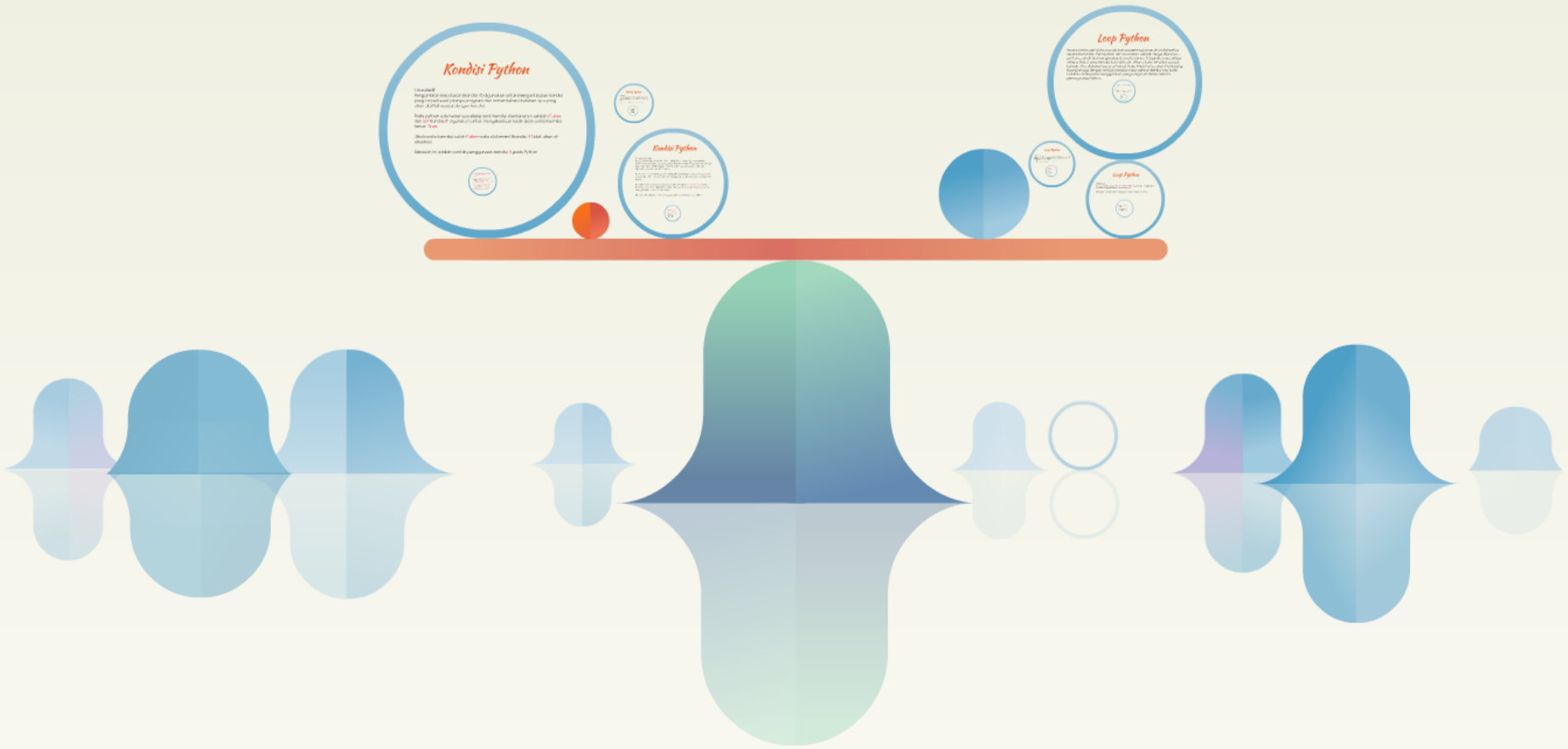
