İçindekiler

3. ÜNİTE: PROGRAMLAMANIN TEMEL KAVRAMLARI

3.1. Programlamada kullanılan işlem ve semboller

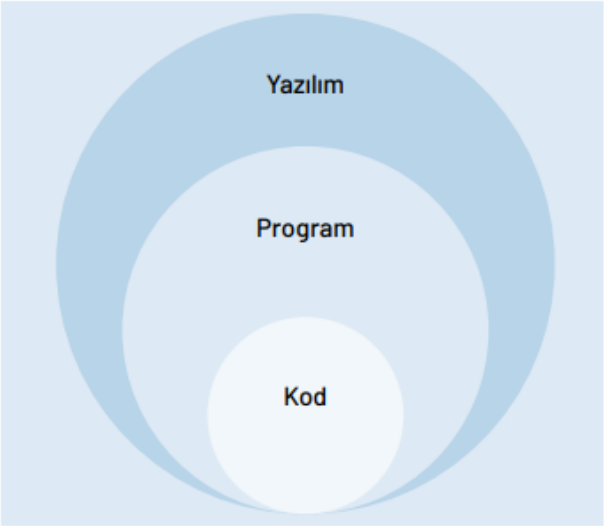
Bilgisayar, donanım ve yazılım olmak üzere iki temel unsurdan oluşur ve bu unsurlar birbirini tamamlar.

Donanım: bilgisayarın duyu organlarıyla algılanabilen fiziksel parçalarıdır.

Yazılım: donanım parçalarının çalışmasını veya bilgisayarın belli bir amaç doğrultusunda daha işlevsel kullanılmasını sağlayan kodlar ile yazılmış programları kapsayan sistemlerdir.

Programlar yazılırken doğru, güvenilir, kullanışlı, masrafsız ve basit olması kriterlerine göre tasarlanır. Kullanılacak amaç doğrultusunda programlama dili seçilir ve seçilen dilin kurallarına uygun şekilde program oluşturulur.

Programlama dillerinde basit farklılıklar bulunurken ortak kullanılan kavramlar, programlama dillerinin temelini oluşturur.



Bu kavramlardan;

- operatörler,
- değişkenler,
- döngü koşulları

gibi temel programlama kavramları kullanılarak program yazımı gerçekleşir. Temel kavramlar birçok dilde benzer şekilde kullanılır.

Operatörler: Programlamada veri üzerinden matematiksel işlemler, değişkenlere değer atama gibi işlemler gerçekleştirmeyi sağlar.
Değişkenler: Programlamada veri saklamayı, yönetmeyi ve gerektiğinde verinin değişebilmesini sağlayan isim alanlarıdır.

Değişken isimleri “a, b” gibi karakter veya “sayi1, sayi_2” gibi karakter toplulukları ile ifade edilerek değişkenlerde sayıların, metinlerin veya başka tür verilerin saklanması sağlanır. Oluşturulan **döngü koşulları** gibi temel yapılar kullanılarak da programın akış yönü belirlenir**.** **Döngü yapıları** ile defalarca tekrarlanan işlemlerde fazla komut satırı yazımının önüne geçilir.

Kullanılan operatörlerin gösterimi aynı olsa da operatörler, kullanılan yazılım dillerinde **farklı işlevleri** yerine getirebilir.

Örneğin

- " = " (eşittir) operatörü, bir yazılım dilinde değişkene değer ataması işlemini gerçekleştirirken başka bir dilde " == " eşitlik ifade etmek için

- "< >" operatörleri, birçok dilde küçüklük ve büyüklük karşılaştırmaları için kullanılırken bazı programlarda "" arasına "" gibi programlama diline ait kütüphane dosyaları yazılarak kütüphane dosyalarının programa dâhil edilmesi sağlanır.

- "/" operatörü, bölme işlemi için kullanılır. Bazı dillerde kodların açıklama satırı olarak yazılmasını sağlayan kullanım şeklinin yanı sıra komut çıktısının bir alt satıra yazılmasını sağlayan kullanımı da vardır. Örneğin bir dilde "//" şeklinde açıklama yazmayı sağlarken başka bir dilde "/*.....*/" arasına açıklama yazarak kullanılır. Python programlama dilinde ise "#" diyez karakteri açıklama satırı için kullanılır.

3.1.1. Operatörler

Aritmetiksel Operatörler Temel matematik işlemlerini yapmayı sağlayan karakterlerdir. En genel kullanımı

Aritmetiksel Operatörlerin Gösterimi

OPERATÖRLER	OPERATÖR TANIMI	ÖRNEK	İŞLEM AÇIKLAMASI
+	Toplama	10+5	15
-	Çıkarma	10-5	5
*****	Çarpma	10*5	50
/	Bölme	10/5	2
**** veya ^**	Üs Alma-Kuvvet Alma	10**5 veya 10^5	10x10x10x10x10 = 100000
%	Mod Alma	10%5	10/5 işleminde kalan hanesindeki sonuç = 0

ÖRNEK

Program Kodu	Değerler ve İşlemler	Örnek Ekran Görüntüsü
a=10;	a değişkenine 10 değeri atanmış	
b=5;	b değişkenine 5 değeri atanmış	
print(a+b)	10+5	15
print(a-b)	10-5	5
print(a*b)	10*5	50
print(b/a)	5/10	0.5
print(ab)**	10**5 ya da 10^5	100000 (10^5)
print(a%2)	10/2 (mod)	0 (10/2 kalan)
print(b%3)	5/3 (mod)	2 (5/3 kalan)

Mod alma işlemi bir sayının başka bir sayıya bölünmesiyle **kalan** hanesindeki değeri ifade eder. **Matematikte** olduğu gibi programlamada da **işlem önceliği aynıdır**. İşlem önceliğine göre yapılacak işlemler aşağıda sırasıyla listelenmiştir:

1. Parantez içleri ()
2. Üs alma işlemi **, ^
3. Mod alma, çarpma, bölme işlemi % , * , /
4. Toplama ve çıkarma işlemi +, -

İşlemler sırasıyla gerçekleştirilerek problemin çözümü sağlanmış olur.

Karşılaştırma Operatörleri

(Önemli *)

OPERATÖR	OPERATÖR TANIMI	ÖRNEK	İŞLEM AÇIKLAMASI
==	Eşittir	a==b	a ile b eşit mi?
!=	Eşit değildir	a!=b	a ile b eşit değildir mi?
>	Büyüktür	a>b	a, b'den büyük mü?
<	Küçüktür	a<b	a, b'den küçük mü?
>=	Büyük eşittir	a>=b	a , b'den büyük veya b'ye eşit mi?
<=	Küçük eşittir	a<=b	a, b'den küçük veya b'ye eşit mi?

Örnek

Program Kodu	Koşul ifadesi	Örnek Ekran Görüntüsü
a=4;	a = 4	
b=8;	b = 8	
print(a==b)	4 == 8 (Eşittir)	FALSE
print(a!=b)	4 != 8 (Eşit değildir)	TRUE
print(a>b)	4 > 8 (büyüktür)	FALSE
print(a<b)	4 < 8 (küçüktür)	TRUE
print(a>=b)	4 > = 8 (büyük ya da eşit)	FALSE

(Önemli *)

Mantıksal Operatörler

İki veya daha fazla verinin birbiri ile kıyaslandığı durumlarda kullanılır. Yapılan kıyaslamalar, verilen mantıksal işlemlere göre gerçekleştirilir ve sonuç olarak boolean tipi (true/false) değerler üretilir.

OPERATÖR	OPERATÖR TANIMI	ÖRNEK	İŞLEM AÇIKLAMASI
&& (and)	Ve [and (end)] İşlemi	a<15 && b==20	İki şartın da sağlanması durumunda sonuç true olur.
|| (or)	Veya [or (or)] İşlemi	a>15 || b==20	Şartlardan birinin sağlanması yeterlidir, sonuç true olur
^ (xor)	Özel Veya [x or (eksor)] İşlemi	a==10 ^ b>25	Şartların sonuçları farklı ise true olur.
! (not)	Değil [not (nat)] İşlemi	!(a<b)	Verilen şart sonucunun tersini alır.

