Python programlama dilini öğrenebilmeniz için bu ders notu hazırlanmıştır. Ders notunu dikkatlice okuyup bir defa(öğrenme stilinize göre iki defa) defterinize yazınız. Verilen örnek programları telefonunuza yüklediğiniz python3 ide editöründe yazıp çalıştırınız. Örnekleri yazarken ilkin bire bir aynı yazıp denedikten sonra kendinizi geliştirmek için örnek üzerinde değişiklik yapın. İnternetten rastgele kod alıp çalıştırmayın kendinize veya başkalarına zarar verebilirsiniz.

Değişken Nedir : Yazdığımız uygulamalarda, projelerde kimi zaman bir sayıyı, kimi zaman bir isimi, haftanın günlerini, Türkiye'nin illerini, kayıt olan kişinin soyadını yapılan işleme ve amaca göre geçici bir süreliğine hafızada tutmak isteyebiliriz.

Programcılıkta kullandığımız dilin izin verdiği veri tipleri çerçevesinde istediğimiz verileri, bilgileri saklamamızı sağlayan yapılara değişken (variable) denilmektedir. Değişkenlere verdiğimiz isimler ile erişebilir, birbirinden ayırabilir, işlemlerimizi yapabilir. Kurallar çerçevesinde değişkenlere verdiğimiz isimlerin bilgisayar için bir önemi yoktur, sadece kodun okunabilirliği ve anlaşılabilir olması için yazılımcılar için önemlidir.

Programlama dilinde işlediğimiz verileri bilgisayarın hafızasında tutmak için yapmış olduğumuz tanımlamalardır.

Python da değişken tanımlama : Değişkenler, değerleri saklamak için ayrılmış bellek konumlarından başka bir şey değildir. Kısaca, bir değişken oluşturduğunuzda hafızada yer ayırmanız anlamına gelir.

Bir değişkenin veri türüne bağlı olarak, yorumlayıcı hafızayı ayırır ve ayrılmış hafızada neyin saklanabileceğine karar verir. Bu nedenle, değişkenlere farklı veri türleri atayarak, değişkenlere tam sayı, ondalıklı sayı veya karakter(yazı-metin[string]) kaydedebilirsiniz.

Python'da değişkenler nesnelerden oluşmaktadır ve kullanmadan önce de tanımlama zorunluluğu yoktur. Ne zaman değişkeni kullanırsanız ve değer atarsanız statik olmayan bir şekilde değişken tanımlanacaktır. Değişken içindeki verinin tipi değiştiğinde de dinamik olarak tip değişecektir.

```
1  tamsayi=27
2  ondalikli =27.0
3  metin="Bilgisayar Bilimi"
4  okul="Nizip Ticaret Borsası Anadolu Lisesi"
5  okulNo=2744
6  okulPuanı=85.50
7
```

Değişkene veri atamak için = operatörü kullanılır. Yukarıda tamsayı, ondalıklı, metin, okul, okulNo ve okulPuanı gibi değişkenler tanımlanmıştır Bu değişkenlere = operatörü ile değerler atanmıştır. Bir kişinin adını ve soyadını hafızaya alıp saklamak istediğimizde bir değişken ve o değişkene ait değer tanımlarız

```
adSoyad= "Şehabettin ŞAHBAZ"
```

İlk kısım değişkenin adıdır. Değişken adı değişken ile ilgili herşeyi temsil eder.	İkinci kısım değişkenin taşıdığı değerdir. Değer hangi veri türünde ise değişken o veri türünde olur.
Değişkene değer aktarmak için = operatörü kullanılır	

VERİ TÜRLERİ

A) **String(str)** : yazı, metin . Yazılabilen her şey. Tırnak içinde yazılması şart. Programlama dilinin veriyi string veri türünde kabul etmesi için `"değer tırnak içinde yazılmalıdır"`
String veri tipinde oluşturduğunuz değişken veya değerler **str** sınıfına kaydolur.

```
>>> type("merhaba")
<class 'str'>
```

type() fonksiyonu değişken veya değerin hangi veri tipinde olduğunu öğrenmemize yarar.