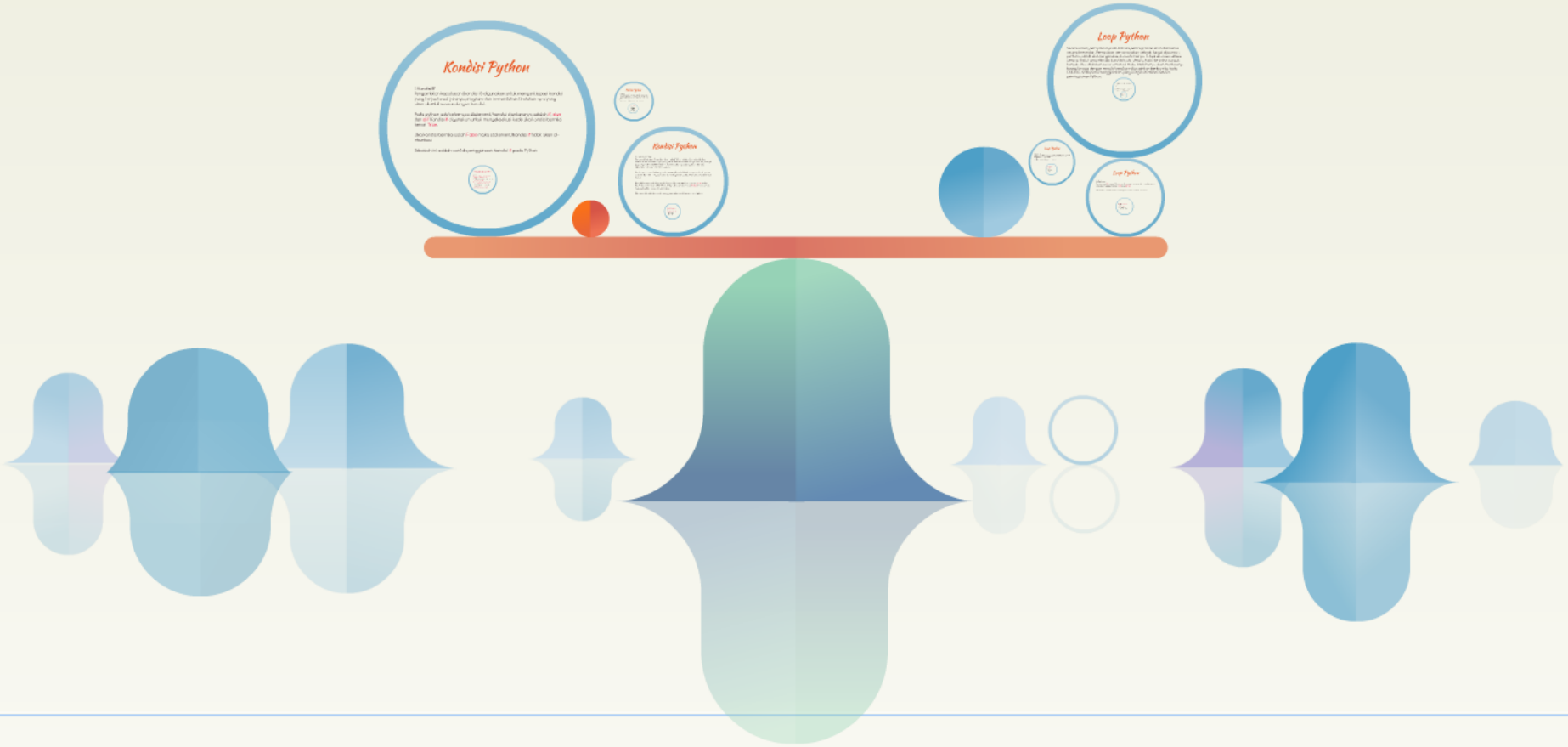
2. Pemrograman Berorientasi Objek