Örnek:

Program Kodu

```
a=10
b=20
print(a<15 and b==20)
print(a<15 or b==20)
print(a==10 ^ b>20)
print (not(a<b))
```

Örnek Ekran Görüntüsü

```
True
True
False
False
```

3.1.2. Değişkenler

Programlamada veri saklamayı, yönetmeyi ve gerektiğinde verinin değişebilmesini sağlayan isim alanlarıdır.

Programlama işlemlerinde kullanılan veriler (değerler), değişkenler ile isimlendirilerek kullanılır. Değişken için verinin türüne göre bilgisayar tarafından hafıza alanları ve bu alanları gösteren adresler oluşturulur. Değişkenlere verilen isimler ile adresteki veriye erişim sağlama, adrese veri atama, adresteki veriyi okuma ve değiştirme gibi işlemleri yapabilmesi sağlanır.

Değişken isimleri "a, b" gibi karakter veya "sayi1, sayi_2" gibi karakter toplulukları ile ifade edilerek değişkenlerde sayıların, metinlerin veya başka tür verilerin saklanması sağlanır. Oluşturulan **döngü koşulları** gibi temel yapılar kullanılarak da programın akış yönü belirlenir**.** **Döngü yapıları** ile defalarca tekrarlanan işlemlerde fazla komut satırı yazımının önüne geçilir.

DEĞİŞKEN YAZIM KURALLARI:

- **Bir sayı ile başlayamaz.**
- **Boşluk veya özel karakter içeremez (alt çizgi _ hariç).**
- **Program komutları ('if; 'for' vb.) isim olarak kullanılamaz.**
- **Anlamlı ve hatırlanması kolay isimler seçilmelidir!**

3.2. Veri Türleri

Bilgisayar işlemlerinde kullanılan en temel yapıya **veri** denir. Veriler, kullanılacak işlemlere göre türü belirtilerek **değişkenler** ile beraber tanımlanır. Örneğin veri sayısal değerlerden oluşuyor ise sayısal veri türü; metinsel değerlerden oluşuyor ise metinsel veri türü belirtilerek değişkene tanımlanır. Değişkene atanacak verinin çeşidine göre tür belirtilir. İşlemlerin sınırı ve hafızada kaplayacağı alan türe göre belirlenir. Temel veri türlerini temsil eden ilkel veri türleri, programlama dillerinde yerleşik olarak bulunan türlerdir. **Sayısal** ve **metinsel veri türü** olmak üzere çeşitleri mevcuttur ve bu veri türleri kullanım alanlarına göre çeşitlilik gösterir.

VERİ TÜRLERİ				
Sayısal Veri Türleri		**Metinsel Veri Türleri**		**Mantıksal Veri Türleri**
Tam Sayılar	**Ondalıklı Sayılar**	**Karakter**	**Karakter Dizisi**	**Boolean**
byte (1 byte)	float (4 byte)	char (1 veya 2 byte)	string (1 byte)	true (1 byte)
short (2 byte)	double (8 byte)			false (1 byte)
int (4 byte)	long double (12 byte)			
long (4 byte)				

Byte:

Bellekte 1 byte yer kaplar. Basit ve işlem sonuçlarının küçük kapasitelere ihtiyaç duyacağı durumlarda kullanılır. Byte veri türünün tanımlaması ***byte a=10;*** şeklindedir.

Integer:

Yaygın kullanılan veri türüdür. 4 byte' lık yer kaplar. Byte veri türüne göre daha karmaşık ve büyük verilerle çalışma imkânı sağlar. Integer veya kısaltması olan int kelimeleriyle belirtilir. Integer veri türünün tanımlaması ***int a=100000; veya a=100000;*** şeklindedir.

Long:

Long [(uzun)], Integer veri türünden daha fazla kapasiteye sahiptir. 8 byte' lık yer kaplar. Integer veri türüne göre daha karmaşık ve büyük verilerle çalışma imkânı sağlar. Long veri türünün tanımlaması ***long a=1000000;*** şeklindedir.

Float:

Float [(kesirli sayı)], 4 byte' lık yer kaplar. Ondalıklı sayı ifadelerin olduğu basit işlemlerde tercih edilir. ***Noktadan sonra altı karakter*** kullanımını sağlar. Bazı dillerde verinin sonuna **"f" ya da "F"** şeklinde belirtilerek kullanılır. Float veri türünün tanımlaması ***float a= 3.14f*** şeklindedir.

Örnek Veri Türleri ve Değişkenler

Python Veri Türleri	Değişken Örnekleri
İnteger	Yas=17
Float	Fiyat=199.99
String	Ad=”Tarık”
Boolean	Aktif_mi=”True”
List	Renkler=[“sarı”,”lacivert”,”kırmızı”,”siyah”]
Tuple	Koordinat=(10,30)
Dictionary	Ogrenci={“ad” : ”Ahmet”, ”yas” : “17”}

Bir metin ile bir sayıyı matematiksel olarak toplayamazsınız. Tıpkı 3 elma ile 3 armudun toplanamaması gibi. Veri türleri bu uyumu sağlar.

“Tür seçimi doğru yapılan değişkenler ile karşılaşan bilgisayar hata vermez. “

Örneğin:

Sayısal bir veri tanımlanırken **integer** [(**tamsayı**)] türü belirtilir.

int sayi= 10

Metinsel bir veri tanımlanırken **string** [(**dizi**)] türü belirtilerek

string ders= “Programlamaya Giriş ve Algoritmalar”

gibi tanımlamalar kullanılır.

Tanımlanan “**sayi**” ve “**ders**” değişkenleri, doğru ve değişken içindeki değerler türleri ile uyumludur.

Tür ile değişken değerleri birbiriyle uyumlu olmaz ise hata meydana gelir ve program çalışmaz.

Örnek

Sayısal ve metinsel veri türü kullanılarak oluşturulan programın kodunu ve ekran çıktısını yazınız.

Program Kodu	Örnek Ekran Görüntüsü
a=5 **b=10** **c=a*b** **isim=”Merhaba”** **print(c)** **print(isim)**	**50** **Merhaba**

Veri türü: Sayısal (Integer)	Veri türü: Metinsel (String)
a = 10 **sonuc = a * 5** **Print(sonuc)**	b = "10" sonuc = b * 5 Print(sonuc)
Ekran Çıktısı : **50** (Burada matematiksel bir çarpma işlemi yapılır.)	**Ekran Çıktısı** **sonuc = “10” * 5** **1010101010** (Burada metin 5 defa METİN yan yana yazdırılır.)

Diziler (list)

Bazen aynı türden çok sayıda malzemeyi bir arada tutmamız gerekir. Örneğin, **bir haftanın günleri** veya **bir sınıfın notları**. Bunları tek tek kutulara koymak yerine, hepsini bölmeli bir rafta saklarız. Bu raflara dizi (Python'da genellikle liste denir) adı verilir.

Dizi Mantığı

- Tek bir isimle ('notlar') birden fazla veriyi saklar.
- Her verinin bir sıra numarası (indis) vardır ve bu numara O'dan başlar.

