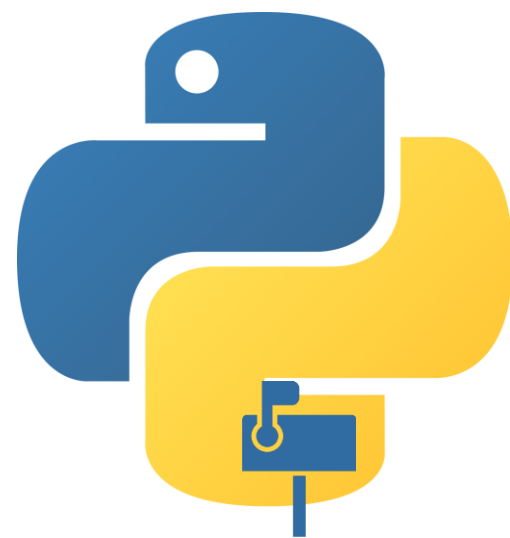


PYTHON

- TEMEL KAVRAMLAR -





PROGRAMLAMA DİLLERİ

- Programcıların bilgisayarla iletişim kurmasını sağlayan
- Bilgisayara neyi nasıl yapacağını bildiren
- Belirli görevleri olan komutlardır.
- Kod, bilgisayarlara ne yapacaklarını liste halinde yazmak gibi düşünülebilir.
- Bu yapılacaklar listesi, son derece ayrıntılı ve bazı mantıklarla yazılmalıdır.

KULLANIM AMACINA GÖRE PROGRAMLAMA DİLLERİ



1-) Uygulamalar ve program geliştirme

Uygulama ve program geliştirme, günlük olarak çalıştığınız programları içerir. Örneğin, bu web sayfasını görüntülemek için kullandığınız İnternet tarayıcısı bir program olarak kabul edilir. Bir program geliştirmek istiyorsanız, aşağıdaki dilleri göz atabilirsiniz.

- C
- C#
- C++
- Java
- Visual Basic

KULLANIM AMACINA GÖRE PROGRAMLAMA DİLLERİ



2-) Yapay zeka geliştirme

Yapay zeka veya ilgili alanlar, bilgisayar oyunlarında karakter etkileşimlerinin, karar veren programların bölümlerinin, sohbet botlarının ve daha fazlasının oluşturulmasını içerir. Bir yapay zeka geliştirmek istiyorsanız, aşağıdaki dilleri göz önünde bulundurmalısınız:

- Matlab
- R
- C#
- C
- Prolog
- Python

KULLANIM AMACINA GÖRE PROGRAMLAMA DİLLERİ



3-) Veritabanı geliştirme

Veritabanı geliştiricileri veritabanları oluşturur ve sürdürür. Bir veritabanı oluşturmak veya bakımını yapmakla ilgileniyorsanız, aşağıdaki dillerden birini göz önünde bulundurmalısınız:

- MySQL
- SQL
- Visual FoxPro
- MongoDB



KULLANIM AMACINA GÖRE PROGRAMLAMA DİLLERİ



4-) Oyun geliştirme

Oyun geliştirme, bilgisayar oyunları veya diğer eğlence yazılımlarını oluşturmayı içerir. Bir oyun geliştirmek istiyorsanız, aşağıdaki dilleri göz önünde bulundurmalısınız:

- C
- C#
- C++
- DarkBASIC
- Java
- Javascript

KULLANIM AMACINA GÖRE PROGRAMLAMA DİLLERİ



5-) Bilgisayar sürücüleri veya diğer donanım geliştirme

İnternet ve web sayfası geliştirme internetin özüdür. Geliştiriciler olmasaydı İnternet olmazdı. Web sayfaları, İnternet uygulamaları veya İnternet ile ilgili diğer görevleri oluşturmakla ilgileniyorsanız, aşağıdaki dilleri göz önünde bulundurmalısınız:

- PHP
- HTML
- Java
- Javascript
- Perl
- Python
- XML



PROGRAMLAMA DİLLERİ SYNTAX (SÖZ DİZİMİ)

- **Syntax, dilimizde** olduğu gibi cümlelerin anlaşılabilir ve okunaklı olabilmesi için nasıl dil kuralları bulunuyorsa yazılım dillerinde de aynı işlevi görerek yazdığımız programın bilgisayar tarafından okunabilmesine olanak tanır. Bir programın kurallarını ve sınırlarını belirlememizde yardımcı olur.
- **Syntax** olmasaydı, bir yazılımda anlatmak istediklerimizi tam olarak anlatamaz çalıştırmak istediğimiz şeyleri doğru bir şekilde çalıştıramazdık.
- **Kısaca syntax** olmasaydı, yazdığımız işlemleri rastgele olarak çalıştıran bir yazılım ile ve bir çok hata ile karşı karşıya kalırdık.