Kondisi dan Loop Case in python



2. Pemrograman Berorientasi Objek

Kondisi dan Loop Case in python



Kondisi Python

1. Kondisi If

Pengambilan keputusan (kondisi if) digunakan untuk mengantisipasi kondisi yang terjadi saat jalanya program dan menentukan tindakan apa yang akan diambil sesuai dengan kondisi.

Pada python ada beberapa statement/kondisi diantaranya adalah if, else dan elif Kondisi if digunakan untuk mengeksekusi kode jika kondisi bernilai benar True.

Jika kondisi bernilai salah False maka statement/kondisi if tidak akan di-eksekusi.

Dibawah ini adalah contoh penggunaan kondisi if pada Python

```
#Kondisi if adalah kondisi yang akan dieksekusi
oleh program jika bernilai benar atau TRUE

nilai = 9

#jika kondisi benar/TRUE maka program akan
mengeksekusi perintah dibawahnya
if(nilai > 7):
    print("Sembilan Lebih Besar Dari Angka Tujuh ")
# Kondisi Benar, Dieksekusi

#jika kondisi salah/FALSE maka program tidak
akan mengeksekusi perintah dibawahnya
if(nilai > 10):
      print("Sembilan Lebih Besar Dari Angka
      Sepuluh")
# Kondisi Salah, Maka tidak tereksekusi
```

Kondisi Python

2. Kondisi If Else

Pengambilan keputusan (kondisi if else) tidak hanya digunakan untuk menentukan tindakan apa yang akan diambil sesuai dengan kondisi, tetapi juga digunakan untuk menentukan tindakan apa yang akan diambil/ dijalankan jika kondisi tidak sesuai.

Pada python ada beberapa statement/kondisi diantaranya adalah if, else dan elif Kondisi if digunakan untuk mengeksekusi kode jika kondisi bernilai benar.

Kondisi if else adalah kondisi dimana jika pernyataan benar True maka kode dalam if akan dieksekusi, tetapi jika bernilai salah False maka akan mengeksekusi kode di dalam else.

Dibawah ini adalah contoh penggunaan kondisi if else pada Python

#Kondisi if else adalah jika kondisi bernilai TRUE maka akan dieksekusi pada if, tetapi jika bernilai FALSE maka akan dieksekusi kode pada else

```
nilai = 3

#Jika pernyataan pada if bernilai TRUE maka if akan dieksekusi, tetapi jika FALSE kode pada else yang akan dieksekusi.
if(nilai > 7):
    print("Selamat Anda Benar")
else:
    print("Maaf Anda Tidak Benar")
```

Kondisi Python

3. Kondisi Elif

Pengambilan keputusan (kondisi if elif) merupakan lanjutan/percabangan logika dari "kondisi if". Dengan elif kita bisa membuat kode program yang akan menyeleksi beberapa kemungkinan yang bisa terjadi. Hampir sama dengan kondisi "else", bedanya kondisi "elif" bisa banyak dan tidak hanya satu.

Dibawah ini adalah contoh penggunaan kondisi elif pada Python

```
#Contoh penggunaan kondisi elif

hari_ini = "Minggu"

if(hari_ini == "Senin"):
    print("Saya akan kuliah")
elif(hari_ini == "Selasa"):
    print("Saya akan kuliah")
elif(hari_ini == "Rabu"):
    print("Saya akan kuliah")
elif(hari_ini == "Kamis"):
    print("Saya akan kuliah")
elif(hari_ini == "Jumat"):
    print("Saya akan kuliah")
elif(hari_ini == "Sabtu"):
    print("Saya akan kuliah")
elif(hari_ini == "Minggu"):
    print("Saya akan libur")
```