Örneğin:

4 elemanlı bir sayı dizisi (listesi)

sayilar = [10, 20, 30, 40]

indisteki (yani ikinci) elemana ulaşmak için : print(sayilar[1]) = 20

sayilar[0]=10 (dizinin 1. Elemanıdır)

sayilar[3]=40 (dizinin son elemanıdır)

Değişken Yazım Kuralları:

- Bir **sayı** ile başlayamaz.
- Boşluk veya özel karakter içeremez (**alt çizgi _ hariç).**
- Program komutları **(if, for, while, and, or, vb.)** isim olarak kullanılamaz.
- Türkçe karakterlerin **(ç, ğ, ı, İ, ö, ş, ü)** kullanılmaması daha uygundur.
- Anlamlı ve hatırlanması kolay isimler seçilmelidir!
- Metinsel verileri aktarırken =”metin” (çift tırnak içerisinde) ya da =’metin’ (tek tırnak içerisinde) şeklinde aktarılmalıdır.
- Bazı programlama dillerinde harf duyarlılığı vardır. Dolayısıyla **İsim** değişkeni ile isim değişkeni aynı şey demek değildir.

3.3. FARKLI VERİ TÜRLERİNİN PROGRAMDA KULLANIMI

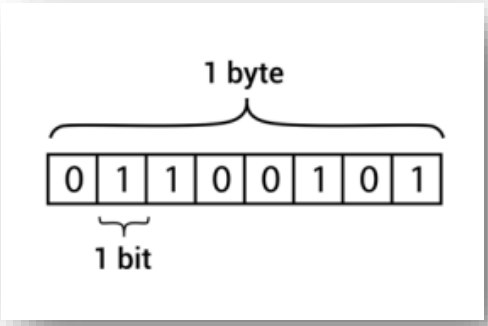
Sayısal bir işlem gerçekleştirilecek ise bu verinin **sayıyı**; **metinsel** bir işlem yapılacaksa bu verinin bir **metni** temsil etmesi gerekir. **Mantıksal** bir sorguyu ifade etmesi istendiğinde de **doğru ya da yanlış** gibi değerleri ifade edecek tanımlamaların ayrıntılı olarak yapılması gerekir.

Örnek:

Soru
sayi_1=20 ve sayi_2=30 değişkenlerini oluşturarak sayi_1 ve sayi_2 değişkenlerinin toplamını toplam değişkenine aktaran ve toplam değişkenini ekrana yazdıran programın kodu
sayi_1=20 sayi_2=30 toplam=sayi_1+sayi_2 print(toplam)
SONUÇ: 50

Soru
sayi_1 ve sayi_2 değişkenlerinin birbirine eşit olmadığını kontrol eden ve bu kontrolün sonucunu (True veya False) ekrana yazdıran programın kodu
sayi_1=20 sayi_2=30 esit_degil=(sayi_1 != sayi_2) print(esit_değil)
SONUÇ: true

Soru
kelime= "Merhaba" metin değişkenini oluşturup ekrana yazdıran programın kodu
kelime="Merhaba"; print(kelime)
SONUÇ: Merhaba



Bu tanımlamalarla veri türlerinin çeşitliliği ortaya çıkar ve türler, bilgisayarda kaplayacakları alana göre kategorilendirilir. Bilgisayar ikilik sayı sistemine göre çalıştığı için **bilginin varlığı "1", yokluğu "0"** ile temsil edilir. Hafızada tutulan bu **"1" ve "0"** değerlerinin her birine **bit** denir ve ikili basamak anlamına gelen binary digit [baynıri dicit (ikili sayı)] kelimelerinin kısaltılmasından oluşur.

Veri türleri değişkenlerle eşleştiğinde Random Access Memory [RAM(rastgele erişimli bellek)] de kapasitelerine göre değişkenlerin alabileceği değerler belirlenir ve veriler bu değerlere göre alan kaplar. Sekiz tane "1" ve "0" değerinin yan yana gelmesiyle en temel hafıza birimi **byte (8 bit)** oluşur ve türlerin kapasiteleri **byte** cinsinden ifade edilir.

VERİ TÜRLERİ				
Sayısal Veri Türleri		**Metinsel Veri Türleri**		**Mantıksal Veri Türleri**
Tam Sayılar	**Ondalıklı Sayılar**	**Karakter**	**Karakter Dizisi**	**Boolean**
byte (1 byte)	float (4 byte)	char (1 veya 2 byte)	string (1 byte)	true (1 byte)
short (2 byte)	double (8 byte)			false (1 byte)
int (4 byte)	long double (12 byte)			
long (4 byte)				

Byte:

Bellekte 1 byte yer kaplar. Basit ve işlem sonuçlarının küçük kapasitelere ihtiyaç duyacağı durumlarda kullanılır. Byte veri türünün tanımlaması ***byte a=10;*** şeklindedir.

Örnek Byte Türünde Değişken Tanımlama:

```
byte a=10

byte sayi_1=55
```

Integer:

Yaygın kullanılan veri türüdür. 4 byte' lık yer kaplar. Byte veri türüne göre daha karmaşık ve büyük verilerle çalışma imkânı sağlar. Integer veya kısaltması olan int kelimeleriyle belirtilir. Integer veri türünün tanımlaması ***int a=100000; veya a=100000;*** şeklindedir.

Örnek integer Türünde Değişken Tanımlama:

int not_1

int not_1=100

int sayi_2=12000

Long:

Long [(uzun)], Integer veri türünden daha fazla kapasiteye sahiptir. 8 byte' lık yer kaplar. Integer veri türüne göre daha karmaşık ve büyük verilerle çalışma imkânı sağlar. Long veri türünün tanımlaması ***long a=1000000;*** şeklindedir.

Örnek Long Türünde Değişken Tanımlama:

Long buyuk_sayi=6000000

Long uzun_sayi=15000000

Float:

Float [(kesirli sayı)], 4 byte' lık yer kaplar. Ondalıklı sayı ifadelerin olduğu basit işlemlerde tercih edilir. ***Noktadan sonra altı karakter*** kullanımını sağlar. Bazı dillerde verinin sonuna **"f" ya da "F"** şeklinde belirtilerek kullanılır. Float veri türünün tanımlaması ***float a= 3.14f*** şeklindedir.

Örnek Float Türünde Değişken Tanımlama:

(bazı bölme işlemlerinin sonucu tam sayı çıkmayacağı için bölüm sonucunu ya da ortalama gibi değerleri genelde ondalıklı sayı olarak tanımlarız)

float ortalama=55.60

Float bolum_sonucu=23.5

Örnek
İki sayının toplamını bulan programı yazınız.
sayi_1=10 sayi_2=20 toplam=sayi_1+sayi_2 print("Toplam:" , toplam)
Ekran Çıktısı: Toplam: 30.0

Örnek:
Yarıçapı 10 cm, pi sayısı 3.14 olan dairenin alanını hesaplayan ve ekrana yazdıran programın kodunu ve ekran çıktısını yazınız.
float pi =3.14 int r=10 alan=pi*r*r print(“Dairenin Alanı:” , alan)
EKRAN ÇIKTISI: Dairenin Alanı:**314.0**

Boolean:

“True” ve “False” şeklinde iki değer alabilen ve kullanılan dile göre bu değerlerin yerini “1” ve “0” değerinin alabildiği bir veri türüdür. Boolean veri türü; kontrol ifadelerinde, mantıksal işlemlerde ve karar yapılarında kullanılarak işlemlerin gerçekleşmesi sağlanır.

Örnek:

Deger_1=”Merhaba” **Deger_2=””** **Deger_3=0** **Print(deger_1)** **Print(deger_2)** **Print(deger_3)**	 True False True

String:

Karakter dizisi veri türü olarak bilinir. Karakterler birleşerek metinleri oluşturur. Metinsel veri türlerinde karakter, sözcük, tümce, paragraf gibi metinlerin kullanıldığı durumlarda bu veri türü kullanılır. Karakterden fazla olan metinler için bazı dillerde char veri türü de kullanılmaktadır. Genelde metinler, **“ ” (çift tırnak)** içinde gösterilir.