PROGRAMLAMA DİLLERİ SYNTAX (SÖZ DİZİMİ)



JAVA

```
public static Main{  
  
    public static void main (String [] args) {  
  
        int a=5;  
  
        int b=9;  
  
        int c=a+b;  
  
        System.out.println("a+b = " + c);  
  
    }  
}
```

PYTHON

```
a=5  
  
b=9  
  
c=a+b  
  
print("a+b =", c)
```

PHP

```
<?php  
$a=5;  
$b=9;  
$c=$a+$b;  
echo($a." + ".$b." = ".$c);  
  
?>
```

PYTHON



- Aralık 1989, Amsterdam : Guido Van Rossum
- Monty Python's
- Ocak 26, 1994 1.0.0
- Ekim 16, 2000 2.0
- Aralık 3, 2008 3.0
- ...
- 2.x – 3.x

www.python.org



PYTHON



- Genel kullanıma yönelik
- Yüksek seviyeli
- Nesne yönelimli
- Modüler
- Platform bağımsız





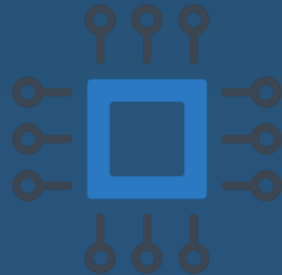
PROGRAMLAMA DİLİ SEVİYELERİ



} İnsan mantığı

Yüksek seviye diller	Python , Ruby , ...
Orta seviye diller	C# , Java , ...
Düşük seviye diller	C , C++ , ...

} Programlama dilleri



} Makine mantığı



NEDEN PYTHON?




















- Geniş ölçekte geliştirme (web, masaüstü, farklı platformlar)
 - Bilimsel uygulamalar
 - Veri madenciliği uygulamaları
 - Yapay zeka uygulamaları
-
- Google, Wikipedia, NASA, CERN, ...





NEDEN PYTHON?

- 2003 yılından beri en popüler 10 dil arasında

Language Rank	Types	Spectrum Ranking
1. Python	  	100.0
2. C++	  	99.7
3. Java	  	97.5
4. C	  	96.7
5. C#	  	89.4
6. PHP		84.9
7. R		82.9
8. JavaScript	 	82.6
9. Go	 	76.4
10. Assembly		74.1

<https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages>



GELİŞTİRME ORTAMLARI

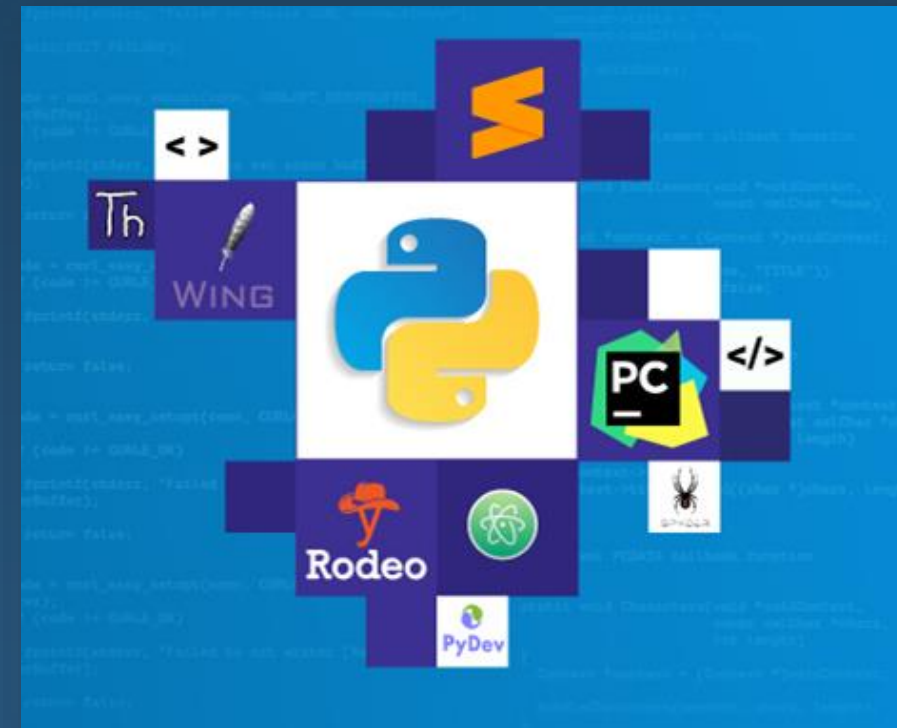
- Programları yazmak, saklamak, düzenlemek, çalıştırmak, paketlemek gibi süreçler için kullanılan editörler
- IDE : Integrated Development Environment
 - Geliştirme sürecinin tümünü organize eden araçları barındıran yazılımlar



PYTHON IDE



- IDLE
- PyCharm
- Spyder
- Jupiter Notebook
- Visual Studio Code
- Atom
- Thonny
- ...





