops Continuation: Break and Continue

`break` berfungsi untuk menghentikan program ketika kondisi telah terpenuhi di dalam `for loop`

In [31]:
```python
# Contoh
flavors = ["vanilla", "strawberry", "chocolate", "mint", "banana", "rubber"]

for flavor in flavors:
    print(flavor)
    if flavor == "mint":
        break
```

```
vanilla
strawberry
chocolate
mint
```

`continue` berfungsi untuk mengabaikan sesuatu

In [32]:
```python
# Contoh
for flavor in flavors:
    if flavor == "mint":
        continue
    print(flavor)
```

```
vanilla
strawberry
chocolate
banana
rubber
```

In [33]:
```python
# Contoh Soal
# Ketika di restoran, maka kita akan memberikan tips kepada pegawai restoran.
# Buatlah sebuah program dimana kita akan mengiterasi sebuah list yang berisi nilai mata
# uang rupiah. Jika tips yang diberikan bernilai diatas atau sama dengan Rp. 5.000, maka pegawai
# akan mengucapkan "Terima kasih atas tips yang bernilai X Rupiah" (kalimat tersebut harus dioutput
# ke console) dan output juga total tips yang didapat di akhir "Total tips: Rp X"

def receive_tips(tips):
    total = 0
    for tip in tips:
        if tip < 5000:
            total += tip
            print(tip)
        else:
            total += tip
            print("Terima kasih atas tips yang bernilai Rp. " + str(tip))

    print("Total tips: Rp. " + str(total))

receive_tips([5000, 10000, 25000, 2000])

# Untuk sedikit tantangan cobalah untuk tidak mengouput tips dibawah 5000 ke consol
# tetapi tetap hitung tips tersebut dan tambahkan ke total (hint: gunakan continue (hanya perlu
# mengganti 1 baris dari kode yang telah ada))
```

```
Terima kasih atas tips yang bernilai Rp. 5000
Terima kasih atas tips yang bernilai Rp. 10000
Terima kasih atas tips yang bernilai Rp. 25000
2000
Total tips: Rp. 42000
```

Loops Continuation: Nested Loops

Apa yang dimaksud dengan `nested loops`? `Nested Loops` berarti ada loop di dalam loop.

In [34]:
```python
# Contoh
for i in [1, 2, 3]:
    for j in [4, 5, 6]:
        print(str(i) + str(j))
```

```
14
15
16
24
25
26
34
35
36
```

In [35]:
```python
# Contoh soal
# Tuliskan fungsi untuk mengoutput sebuah segitiga siku - siku.
"""
segitiga(5)
Output:
*
**
***
****
*****

segitiga(3)
Output:
*
**
***
"""

def segitiga(n):
    for i in range(n):
        line_print = "*"
        for j in range(i):
            line_print += "*"
        print(line_print)

segitiga(5)
print("-" * 45)
segitiga(3)
```

```
*
**
***
****
*****
---------------------------------------------
*
**
***
```

Matriks: List of Lists

In [36]:
```python
r1 = [1, 2, 3]
r2 = [4, 5, 6]
r3 = [7, 8, 9]

matrix = [r1, r2, r3]
print(matrix)
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

In [37]:
```python
# Dari contoh matriks di atas kita ingin mengoutput list of lists dalam bentuk matriks yang sempurna
def print_matrix(m):
    for row in m:
        line_to_print = ""
        for col in row:
            line_to_print += str(col) + " "
        print(line_to_print)

print_matrix(matrix)
```

```
1 2 3
4 5 6
7 8 9
```

Week 2 Programming Assessment: https://github.com/winstencoellins/progamming-fundamentals-week2