Örnek:

Lise 12. Sınıf derslerini giriniz.
string ders_1=”Türk Dili ve Edebiyatı” **string ders_2=”Matematik”** **string ders_3=’Bilişim Teknolojileri ve Yazılım’** **…**

Char:

Char [(karakter)], metinsel veri türü olarak **0 ile 127** arasında değer alabilen ASCII karakter verilerinin kullanımı için kullanılır. **ASCII, American Standard Code for Information Interchange** kelimelerinin ilk harflerinin kısaltmasından oluşur ve Türkçe karşılığı, **Bilgi Değişimi için Amerikan Standart Kodu** anlamına gelen 7 bitlik karakter kümesidir. ASCII karakter kümesinde harfler, özel semboller, sayılar, kontrol karakterleri ve işaretler bulunur ve hepsinin sayısal bir karşılığı vardır. Bu sayılarla metinsel işleme işlemleri gerçekleştirilir ve bellekte tutulur.

Örnek:

Bir deneme sınavı kitapçık türü ve cevaplara göre cevap anahtarı ile karşılaştırılacaktır. İlgili verileri değişkenlerle nasıl tanımlarız.
char kitapcik_turu=”A” **char cevap_1=”C”** **char cevap_2=”E”** **char cevap_3=”D”**

Diziler:

Birden fazla aynı tür verinin sıralanarak tek bir isimle hafızada tutulmasını sağlayan yapılara **dizi** denir. Aynı türden birçok verinin aynı anda girişinin yapılması, işlenmesi ve bellekte tutulmasını sağlayan yapılar olarak kullanılan **diziler, array [(dizi)]** kelimesiyle de ifade edilir. İsminden de anlaşılacağı gibi verilerin dizildiği ve bir küme hâlinde tek bir isimle depolandığı belleklerdir. Dizi içinde sıralanacak veriler **"[ ]"** içinde belirtilerek yazılan sıralamaya göre 0'dan başlayarak numarandırılır ve bu numaralara **indis** denir. İndis numaralarına göre dizi elemanlarına ulaşım sağlanır.

Örnek:

int sayi[]={10,20,30,40}

İndis no	Sayi[0]	Sayi[1]	Sayi[2]	Sayi[3]
Değeri	**10**	**20**	**30**	**40**

Eleman sayısı belirtilerek tanımlanan bir "sayi" dizisi ise aşağıdaki şekilde de tanımlanabilir.

int[] sayi=new int[4];

sayi[0]=10;

sayi[1]=20;

sayi[2]=30;

3.4. KARAR YAPILARI

Bir şart belirleyerek bu şartın gerçekleşmesi ya da aksi durumuna göre işlem seçenekleri oluşturularak karar verme işlemi gerçekleştirilir.

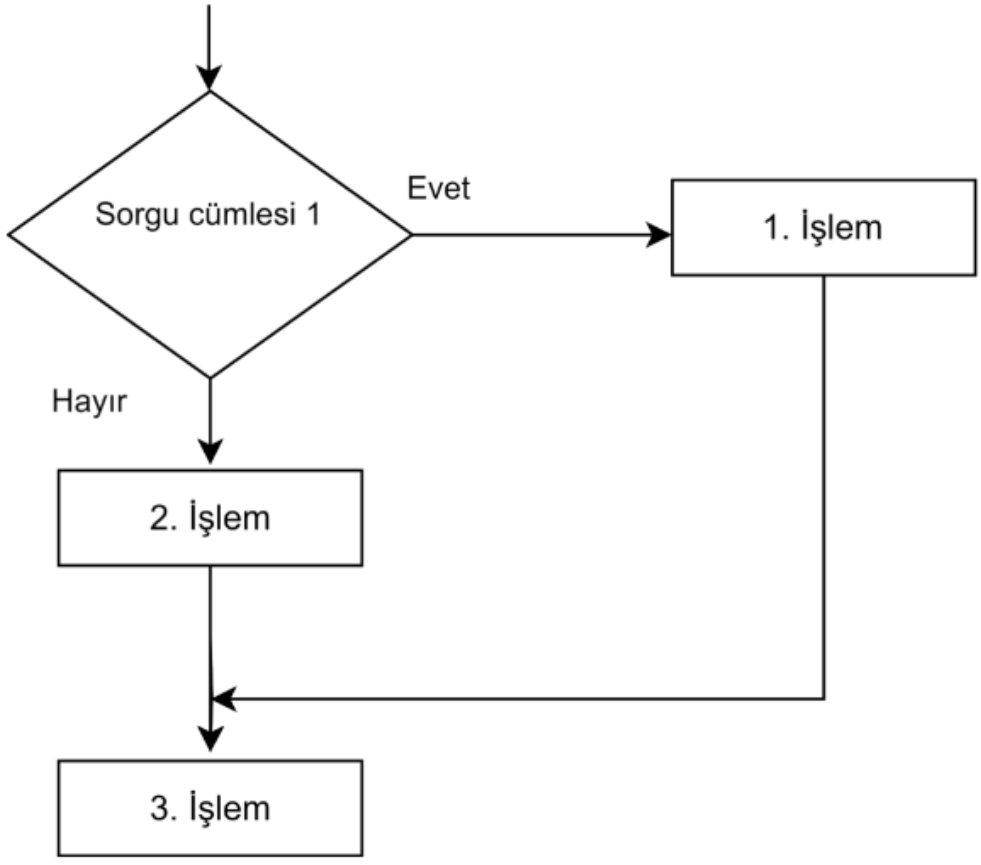
Gün içinde havanın durumuna göre kıyafet seçimi yapılırken bile karar mekanizmaları kullanılır. Havanın yağışlı olması durumunda yağmurluk giyme seçeneği olası çözüm gibi görünürken şemsiye kullanmak da yağmurda ıslanmamak için iyi bir çözüm yoludur.

Kullanılan sorgu ve karar verme sürecinin programlama dillerinde kullanılması için oluşturulan yapılara **karar yapıları** denir.

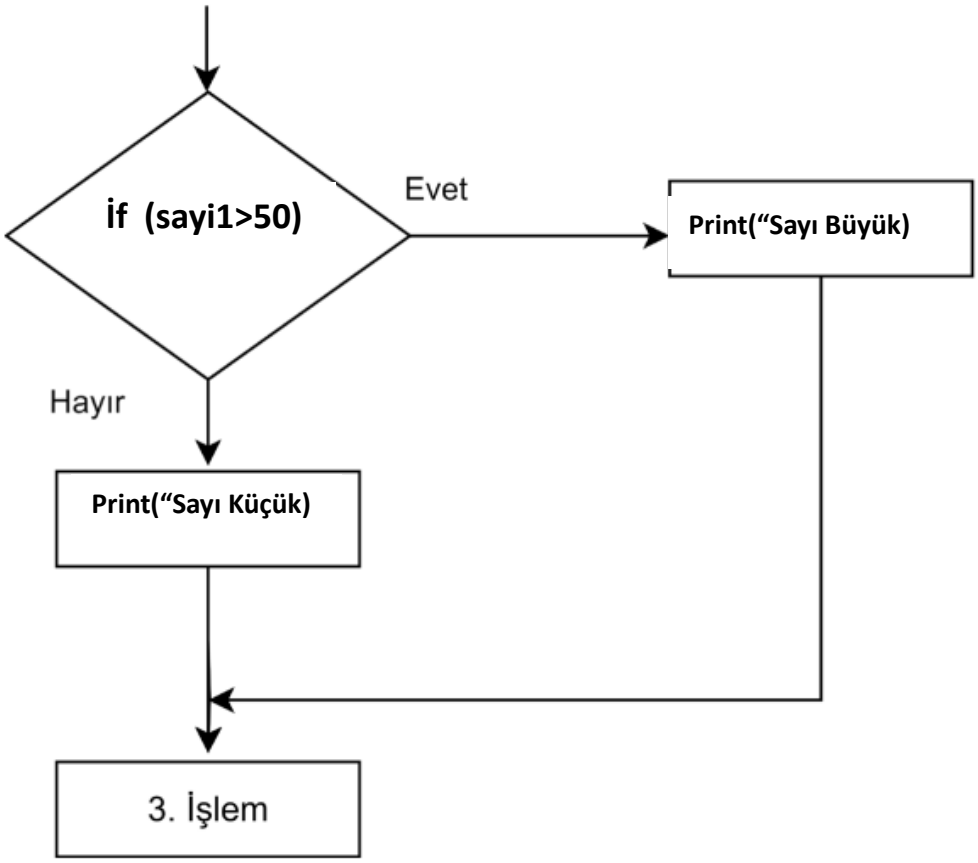
Mantıksal operatörler, karar yapılarının sorgularını oluşturmayı sağlar. Bu operatörler, iki veya daha fazla mantıksal sınamanın sonucunu "**True**" ve "**False**" değerleri ile gösterir.