KOD DOSYASI

- Ön tanımlamalar (modül tanımlamaları)
- Açıklama satırları (yorumlar)
- Kod satırları

1	<code>import os</code>	→	ön tanımlama
2	<code><i>#işletim sistemi adını yaz</i></code>	→	açıklama satırı
3	<code>print(os.name)</code>	→	kod satırı



PYTHON SÖZ DİZİMİ (SYNTAX)

- Girintilere dayalı

```
for i in range(50) :  
    if i%3==0:  
        pass  
    print(i)
```

DEĞİŞKEN



- Programdaki her veri için bellekte yer tahsis edilir.
- Bellek adresleri ile işlem yapmak zor olduğu için bölgelere verilen sembolik isimlere «değişken» denir.

	BELLEK (RAM)	
	3	0x0011..
	3214521454.111	
	?	0x0012..
	

DEĞİŞKEN



- Değişkenler bellek üzerindeki değer tutan hücrelerdir.
- Her değişken için, içinde barındırdığı verinin türüne göre farklı büyüklükte alan tahsis edilir.
- Tam sayı, ondalıklı sayı, karakter....

	BELLEK (RAM)	
	3	0x0011..
	3214521454.111	
	?	0x0012..
	

DEĞİŞKEN TANIMLAMA



- Değişkenler Python'da şu şekilde tanımlanır:

a=3

b=3214521454.111

c='?'

BELLEK (RAM)	
a {	3
b {	3214521454.111
c {	?

0x0011..

0x0012..



DEĞİŞKEN ADLANDIRMA KURALLARI

- Değişken adlarının başında rakam bulunamaz.

~~1sayi~~

- Değişken adları içerisinde, başında olmamak kaydıyla, rakam kullanılabilir.

sayi1



DEĞİŞKEN ADLANDIRMA KURALLARI

- Değişken adlarında boşluk kullanılamaz.

~~ilk_sayi~~

- Değişken adlarında özel karakterlerden sadece alt çizgi (_) kullanılabilir. Değişken adlarının başında da alt çizgi kullanılabilir.

ilk_sayi _ilkSayi



DEĞİŞKEN ADLANDIRMA KURALLARI

- Programlama dilinin rezerve kelimeleri değişken adı olamaz.

~~True~~

~~return~~

~~class~~

class

- Değişkenleri adlandırırken İngiliz alfabesindeki karakterleri kullanmanız tavsiye edilir.

«sayı» yerine «sayi»

«küçük_değer» yerine «kucuk_deger»



DEĞİŞKEN ADLANDIRMA KURALLARI

- Python büyük-küçük harf duyarlı (case-sensitive) bir dildir.
 - Örneğin `sayi` ve `Sayi` ismiyle tanımlanan iki değişken, okunuşları aynı olsa da, yazılışları farklı olduğu için Python açısından iki ayrı değişkendir.

`sayi=10`

`Sayi=20`



DEĞİŞKENLERE DEĞER ATAMA

- Değer ataması yapılırken = (eşittir) operatörü kullanılır.

a=3 b=5

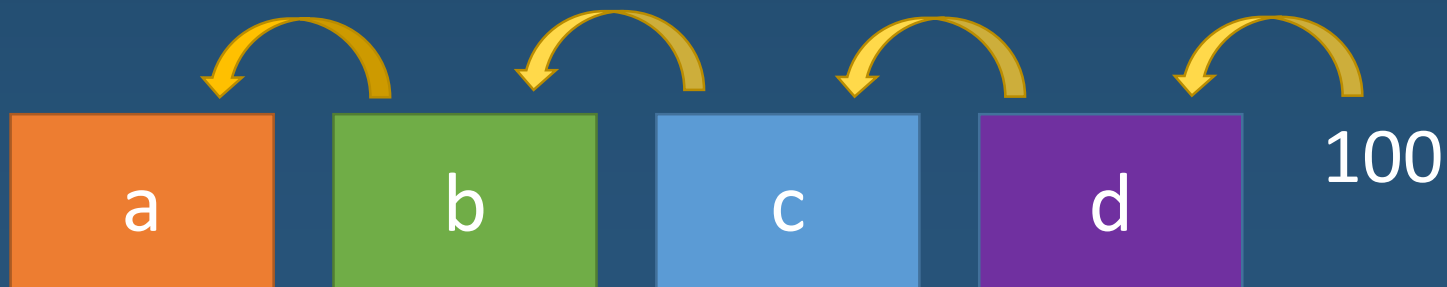
- Atama her zaman sağdan sola doğru yapılır.
- 'a=3' ifadesi doğrudurken '3=a' ifadesi atama için yanlıştır.