B) SAYI

1) **integer(int):** tam sayılardır. karakter kümesi içindekilerin yanyana yazılması ile oluşan bütün sayıları kapsar.
Karakter kümesi={-,+,0,1,2,3,4,5,6,7,8,9}
Ör:

```
x=1
```

x değişkenine 1 tamsayı değeri atanmış

```
y=-1656
```

y değişkenine -1656 tamsayı değeri atanmış

`>>> type(x)` `<class 'int'>`	**İnteger** veri tipinde oluşturduğunuz değişken veya değerler **int** sınıfına kaydolur. x'in değeri int sınıfında yer alan bir değer, yani x tamsayı
`>>> type(27)` `<class 'int'>`	Bütün tam sayı değerleri int sınıfında değerlerdir.Tam sayılar ve değişkenler tırnak içinde yazılmazlar. Sadece stringler tırnak içinde yazılır.

2) **float(float)** : ondalıklı sayılardır. Karakter kümesi ={ -+0,1,2,3,4,5,6,7,8,9 ve nokta(.) }

nokta(**.**) ondalıklı kısmı ayırmak için kullanılır. 10**.**5 gibi

not: sayılar tırnak içinde yazılmazlar. Tırnak içinde yazılırsa sayı olmaktan çıkar string'e dönüşür.

`>>> ort=85.571`
`>>> type(ort)`
`<class 'float'>`
	ort değişkenine 85.571 ondalıklı sayısı değer olarak atanmış. Ondalıklı sayılar **float** sınıfında yer alır.

C) Mantıksal Veri **boolean(bool) : True**(doğru) ve **False**(yanlış) olarak iki değer alabilir. Tırnak içinde yazılmaz. Karsılaştırma işlemlerinin doğal sonucu mantıksal veri türüdür.

`>>> geçmeDurumu= True`
`>>> type(geçmeDurumu)`
`<class 'bool'>`
	GeçmeDurumu değişkeni mantıksal veri türünde değer taşıdığı için **bool** sınıfında yer alır

`>>> 5>3`
`True`
`>>> type(5>3)`
`<class 'bool'>`
	Bütün karar-karşılaştırma işlemleri mantıksal veri türünde değer dönderir. Yani sonuçları **True** yada **False** dir. **bool** sınıfında yer alırlar.

OPERATÖRLER

Python'da aritmetik(matematik), karar-karşılaştırma(ilişkisel) ve mantıksal işlemler için operatör (işleç) adı verilen semboller ve özel sözcükler kullanılır.

			x = 20 y = 5	sonuc
+	Toplama		sonuc = x + y	25
-	Çıkarma		sonuc = x - y	15
*	Çarpma		sonuc = x * y	100
/	Bölme		sonuc = x / y	4.0
%	Mod Alma		sonuc = y % x	0
//	Tam Bölme		sonuc = x // y	4
**	Üs alma		sonuc = 2 ** 3	8

1) ARİTMETİK OPERATÖRLERİ

Toplama, çıkarma, çarpma ve bölme gibi işlemler başta olmak üzere aritmetik işlemleri yapmak için kullanılan operatörlerdir

Örneklerin çözümünü önce siz yapın sonra çözüme bakın

Toplama Operatörü (+)

```
x = 10
y = 20
toplam = x + y + 20
print(toplam)
```

Değişkenlerin değerleri ile 20 toplanır ve toplam değişkenine 50 değeri atanır.

Çıkarma Operatörü (-)

```
x = 10
y = 5
sonuc = x – y
print(sonuc)
```

x' den y çıkarılınca 5 değeri kalır ve sonuc değişkenine atanır.

Çarpma Operatörü (*)

```
x = 5
y = 3
z = (x + y) * (y - x)
print(z)
```

Parantez içlerindeki (x+y)' nin değeri 8 ile (y-x)' nin değeri -2 çarpılır ve sonuç z değişkenine -16 olarak eşitlenir.

Bölme Operatörü (/)

```
x = 5
y = 2
z = (x * y) / y
print(z)
```

Parantez içindeki (x*y)' nin değeri 10 sayısı y' nin değeri olan 2' e bölünür ve sonuç 5 olarak z değişkenine aktarılır.