3.4.1. If-else Yapısı

If ve **else** komutları birleşerek **if-else** yapısını oluşturur. If komutu ile belirtilen **sorgu** gerçekleştiğinde cevabın "**true**, evet" olması durumunda if karar yapısının blokları arasındaki işlemler gerçekleştirilir. Cevabın "**false**, hayır" olması durumunda else blokları arasındaki işlemler gerçekleştirilir. **Örnek** bir akış şeması içerisinde yer alan **if-else** bölümü:



Şema içerisinde örnek kodlarla akışı gösterelim:



If-else Yapısının Tanımlanma Çeşitleri

ÖRNEK - 1	ÖRNEK - 2	ÖRNEK - 3	PYTHON

if (sorgu)
{
1. İşlem
}
	if (sorgu)
	{
	1. İşlem
	}
	else
	{
	2. İşlem
	}
	3. İşlem
		if (sorgu)
		1.İşlem
		
		else
		
		2. İşlem
		3. İşlem
			if sorgu:
			1.İşlem
			
			elif sorgu:
			
			2. İşlem
			
			else:
			
			3. İşlem

Örnek Soru 1:

Öğrencinin girdiği iki sınavın not ortalaması 50 ve üzeri ise, devamsızlık süresi 10 günün altında ise geçti, değilse kaldı yazan programı yazınız. (Not 1=60, Not 2=80, devamsızlık=5)

Örnek 1 Program Kodu:

```
not1=60
not2=80
devamsizlik=5

ortalama=(not1+not2)/2

if (ortalama >= 50) and (devamsizlik<10):
        print("GEÇTİ")
else:
        print("KALDI")
```

NOT: **input** komutu kullanıcıdan veri almamızı sağlar. Bu komutla ekrana kullanıcıdan istenilen verinin ne olduğu yazılır ve ekranda kullanıcı veriyi girip enter tuşuna basana kadar beklenir.

Örneğin: (Önemli *)

```
adiniz = str(input("Adınızı giriniz:"))

sayi=int(input("Bir sayı giriniz:")) (Önemli * )

yas =int(input("yaşınızı giriniz:"))

pi_sayisi= float(input("Pi sayısı değerini giriniz:"))
```

Örnek Soru 2:

Bir basketbolcu, belirlediği iki zaman diliminde potadan geçirebildiği top sayılarının ortalamasını öğrenmek istemektedir. Basket sayılarının ortalamasının 60'a eşit ve yüksek olma durumunda ekrana "Performansınız çok iyi.", aksi durumda "Performansınız için çalışmalısınız!" yazan programın kodunu ve ekran çıktısını yazınız.

Örnek 2 Program Kodu:

```
basket_1 = int ( input ("Birinci sürede attığınız basket sayısını giriniz: "))
basket_2 = int ( input ("İkinci sürede attığınız basket sayısını giriniz: "))
ort = (basket_1 + basket_2) / 2
print("Basketbol Ortalamanız: ",ort)
if (ort >= 60):
print("Performansınız çok iyi.")
else:
print("Performansınız için çalışmalısınız!")
```

Örnek Soru 3:

Akıldan tutulan bir sayıyı tahmin etme oyununa ait program kodunu ve ekran çıktısını yazınız.

Örnek 3 Program Kodu:

```
tahmin_edilecek_sayi= 11
kullanici_tahmin = int(input("Aklımdan tuttuğum sayıyı tahmin et! "))
if tahmin_edilecek_sayi == kullanici_tahmin:
   print("Aferin! Doğru bildin...")
else:
   print("Ne yazık ki tahmin edemedin, tuttuğum sayı:",tahmin_edilecek_sayi)
```

Karar Yapısıyla Program Yazalım:

Bir şirket, stajyer öğrenci eğitimi programı için başvuru yapanları bazı kriterlere göre değerlendirip ön görüşmeye çağırmak istemektedir. Şirketin kriterleri, öğrencinin en az bir **programlama dili** bilmesi ve yabancı dilinin "**İngilizce**" olması şeklindedir.

"**Programlama dili biliyor musunuz?** ve "**İngilizce biliyor musunuz?**" sorularına evet "**E**", hayır "**H**" cevapları ile veri girişi yapmalarını sağlayan ve ekranda şartları sağlayanlar için "**Ön**

görüşme kriterlerine uygunsunuz.”, sağlamayanlar için “**Ön görüşme kriterlerine uygun değilsiniz.**“ mesajını ekrana yazdıran programın kodunu ve ekran çıktısını yazınız.

NOT: strip().upper() : girilen değeri büyük harfe çevirir. (e → E)

ÖRNEK 4

```
programlama_dili = input("Programlama dili biliyor musunuz? (E/H): ").strip().upper()
ingilizce_bilgisi = input("İngilizce biliyor musunuz? (E/H): ").strip().upper()
if programlama_dili == "E" and ingilizce_bilgisi == "E":
        print("Ön görüşme kriterlerine uygunsunuz.")
else:
        print("Ön görüşme kriterlerine uygun değilsiniz.")
```

Örnek 5

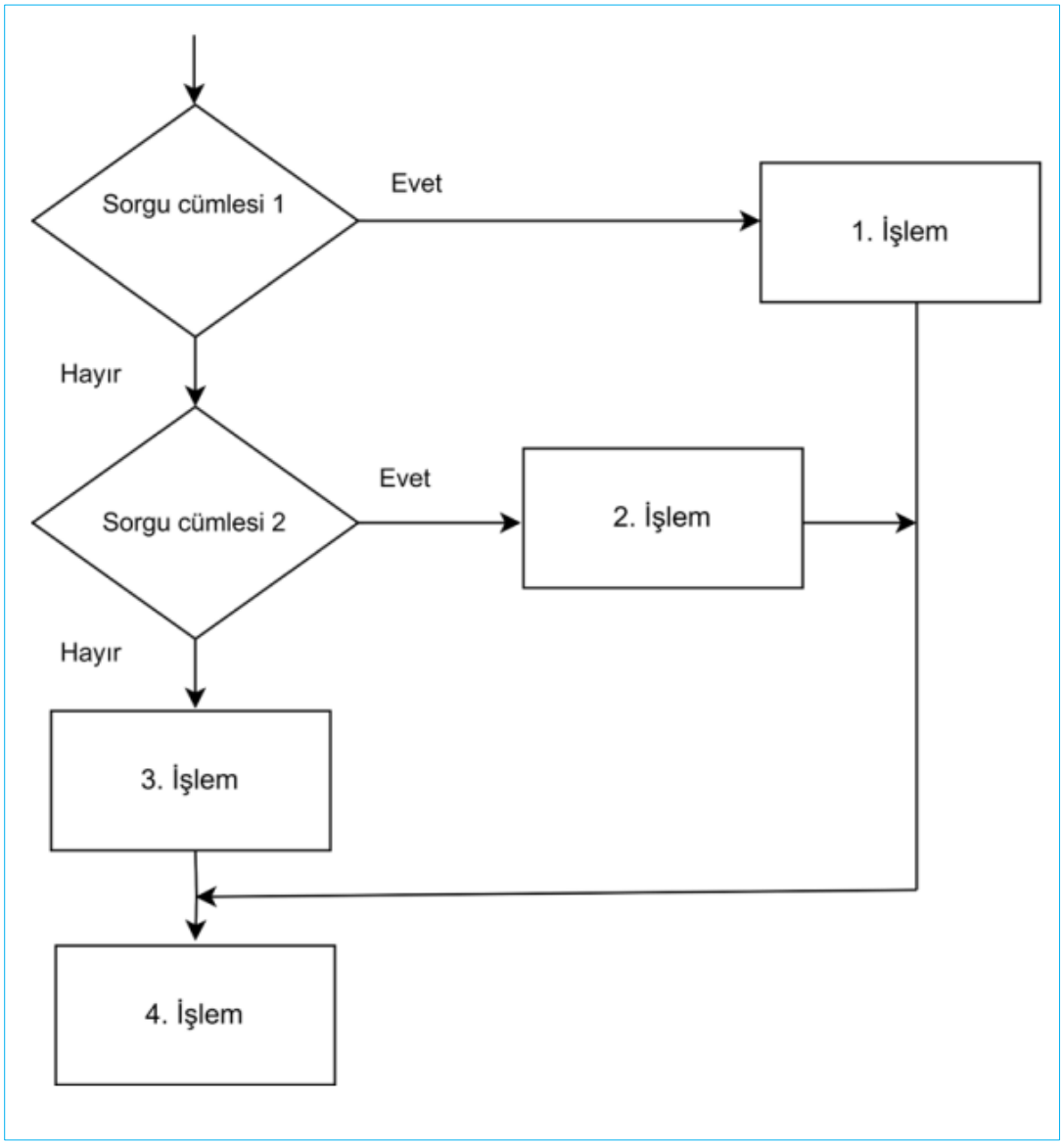
Yaşı 18’den büyük olup **VE İngilizce bilen** kişileri seçip geri kalanları eleyeceğimiz bir program kodu yazın.

```
yas=int(input("Lütfen yaşınızı giriniz: "))
ingilizce=str(input("İngilizce biliyor musunuz? E/H : ").strip().upper())
if (yas>18) and ( ingilizce=="E"):
   print("Ön elemeyi geçtiniz.")
else:
   print("Üzgünüz, ön elemeyi geçemediniz.")
```