DEĞİŞKENLERE DEĞER ATAMA

- Birden çok değişkene aynı değer atanırken (zincir atama):

`a=b=c=d=100`

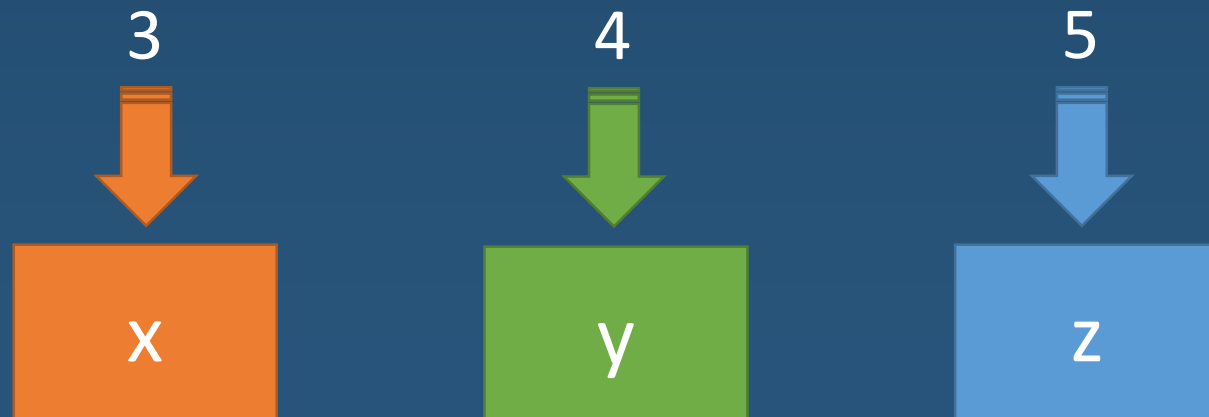


DEĞİŞKENLERE DEĞER ATAMA



- Birden çok değişkene eş zamanlı değer atanırken:

$x, y, z = 3, 4, 5$

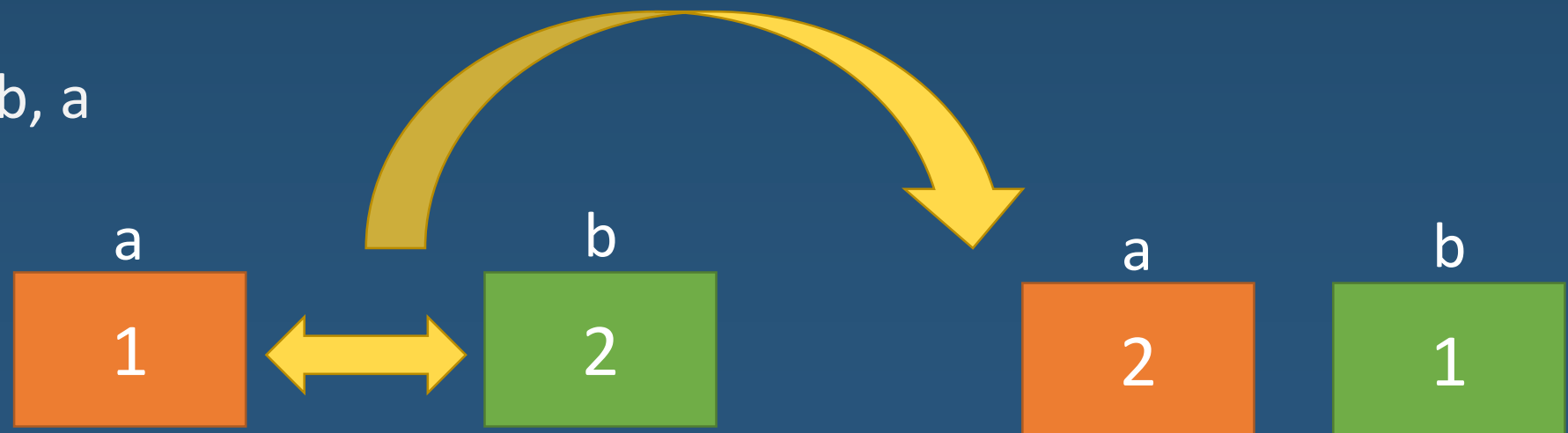




DEĞİŞKENLERE DEĞER ATAMA

- Değişkenlerin değerleri eş zamanlı yer değiştirilirken:

```
a = 1  
b = 2  
a, b = b, a
```

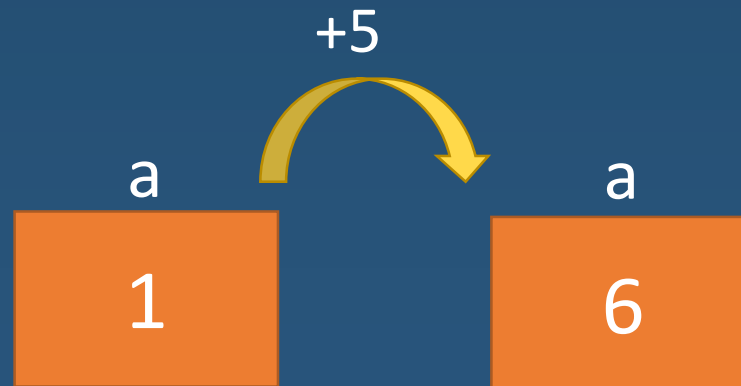




DEĞİŞKENLERE DEĞER ATAMA

- Değişkenlerin değeri kendi üzerinde değiştirilirken:

```
a = 1  
a = a+5
```



VERİ TÜRLERİ



- Python dilinde 3 temel veri tipi vardır:
 - ✓ Sayısal (Number)
 - ✓ Karakter dizisi (String)
 - ✓ Mantıksal (Boolean)
- Her değişken üzerine atanan verinin türüne göre dinamik olarak biçimlenir.