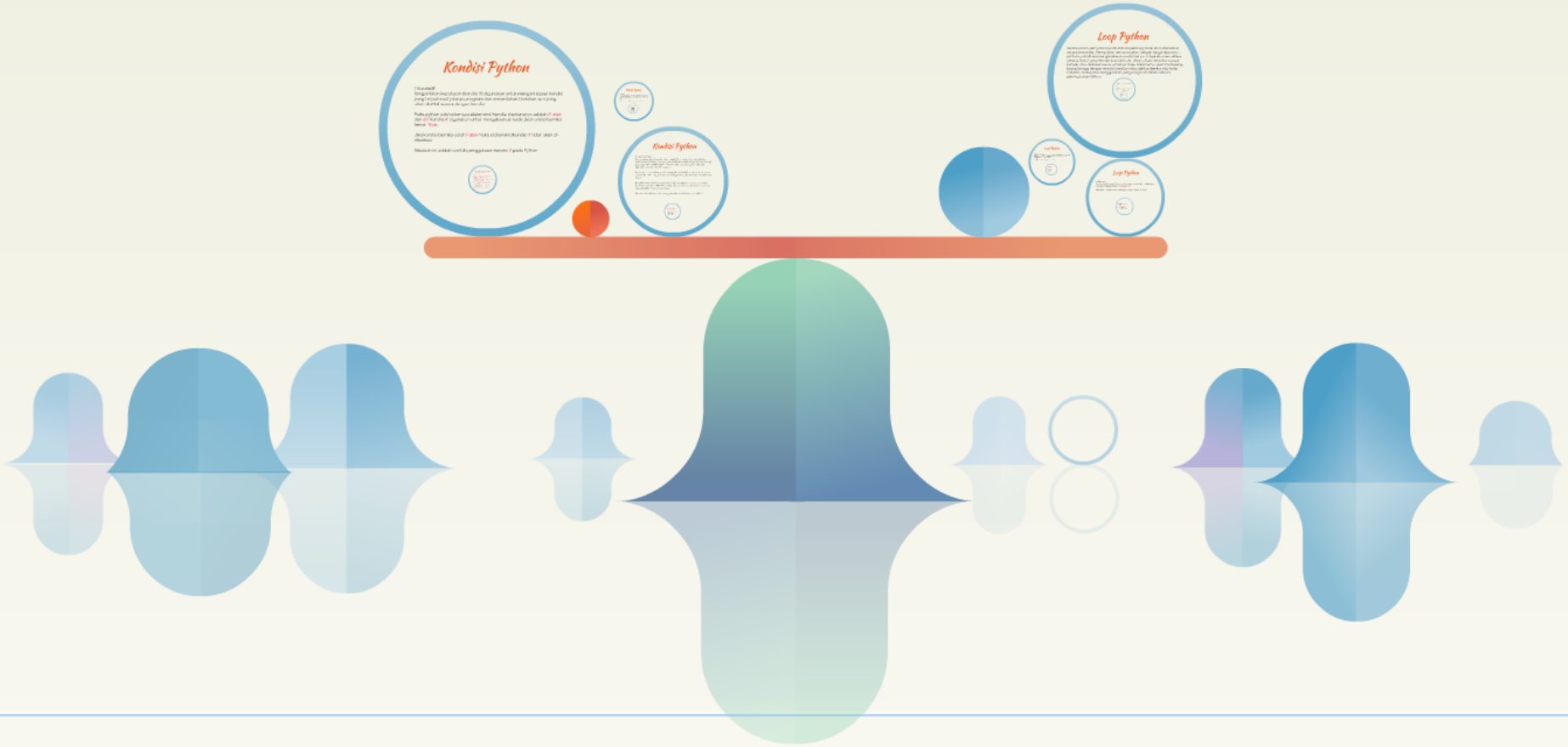
```
#Contoh penggunaan kondisi elif

hari_ini = "Minggu"

if(hari_ini == "Senin"):
    print("Saya akan kuliah")
elif(hari_ini == "Selasa"):
    print("Saya akan kuliah")
elif(hari_ini == "Rabu"):
    print("Saya akan kuliah")
elif(hari_ini == "Kamis"):
    print("Saya akan kuliah")
elif(hari_ini == "Jumat"):
    print("Saya akan kuliah")
elif(hari_ini == "Sabtu"):
    print("Saya akan kuliah")
elif(hari_ini == "Minggu"):
    print("Saya akan libur")
```