Mod Alma Operatörü (%)

```
x = 1001
y = 2
z = x % y
print(z)
```

x değişkeninin değeri olan 1001 sayısının 2' ye bölümünden elde edilen kalan sayı 1' dir.

Tam Bölme Operatörü (//)

```
x = 15
y = 2
z = x // y
print(z)
```

x' in y değrine kalansız bölümünden çıkan sonuç 7 değeri z değişkenine atanır.

Üs Alma Operatörü (**)

```
x = 2
y = 5
z = x ** y )# 2*2*2*2*2
print(z)
```

2' nin 5 defa yan yana çarpımından elde edilen 32 değeri z değişkenine aktarılır.

2) **İLİŞKİŞEL OPERATÖRLER (KARAR – KARŞILAŞTIRMA OPERATÖRLERİ)**

Değişkenler veya değerler arasında büyüklük, küçüklük, eşitlik gibi karşılaştırma işlemleri yapabilen operatörlerdir.

İlişkisel operatörleri okurken ……mi? Şeklinde okursak doğru sonuca ulaşmamız daha kolay olur.

Ör: 5==3 işlemi için "beş üç'e eşit mi"

İşaret	İşlem	OKUNUŞU	Örnek	Sonuç
==	Eşit mi?	***Beş üç'e eşit mi?***	5==3	False
!=	Farklı mı?	***Beş üç'ten farklı mı?***	5!=3	True
>	Büyüktür?	***Beş üç'ten büyük mü?***	5>3	True
<	Küçüktür?	***Beş üç'ten küçük mü?***	5<3	False
>=	Büyük veya eşittir?		3>=3	True
<=	Küçük veya eşittir?		5<=3	False
is	Değerler eşit mi?		'Elif' **is** 'elif'	False
is not	Değerler farklı mı?		'Elif' **is not** 'elif'	True
in	İçeriyor mu?		'bil' **in** 'bilişim'	True
not in	İçermiyor mu?		'bil' **not in** 'bilişim'	False

Örneklerin çözümünü önce siz yapın sonra çözüme bakın

Örnek

```
a = 10
b = 20
sonuc = (a<b)
print(sonuc)
```

a, b' den küçük mü diye sorduğumuzda geriye doğruysa True yanlış ise False değeri gelir. Burada a, b' den küçük olduğundan dolayı True bilgisi gelir.

Örnek

```
a = 20
b = 20
sonuc = (a!=b)
print(sonuc)
```

Burada ise; a, b'ye eşit değil mi diye soruyoruz ve eşit olduğundan dolayı False bilgisi gelir.

Örnek

```
a = 20
b = 20
sonuc = (a=<b)
print(sonuc)
```

Burada ise; a, b' den küçük ya da eşit mi diye soruyoruz ve küçük değil ancak eşit olduğundan dolayı True bilgisi gelir.

Örnek

Doğum yılına göre yaşı 18 ve üstü olan bir kişi ehliyet alabilir. Dolayısıyla doğum yılına göre yaşını hesapladığımız bir kişinin ehliyet alıp alamayacağını (>=) büyük eşit operatörü ile kontrol edebiliriz.

```
dogumYili = 1999
yas = 2020 - dogumYili # 21
ehliyet = (yas >= 18)
```

ehliyet değişkeninin değeri True olacağından dolayı kişi ehliyet alabilir. Çünkü (2020 - 1999) bize 21 yaşını verir ve (21>=18) karşılaştırması bize True değerini döndürür.