SAYISAL TÜRLER (NUMBER)

- 3 temel sayısal veri türü vardır:

- ✓ int : Tamsayı 25
- ✓ float: Ondalıkli sayı 3.14
- ✓ complex: Karmaşık sayı 5+3j

a = 25

a değişkeni Python için **int** türündedir.

b = 3.14

b değişkeni Python için **float** türündedir.

c = 5+3j

c değişkeni Python için **complex** türündedir.



KARAKTER DİZİSİ (STRING)

- Tek karakter ya da karakter grupları (sözcükler)
- Tek ya da çift tırnak içinde gösterilir
`str = "python"` `str = 'python'`
- *Tırnak açıldığı şekliyle kapatılmalı*

`str = "python'` tanımlaması **yanlıştır.**



KARAKTER DİZİSİ (STRING)

- Karakter dizisinin bir parçası olarak tırnak işareti kullanılacaksa:
 - *String içindeki tırnak tek ise, String'i tanımlayan tırnak çift olmalı*
str = 'Güçlü "python" dili'
 - *String içindeki tırnak çift ise, String'i tanımlayan tırnak tek olmalı*
str = "Güçlü 'python' dili"



MANTIKSAL TÜR (BOOLEAN)

- Doğru ya da yanlış ifadeleri

True

False

m1 = True

m2 = False

m3 = 2 < 3



DEĞİŞKENİN VERİ TÜRÜNE ERİŞME

```
a = 12  
type(a)
```

<class 'int'>

```
b = '12'  
type(b)
```

<class 'str'>



STRING TÜRÜ

- Tek ya da çift tırnak içerisinde gösterilir.
- Tırnak açıldığı şekliyle kapatılmalı
`str = "python"` `str = 'python'`
- String içinde de tırnak kullanılabilir
`str = 'Güçlü "python" dili'`
`str = "Güçlü 'python' dili"`

TÜR UYUMU



- Python dilinde tür uyumları sıkı bir şekilde kontrol edilir.
- Bir sayı(int) ile bir String toplanamayacağından dolayı aşağıdaki kod hata verir.

```
1+"2"
```

`TypeError: unsupported operand type(s) for +: 'int' and 'str'`

TÜR DÖNÜŞÜMÜ



- Python'da veri türleri üzerinde işlemler gerçekleştirebilmek için türlerin farklı türlere dönüşümü gerekir.
- Örneğin input() ile okunan her şey doğası gereği String türünde gelir.
- Kullanıcıdan bir sayı girmesini istemişsek bu veri String formunda okunur ve sayıya dönüştürülmesi gerekir.

"2" -> 2



int TÜRÜNE DÖNÜŞÜM

- Herhangi bir türü tamsayıya dönüştürmek için int() dönüşüm komutu kullanılır.

```
a = input("Bir sayı giriniz:")  
b = int(a)  
c = b+5
```

```
b = int(input("Bir sayı giriniz:"))  
c = b+5
```




float TÜRÜNE DÖNÜŞÜM

- Herhangi bir türü ondalıklı sayıya (floating point – kayan noktalı sayı) dönüştürmek için float() dönüşüm komutu kullanılır.

```
sayi = float(input("Bir sayı giriniz:"))  
sonuc = sayi + 2.5
```



String TÜRÜNE DÖNÜŞÜM

- Herhangi bir türü karakter dizisine çevirmek için `str()` komutu kullanılır.

```
no = 185  
caddeAdi = str(no) + ". cadde"  
print(caddeAdi)
```



KARAKTER TÜRÜNE DÖNÜŞÜM

- Sayısal bir değerin karakter türündeki karşılığını bulmak için `chr()` komutu kullanılır.

```
karakter = chr(65)  
print(karakter)
```

Ekrana 65 sayısının kod tablosunda karşılık geldiği A karakterini yazar.



MANTIKSAL TÜRE DÖNÜŞÜM

- Herhangi bir verinin boolean karşılığını elde etmek için `bool()` komutu kullanılır.
- Python dilinde sıfır ya da boş olmayan veriler mantıksal `True` değerine karşılık gelir.

```
print(bool(1))      => True
print(bool('a'))    => True
print(bool(0))      => False
print(bool(""))     => False
```



Operatör Nedir?