.strip() : metnin sağında ve solundaki boşlukları siler.
.upper() : yazılan metni büyük harfe çevirir.

3.4.2. Else If Yapısı

Karar yapılarında kullanılan en karmaşık yapılardan biridir. If bloku ile başlayan sorgu, sonrası alınan cevabın “true, evet” olması ile if blokunun içindeki işlem gerçekleştirilir. Alınan “false, hayır” cevabı sonrasında da else if yapısı ile yeni bir sorgu oluşturulur. Sorgudan gelen cevaba göre işlem ya da gerekli ise yeni bir sorgu tanımlanarak çoklu seçim yapısı oluşturulur.



Else If Yapısının Tanımlanma Çeşitleri

ÖRNEK - 1	ÖRNEK – 2	PYTHON

if (sorgu1)
{
1.İşlem
}
else if (sorgu2)
{
2. İşlem
}
else
{
3. İşlem
}
4. İşlem
	if (sorgu1)
	1.İşlem
	else if (sorgu2)
	2. İşlem
	else
	3. İşlem
	4. İşlem
		if sorgu1:
		1.İşlem
		elif sorgu2:
		2. İşlem
		else:
		3. İşlem
		4. İşlem

NOT: Burada program komutları (**if, elif, else**) dışındaki ifadeler girintili olmak zorundadır.

Örnek Soru 1:

Bir sayının pozitif mi negatif mi olduğunu ulan programı yazalım.

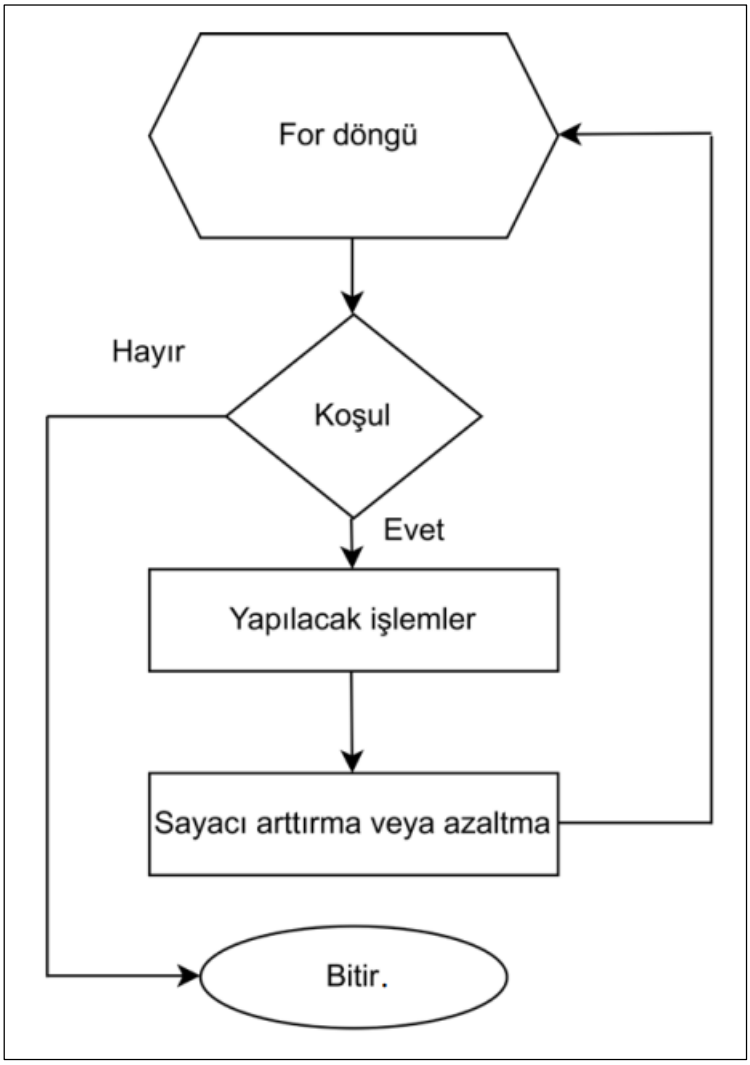
```
Sayi = int(input("Bir sayı giriniz:")) (Önemli * )

if sayi > 0:
   print(sayi,"pozitif bir sayıdır.")
elif sayi == 0:
   print(sayi," ne pozitif ne de negatif bir sayıdır.")
else:
   print(sayi,"negatif bir sayıdır.")
```

3.5. DÖNGÜ YAPILARI

Yazılan programlama komutlarının tekrarlı şekilde çalıştırılması gereken durumlar oluşabilir. Komutların tekrar yazılmadan çalıştırılıp programın işlevselliğini sağlamak için belirli şartlar oluşturulur ve komutların içinde döngüsel bir çalışma akışı sağlanır. Böylelikle komut satırı fazlalığının önüne geçilerek hem programın gereksiz satırlardan arındırılması hem de istenen sonuca daha hızlı ulaşılması sağlanır. Programcının iş yükünü azaltıp daha sade bir görünüm elde edilmesini sağlayan bu tekrarlı yapılara **döngü yapıları** denir.

Kod blokları arasına döngüden çıkma komutları yazılmadığı sürece belirlenen şartlarda işlemler gerçekleştirilir.

3.5.1. For Döngüsü



Bir şarta bağlı olarak tekrarlanma sayısının en başta tanımlandığı döngü yapısıdır. Tanımlamalar ilk satırda belirtildiği için şartın gerçekleşme durumu kadar for blokları arasında yazılan kodlar çalıştırılır. Python dilinde "**for**" döngüsü ile genelde **range** [(aralık)] deyimi kullanılarak döngünün tekrar etme sayısı belirtilir.