2. Pemrograman Berorientasi Objek

Kondisi dan Loop Case in python



Loop Python

Secara umum, pernyataan pada bahasa pemrograman akan dieksekusi secara berurutan. Pernyataan pertama dalam sebuah fungsi dijalankan pertama, diikuti oleh yang kedua, dan seterusnya. Tetapi akan ada situasi dimana Anda harus menulis banyak kode, dimana kode tersebut sangat banyak. Jika dilakukan secara manual maka Anda hanya akan membuang-buang tenaga dengan menulis beratus-ratus bahkan beribu-ribu kode. Untuk itu Anda perlu menggunakan pengulangan di dalam bahasa pemrograman Python.

Loop Python

Python memungkinkan penggunaan satu lingkaran di dalam loop lain. Bagian berikut menunjukkan beberapa contoh untuk menggambarkan konsep tersebut.

Dibawah ini adalah contoh penggunaan Nested Loop.

Di dalam bahasa pemrograman Python pengulangan dibagi menjadi 3 bagian, yaitu :

1. While Loop
2. For Loop
3. Nested Loop

1. While Loop

Pengulangan While Loop di dalam bahasa pemrograman Python dieksesusi statement berkali-kali selama kondisi bernilai benar atau True.

Dibawah ini adalah contoh penggunaan pengulangan While Loop

```
#Contoh penggunaan While Loop
#Catatan: Penentuan ruang lingkup di Python bisa
menggunakan tab alih-alih menggunakan tanda kurung

berhitung = 0
while (berhitung < 9):
    print ("Hitungan: ", berhitung)
    berhitung = berhitung + 1

print ("Program Selesai!")
```

```
#Contoh penggunaan While Loop
#Catatan: Penentuan ruang lingkup di Python bisa
menggunakan tab alih-alih menggunakan tanda kurung

berhitung = 0
while (berhitung < 9):
    print ("Hitungan: ", berhitung)
    berhitung = berhitung + 1

print ("Program Selesai!")
```

Loop Python

2. For Loop

Pengulangan for pada Python memiliki kemampuan untuk mengulangi item dari urutan apapun, seperti list atau string.

Dibawah ini adalah contoh penggunaan pengulangan For Loop.

```
#Contoh pengulangan for sederhana
angka = [1,2,3,4,5]
for x in angka:
    print(x)

#Contoh pengulangan for
buah = ["nanas", "apel", "jeruk"]
for makanan in buah:
    print ("Saya suka makan", makanan)
```

```
#Contoh pengulangan for sederhana
angka = [1,2,3,4,5]
for x in angka:
    print(x)

#Contoh pengulangan for
buah = ["nanas", "apel", "jeruk"]
for makanan in buah:
    print ("Saya suka makan", makanan)
```

Loop Python

2. Nested Loop

Bahasa pemrograman Python memungkinkan penggunaan satu lingkaran di dalam loop lain. Bagian berikut menunjukkan beberapa contoh untuk menggambarkan konsep tersebut.

Dibawah ini adalah contoh penggunaan Nested Loop.

```
#Contoh penggunaan Nested Loop
#Catatan: Penggunaan modulo pada kondisional
mengasumsikan nilai selain nol sebagai True(benar) dan
nol sebagai False(salah)

i = 2
while(i < 100):
   j = 2
   while(j <= (i/j)):
      if not(i%j): break
      j = j + 1
   if (j > i/j) : print(i, " is prime")
   i = i + 1

print("Good bye!")
```

#Contoh penggunaan Nested Loop
#Catatan: Penggunaan modulo pada kondisional mengasumsikan nilai selain nol sebagai True(benar) dan nol sebagai False(salah)

```
i = 2
while(i < 100):
    j = 2
    while(j <= (i/j)):
        if not(i%j): break
        j = j + 1
    if (j > i/j) : print(i, " is prime")
    i = i + 1

print("Good bye!")
```

2. Pemrograman Berorientasi Objek

Kondisi dan Loop Case in python