- Veriler üzerinde işlemler yapmamızı sağlayan, özel bir fonksiyon yüklenmiş semboller





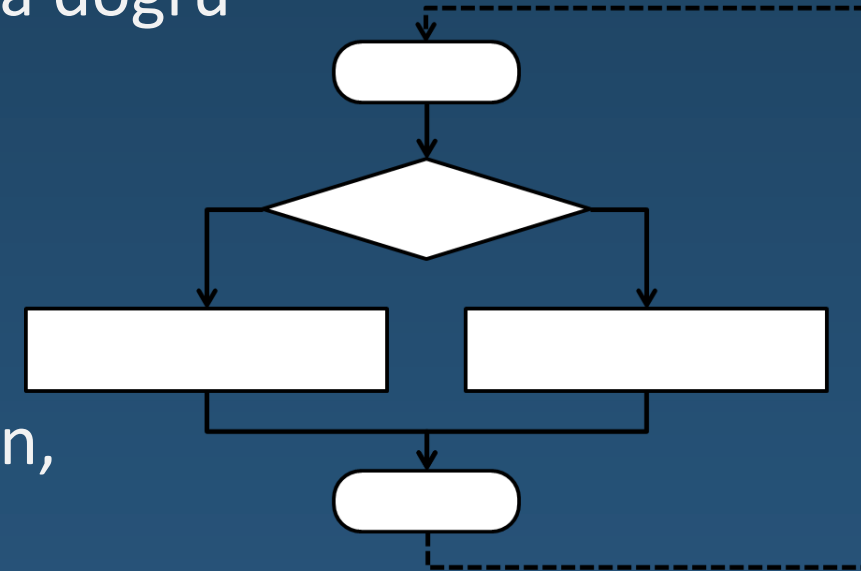
Operatör Türleri

- İşlevlerine göre:
 - Aritmetik operatörler
 - İşaret operatörleri
 - Atama operatörü
 - Karşılaştırma operatörleri
 - Mantıksal operatörler



Akış Kontrolü

- Yazılan programlar normalde ilk satırdan sona doğru akarak tamamlanır.
- Ancak bazen program içinde
 - Standart kod akışını değiştiren,
 - Programı farklı noktalara dallandıran,
 - Kod bloklarını birden çok kez tekrar ettiren,
 - Mevcut çalışan kod bloklarını sonlandıran mekanizmalar yer alır.





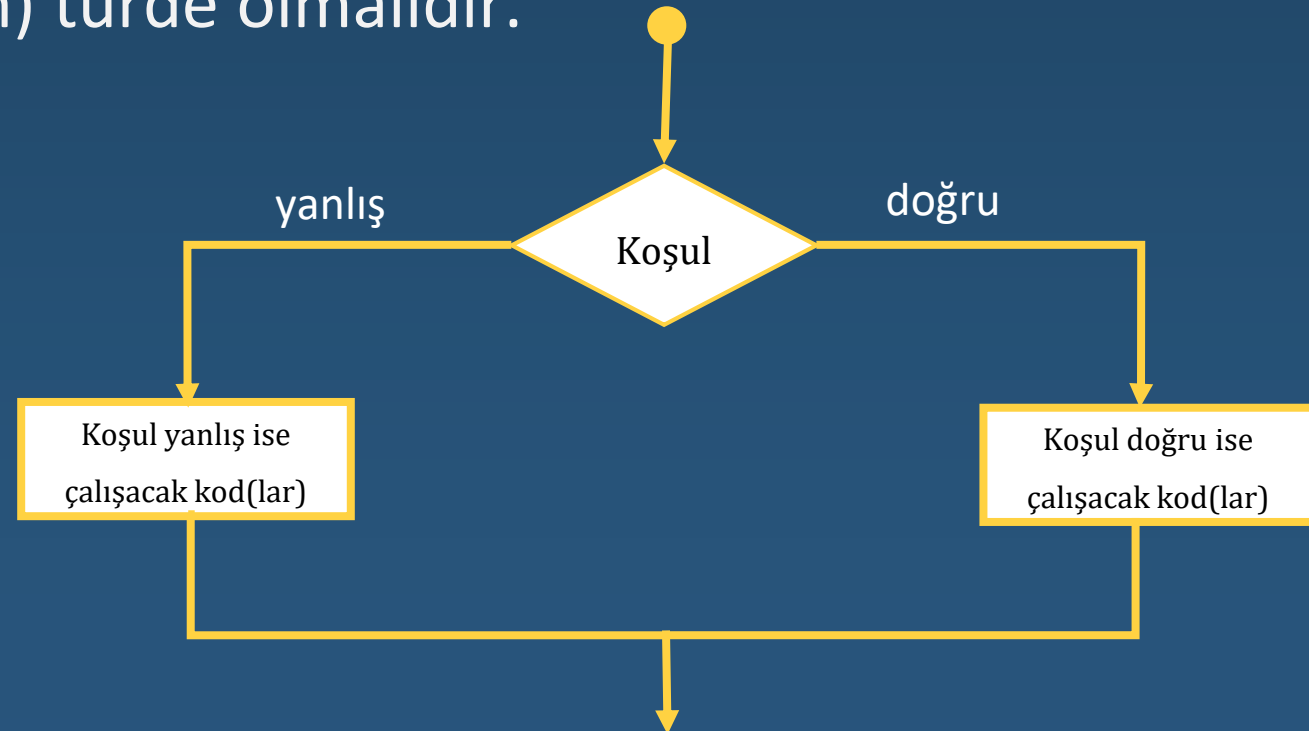
Akış Kontrolü

- Bunlara akış kontrol mekanizmaları denir.
- Karar yapıları (if, else, elif)
- Döngüler (while, for)
- Atlama ifadeleri (break, continue)