Önemli Not:

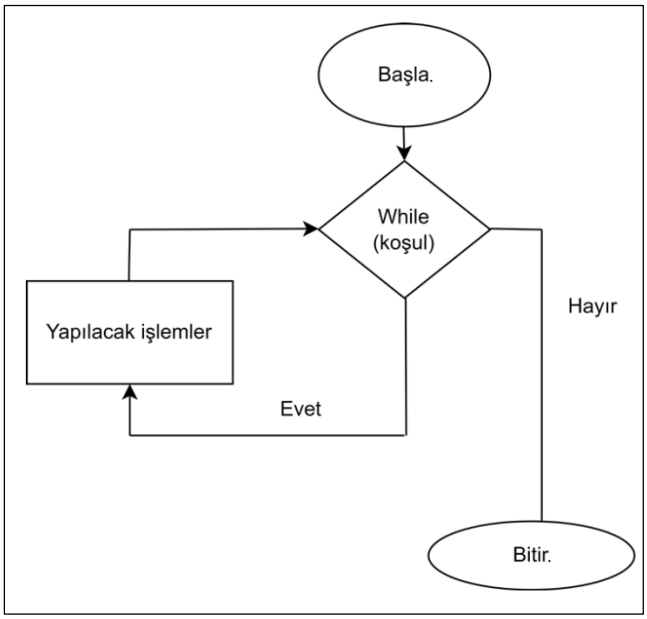
Döngü yapıları ile beraber kullanılan **sayaç** değişkenleri, döngü yapısı içindeki işlemlerin tekrar etme sayısının kontrolünü sağlar. Genellikle "**i, j**" karakterleri sayaç ismi olarak kullanılır. Farklı bir aktarım olmadığı sürece sayaç=0 dan başlar.

Örnek 1: Ekrana “Programlama” kelimesini 5 defa alt alta yazdıran programın kodunu ve ekran çıktısını for döngüsü kullanarak yazınız.	
Çözüm	**Ekran Görüntüsü**
for i in range(5): **print(“Programlama”)**	**Programlama** **Programlama** **Programlama** **Programlama** **Programlama**

Örnek 2: Belirlenen aralıkta 0’dan başlayarak 10’a kadar olan sayıları toplayan ve ekrana yazdıran for döngülü programın kodunu ve ekran çıktısını yazınız.	
Çözüm	**Ekran Görüntüsü**
sonuc=0 **for sayi in range(11):** **sonuc=sonuc+sayi** **print(“Toplam=”,sonuc)**	**Toplam=55**

(Burada dikkat edilmesi gereken döngü içerisinde sayaç değişkeni (sayi) sıfırdan “0” başlayarak her dönüşte 1 artar ve range (aralık) sınırına kadar gelir.)

3.5.2. While Döngüsü



Belirtilen şart sağlanana kadar döngünün tekrar edilmesiyle oluşan **şartlı döngülerdir**. Şart sağlandığı zaman while blokları arasındaki işlemler **bir defa gerçekleşir** ve şart sağlandığı sürece bu işlem devam eder. Genellikle döngünün kaç defa gerçekleşeceğinin belli olmadığı durumlarda kullanılır.

While Örnek 1: Ekrana "İhsan Mermerci Anadolu Lisesi" kelimesini 5 defa alt alta yazdıran while döngülü programın kodunu ve ekran çıktısını yazınız.	
Çözüm	**Ekran Görüntüsü**
isim = "İhsan Mermerci Anadolu Lisesi" i=0 while (i<5): print(isim) i=i+1	İhsan Mermerci Anadolu Lisesi İhsan Mermerci Anadolu Lisesi İhsan Mermerci Anadolu Lisesi İhsan Mermerci Anadolu Lisesi İhsan Mermerci Anadolu Lisesi

While Örnek 2: Belirlenen bir aralık belirtilerek 0' dan başlayarak 10' a kadar olan sayıları toplayıp ekrana yazdıran while döngülü programın kodunu ve ekran çıktısını yazınız.	
Çözüm (Önemli *)	**Ekran Görüntüsü**
i=0 toplam=0 while (i<11): toplam=toplam+i i=i+1 print("Toplam =", toplam)	Toplam=55

3.5.3. Sayaç Yapısı

Döngü yapıları ile belirlenen işlemlerin **tekrar etme sayısı sayaç denilen değişken tanımlamaları ile kontrol edilir**. Sayaç değişkeni ile şart oluşturularak döngünün kontrol edilmesi sağlanır.

Ekrana 0'dan 5'e kadar (5 dâhil) olan sayıları ekrana yazdıran programın farklı döngü kullanılarak yazılmış kodları

For döngüsü ile kodlar yazılırken sayi adında değişken oluşturulup range fonksiyonu kullanılmıştır. (Önemli *)	**While döngüsü ile kodlar yazılırken "i" sayaç olarak tanımlanmıştır.**	**Ekran Çıktısı**
for sayi in range(6): print(sayi)	i=0 while (i<6): print(i) i=i+1	0 1 2 3 4 5

SORU:

Kullanıcıdan 10 tane sayı alıp, her bir sayının tek mi, çift mi olduğunu kontrol eden ve ekrana yazan program kodlarını yazınız. **(Önemli *)**

CEVAP:

```
while (i<11):
    sayi=int(input("Bir sayı giriniz:"))
    if (sayi%2==0):
        print(sayi," sayısı çifttir")
    else:
        print(sayi," sayısı tek sayıdır")
    i=i+1
```

2. sınava ek olarak sadece 1 konumuz var. Teorik sorunuz buradan gelecek

4.1. TASARLANAN ALGORİTMA VE AKIŞ DİYAGRAMLARININ TEST EDİLMESİ

4.1.2 Algoritma Testi

Algoritma testi, algoritmanın belirlenen girdilerle beklenen çıktıları verip vermediğini kontrol etmeyi amaçlar. Bu testler; algoritmanın mantıksal hatalarını, yanlış varsayımlarını veya beklenmedik durumlarda nasıl davrandığını tespit etmeye yardımcı olur. **(Önemli *)**

Algoritma Testinde Kullanılan Araçlar

Beyaz Kutu Testi [white box testing]:

Algoritmanın iç yapısını ve çalışma mantığını dikkate alarak yapılan testlerdir. Bu testte algoritmanın her bir adımı ve karar noktası incelenir.

Siyah Kutu Testi [black box testing]:

Algoritmanın iç yapısına bakılmadan sadece giriş ve çıkış değerlerine odaklanılarak yapılan testlerdir. Algoritmanın verilen girdilere karşılık doğru çıktılar üretip üretmediği kontrol edilir.

Birleşim Testi [integration testing]:

Birden fazla algoritmanın veya modülün birlikte nasıl çalıştığını kontrol etmek için yapılan testlerdir. Algoritmaların birbiriyle uyumlu çalışıp çalışmadığı incelenir.

Sınır Değer Analizi [boundary value analysis]:

Algoritmanın sınır değerlerde nasıl davrandığını kontrol etmek için yapılan testlerdir. Minimum ve maksimum değerler kullanılarak algoritmanın performansı test edilir.

2. Dönem Sınavı İçin Örnek Python Programlama Örnekleri

ÖRNEK – 1:

Aşağıda bir kişinin yaşına göre kategorisini (çocuk, genç, yetişkin, yaşlı) belirleyen algoritma verilmiştir. Bu algoritmanın programını yazınız.