Karar Yapıları

- Programın çalışması esnasında gerçekleşen koşullara bağlı olarak program akışını farklı yollara yönlendirir.
- Koşul ifadesi mantıksal (boolean) türde olmalıdır.
- Python sıfır ya da null'dan farklı her şeyi True kabul eder.






if yapısı

- if – eğer ise
- Herhangi bir koşulun sağlanması durumunda çalışacak kodları tanımlarken kullanılır.

if koşul:

koşula bağlı ifade(ler)

if bloğuna ait kodlar yazılırken bir tab tuşu kadar boşluk bırakılmalı



if yapısı

- if – eğer ise

```
puan=82
if puan>70:
    print("başarılı")
```

```
sayi=100
if sayi%2==0 and sayi%5==0:
    print("sayı çifttir")
    print ("sayı beşin katıdır")
    print ("koşula bağlı olmayan satır")
```



if else yapısı

- Herhangi bir koşulun gerçekleşmesi ve gerçekleşmemesi halinde yapılacak iki ayrı iş varsa if bloğundan sonra else bloğu da kullanılır.

if koşul:

koşul doğruysa yapılacaklar

else:

koşul yanlışsa yapılacaklar

if else yapısı



```
puan=65
if puan>70:
    print("başarılı")
else:
    print("başarısız")
```



if elif yapısı

- Bir koşulun sağlanmaması halinde çalışacak olan kodlar başka bir koşula bağlıysa if bloğunun else bölümünde ikinci bir if yer almalıdır.
- Bu durumda ortaya çıkan else-if yapısı Python'da kısaca **elif** olarak ifade edilir.



if elif yapısı

```
if koşul_1:  
    ifade(ler)  
elif koşul_2:  
    ifade(ler)
```

```
sicaklik=28  
if sicaklik>30:  
    print("hava çok sıcak")  
elif 25<=sicaklik<=30:  
    print("hava normal")  
else:  
    print("hava serin")
```



if elif yapısı

```
sicaklik=28
if sicaklik>30:
    print("hava çok sıcak")
elif 25<=sicaklik<=30:
    print("hava normal")
else:
    print("hava serin")
```

VS

```
sicaklik=28
if sicaklik>30:
    print("hava çok sıcak")
if 25<=sicaklik<=30:
    print("hava normal")
else:
    print("hava serin")
```




İç içe if yapıları (nested)

```
renk1="kırmızı"  
renk2="mavi"  
if renk1=="kırmızı":  
    if renk2=="sarı":  
        print("turuncu")  
    elif renk2=="mavi":  
        print("mor")  
elif renk1=="yeşil":  
    if renk2=="beyaz":  
        print("açık yeşil")
```



Koşula bağlı tek ifade

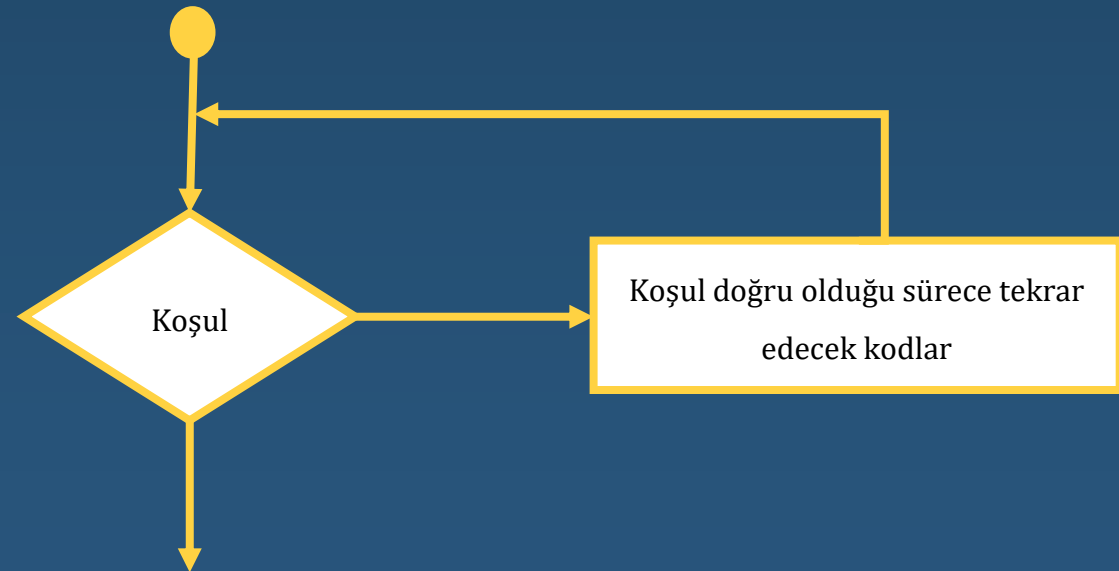
- Bir koşula bağlı olarak çalışacak tek bir kod ifadesi varsa, alt satıra inmeden, tab kadar boşluk bırakmadan koşulun devamına yazılabilir:

```
sayi=7  
if sayi%2==0: print("sayı çifttir")  
else: print("sayı tektir")
```