ALGORİTMA

1. Adım: Yaşı giriniz.

2. Adım: Eğer yaş < 13 ise “Çocuk“ yazdırınız.

3. Adım: Eğer yaş >= 13 ve yaş < 18 ise “Genç“ yazdırınız.

4. Adım: Eğer yaş >= 18 ve yaş < 65 ise “Yetişkin“ yazdırınız.

5. Adım: Eğer yaş >= 65 ise “Yaşlı“ yazdırınız.

6. Adım: Bitir.

PROGRAM KODU:

```
yas=int(input("Lütfen yaşınızı giriniz:"))
if (yas<13):
        print("Çocuk")
elif (yas>=13) and (yas<18):
        print("Genç")
elif (yas>=18) and (yas<65):
        print("Yetişkin")
else:
        print("Yaşlı")
```

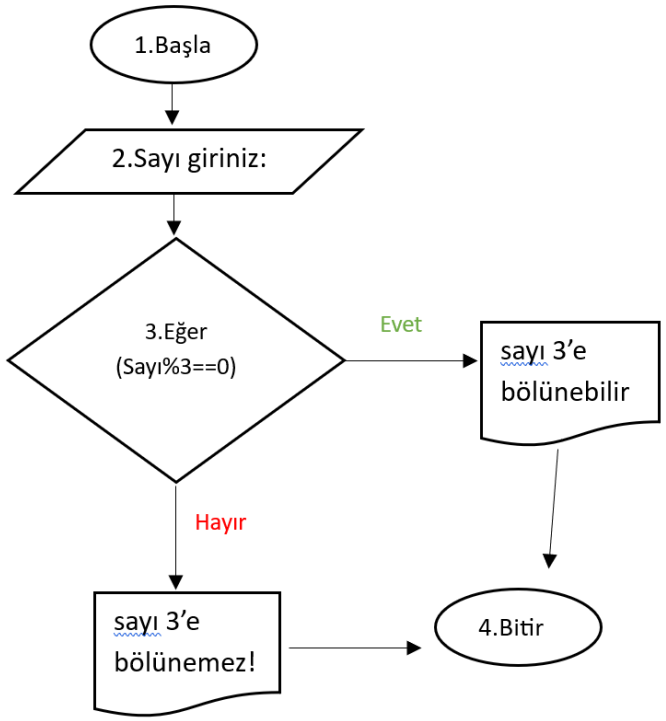
ÖRNEK – 2:

Kullanıcıdan sayı girişi yapmasını isteyen ve sayı 3'e bölünebilir bir sayı ise "Sayı 3'e bölünebilir"; değil ise "Sayı 3'e bölünemez!" mesajını ekranda gösteren akış diyagramı, algoritma ve Python program kodlarını yazınız.

ALGORİTMA

1. Adım: Başla
2. Adım: Kullanıcıdan sayı girmesi istenir.
3. Adım: Eğer sayının 3'e bölümünden kalan 0 ise, sayı 3'e bölünebilir yazdırılır. Değilse Sayı 3'e bölünemez yazdırılır.
4. Adım: Bitir

Akış Diyagramı



PROGRAM KODU: (Önemli *)

```
sayi=int(input("Sayı giriniz: "))
if (sayi%3==0):
    print("Sayı 3'e bölünebilir.")
else:
    print("Sayı 3'e bölünemez!")
```

ÖRNEK -3:

0'dan başlayarak kullanıcıdan bir sınır sayısı belirlemesini isteyen ve o sayıya kadar olan sayılardan çift olanlarının toplamını "Çift sayıların toplamı " şeklinde ekrana yazdıran Python program kodlarını yazınız.

PROGRAM KODU: (Önemli *)

```
sinir_sayi=int(input("Sınır için bir sayı giriniz:"))

toplam=0

for i in range(sinir_sayi):

        if (i % 2 == 0):

                toplam= toplam + i

print("Çift sayılar toplamı=",toplam)
```

ÖRNEK -4:

Kullanıcıdan iki tane sayı ve bir operatör (- çıkarma, + toplama) girmesini sağlayan ve operatöre göre çıkarma, toplama işlemlerini yapan ve sonucu ekrana yazdıran; belirtilmeyen bir sembol girildiğinde "Tanımlı işlem giriniz!" mesajını ekrana yazdıran Python program kod kodunu yazınız.

PROGRAM KODU:

```
sayi_1=int(input("1.sayıyı girin:"))

sayi_2=int(input("2.sayıyı girin:"))

islem=str(input("Yapmak istediğiniz işlemi seçin (-,+):"))

if (islem=="-"):

        print("Çıkarma", sayi_1-sayi_2)

elif (islem=="+"):

        print("Toplama: ", sayi_1 + sayi_2)

else:

        print("Tanımlı İşlem giriniz!")
```

ÖRNEK – 5:

Girilen sayının FAKTÖRİYELİNİ hesaplayan programı yazınız.

PROGRAM KODU:

FOR DÖNGÜSÜ İLE

```
sayi= int(input("Faktöriyeli alınacak sayıyı giriniz: "))
faktoriyel = 1
i=1
if sayi<0:
  print("Üzgünüz, negatif sayıların Faktöriyeli alınamaz.")
elif sayi==0:
  print("0!=1")
else:
        for i in range(1,sayi+1):
                faktoriyel=faktoriyel *i
        print(sayi,"!= ", faktoriyel)
```

WHILE DÖNGÜSÜ İLE

```
sayi= int(input("Faktöriyeli alınacak sayıyı giriniz: "))
faktoriyel = 1
i=1
if sayi<0:
        print("Üzgünüz, negatif sayıların Faktöriyeli alınamaz.")
elif sayi==0:
        print("0!=1")
else:
        while (i<sayi+1):
                faktoriyel=faktoriyel *i
                i=i+1
        print(sayi,"!= ", faktoriyel)
```

2. SINAV İÇİN ÖRNEK SORULAR

(En Önemli *)

ÖRNEK SORULAR VE CEVAPLAR Yazılım geliştirme sürecinde kodlamaya geçmeden önce tasarlanan algoritma ve akış diyagramlarının mantıksal olarak incelenmesi gerekmektedir. Buna göre "Algoritma Testlerinde" kullanılan 4 test yönteminin isimlerini yazınız. ***(15 puan)***

1- ……..

2- ……..

3- ……..

4- ……..

CEVAP 1:

1- Beyaz Kutu Testi

2- Siyah Kutu Testi

3- Entegrasyon Testi

4- Sınır Değer Analizi

1. Aşağıdaki mantıksal operatörlerin karşılaştırma sonuçlarını karşısına yazınız (TRUE/FALSE)?
X= 100 ve Y = 200 ***(15 puan)***

Koşul ifadesi	**Sonuç**
(X == Y)	
(X != Y)	
(X > Y)	

CEVAP 2:

Koşul ifadesi	**Sonuç**
(X == Y)	False
(X != Y)	True
(X > Y)	False

2. 0'dan başlayarak 5000'ye kadar sayıları ekrana yazdıran (**for döngüsü ile**) Python program kodlarını yazınız. ***(10 puan)***

CEVAP 3:

```
for sayac in range(5001):
        print(sayac)
```

3. 0' dan başlayarak 150' e kadar olan sayılardan sadece 3'e bölünenleri ekrana yazdıran PYTHON programın kodunu (**while döngüsüyle**) yazınız. ***(20 puan)***

CEVAP 4:

```
while ( i < 151 ) :
        if ( i % 3 = = 0 ):
                print( i )
                i = i + 1
```

4. Kullanıcıdan bir sayı girişi yapmasını isteyen ve girilen sayının pozitif mi negatif mi olduğunu ekranda yazdıran program kodlarını yazınız. ***(20 puan)***

CEVAP 5:

```
sayi=int(input("Sayı giriniz: "))
if (sayi>0):
        print("Sayı pozitiftir.")
else:
        print("Sayı negatiftir")
```

5. Kullanıcıdan bir sınır sayısı belirlemesini isteyen ve 0'dan başlayıp sınır sayıya kadar olan sayılardan tek olanlarının toplamını "**Tek sayıların toplamı**= " şeklinde ekrana yazdıran Python program kodlarını yazınız. ***(20 Puan)***

CEVAP 6:

```
sinir_sayi=int(input("Sınır için bir sayı giriniz:"))

for sayac in range(sinir_sayi):

        if ( sayac % 2 == 1):

                toplam =  toplam + sayaç

print("Tek sayılar toplamı=",toplam)
```