Döngüler

- Koşul sağlanmaya devam ettiği sürece belirli bir kodu tekrarlayan mekanizmalardır.
- while
- for



while



- *while – iken, olduğu sürece*

while koşul:

koşula bağlı tekrarlanacak ifade(ler)



while

- 1'den 10'a kadar sayıları ekrana yazan program

```
sayi=1  
while sayi<=10:  
    print(sayi)  
    sayi=sayi+1  
print ("döngü sonlandı")
```



Sonsuz Döngü

- Koşul sürekli True ise

while True:

a=int(input ("ilk sayıyı giriniz"))

b=int(input ("ikinci sayıyı giriniz"))

print ("Girilen iki sayının toplamı:", a+b)

for



- *for – için*
- *while* döngüsüne benzer şekilde koşul sağlanmaya devam ettiği müddetçe kodu tekrarlar.
- *while* döngüsünden daha yeteneklidir, kullanım alanı geniştir
- Genelde belirli sayıda tekrar etmesi istenen kodlar ya da bir veri kümesi içindeki elemanlar üzerinde gezinmek için kullanılır.

for



- Veri kümesi üzerinde gezinme
for döngü_değişkeni in üzerinde_dolaşılacak_veri:
döngü_içi_işlemler

```
for i in "abcd":  
    print(i)
```

a
b
c
d

```
harfler = "abcd"  
for i in harfler:  
    print(i)
```




for - range

- range belirli bir aralıkta sayı dizilimi üretmek için kullanılır:

range(10) => 0,1,2,3,4,5,6,7,8,9

range(5,10) => 5,6,7,8,9

range(4,15,2) => 4,6,8,10,12,14

range(10,2,-2) => 10,8,6,4



for - range

- for döngüsü yardımıyla range ile üretilen sayı dizilimleri üzerinde gezinilebilir.
- Bu şekilde belirli bir sırada, sayıda işlem yapılmış olur.

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

*0
1
2
3
4*

for - range



- Tersten sayma

```
for s in range(5,0,-1):  
    print(s)
```

5
4
3
2
1



Atlama İfadeleri

- Bir döngü ya da karar yapısı içerisinde,
 - mevcut işlem adımını kırıp bir sonraki adıma geçmek (**continue**)
 - döngüyü tümünden sonlandırmak (**break**)



break

- break - kırmak
- Belirli bir koşul sağlandığında döngüyü kırmak için kullanılır.
- break ifadesi çalışınca döngü sonlanır ve program döngünün bittiği satırdan itibaren çalışmaya devam eder.



break

- Girilen sayı negatif olmadığı müddetçe sayı isteyip toplama ekleyen, negatifse toplamı ekrana yazıp sonlanan program:

```
toplam=0  
while True:  
    sayi = int(input("Sayı giriniz:"))  
    if sayi<0:  
        break  
    toplam+=sayi  
print("Toplam:", toplam)
```



continue

- continue – devam et
- Döngünün mevcut adımını sonlandırıp bir sonraki adıma geçer
- break gibi döngüyü tümünden sonlandırmaz.



continue

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

*1
3
5
7
9*

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


Örnekler



- 1) 1'den 10'e kadar olan sayıların toplamı
- 2) 1'den 10'a kadar olan tek sayıların toplamı
- 3) 1'den 10'a kadar olan tek ve çift sayıların ayrı ayrı toplamı
- 4) Kullanıcının girdiği sayının rakamlarının toplamı
- 5) Girilen sayının faktöriyelini hesaplama



6) Toplamda 10 sorunun sorulduğu bir sınavda her doğru cevap için 10 puan alınırken her yanlış cevap için -5 puan alınmaktadır. Tüm soruları cevaplayan bir kişinin doğru yanlış sayısı klavyeden girildiğinde toplam puanını ekrana yazan programın kodunu yazınız.

7)

$$\sum_{i=0}^n 5i + 6$$

Yukarıda verilen toplam sembolüne göre klavyeden girilen n sayısı için işlemin sonucunu hesaplayıp ekrana yazan programın kodunu yazınız.



8) İkinci dereceden bir bilinmeyenli bir denklemin delta değerini hesaplayan ve bu değere göre denklemin köklerini ekrana yazan uygulamanın kodunu yazınız.

$$\Delta = b^2 - 4ac$$

$$X_1 = \frac{-b + \sqrt{\Delta}}{2a}$$

$$X_2 = \frac{-b - \sqrt{\Delta}}{2a}$$



9) Toplamda 3 cevap hakkı verilen bir sayı tahmin oyununda;

- Yarışmacı sayıyı doğru tahmin ettiğinde kaçınıcı hakkında doğru tahmin ettiği ile birlikte ekrana yazan ve programı sonlandıran, (Örnek Çıktı: 2. Tahminde bildiniz)
- 3 hakkında da doğru tahmin edemeyen yarışmacıya “Doğru tahmin yapamadınız” mesajı veren programın kodunu yazınız.