Python Karşılaştırma Operatör Örnekleri

1- Girilen 2 sayıdan hangisi büyüktür ?

```
a = int(input('a: '))
b = int(input('b: '))
result = (a > b)
print(f'a: {a} b: {b} den büyüktür: {result}')
```

NOT: Yukarıda ki programda - `print(f'a: {a} b: {b} den büyüktür: {result}')` – satırında string biçimlendirme yapılmıştır. String biçimlendirme ile ilgili ayrıntılı bilgi sayfa7 de String Formatlama

2- Kullanıcıdan 2 vize (%60) ve final (%40) notunu alıp ortalama hesaplayınız. Eğer ortalama 50 ve üstündeyse geçti değilse kaldı yazdırın.

```
vize1 = float(input('1. vize: '))
vize2 = float(input('2. vize: '))
final = float(input('final : '))
ortalama = (((vize1 + vize2) / 2) * 0.6) + (final * 0.4)
print(f'not ortalamanız : {ortalama} ve dersten geçme durumunuz: {ortalama>=50}')
```

3- Girilen bir sayının tek mi çift mi olduğunu yazdırın.

```
sayi = int(input('sayı: '))
tekcift = (sayi % 2 == 0)
print(f'girilen sayı çift olma durumu: {tekcift}')
```

4- Girilen bir sayının negatif pozitif durumunu yazdırın.

```
sayi = int(input('sayı: '))
pozitifmi = (sayi > 0)
print(f'girilen sayının pozitif olma durumu: {pozitifmi}')
```

5- Parola ve email bilgisini isteyip doğruluğunu kontrol ediniz. (email: email@sadikturan.com parola:abc123)

```
email = 'email@sadikturan.com'
password = 'abc123'
girilenEmail = input('email: ')
girilenPassword = input('parola: ')
isEmail = (email == girilenEmail.lower().strip())
isPassword = (password == girilenPassword.lower())
print(f'Email bilgisi doğrumu: {isEmail} ve Parola doğru mu: {isPassword}')
```

3) MANTIKSAL OPERATÖRLER

Pythonda mantıksal operatörleri birden fazla koşulu beraberce değerlendirmek için kullanırız.

Operatör	Açıklama	Kullanım	Sonuç
and	ve operatörü	(8 < 10) and (6 > 5)	True
or	veya operatörü	(5 == 5) or (6 == 5)	True
not	değil operatörü	not(5 == 5)	False

Python And (ve) Operatörü

Python and operatörü ile belirtilen koşulların hepsi doğru ise True, en az bir tanesinin yanlış olduğu durumda ise False değer üretir. Örnek

Ehliyet almak için en az lise mezunu olmak ve aynı zamanda yaş bilgisinin en az 18 olması gibi 2 koşulun aynı anda doğruluğunu mantıksal and operatörü yardımıyla değerlendirebiliriz.

```
yas = 18
mezuniyet = 'lise'
sonuc = (yas>=18) and (mezuniyet=='lise')
```

(yas>=18) bize True bilgisi ve (mezuniyet=='lise') bize True bilgisi getirir. Dolayısıyla (True and True) bize True bilgisi getirir ve kişi ehliyet alabilir diyebiliriz.

Python Or (veya) Operatörü

Python or operatörü ile belirtilen koşulların sadece birinin doğru olması sonucun True olması için yeterlidir. Hepsi yanlış ise sonuç da False olur.

Örnek

Ehliyet almak için mezuniyet bilgisinin lise, üniversite ya da yüksek lisans olması durumlarından sadece birinin doğru olması yeterlidir.

```
mezuniyet = 'lise'
sonuc = (mezuniyet=='lise') or (mezuniyet=='üniversite') or (mezuniyet=='yüksek lisans ')
```

Koşullarımıza baktığımızda sadece ilk koşulun doğru olduğunu görüyoruz ancak sonuc değişkenine yine de True bilgisi gelir çünkü or operatörü ile birleştirilen koşullardan sadece birinin doğru olması yeterlidir.

(True or False or False) => True

Hatta yaş sorgusunu da örneğe dahil edersek aynı anda and ve or operatörlerini kullanmış oluruz.

Örnek

```
yas = 18
mezuniyet = 'lise'
sonuc = (yas>=18) and ((mezuniyet=='lise') or (mezuniyet=='üniversite') or (mezuniyet=='yüksek
lisans '))
```

yaş 18 ve üstünde olduğunda bize True gelir. Mezuniyet ise lise, üniversite ya da yüksek lisans değerlerinden birine eşit olduğu durumda True gelir. Sonuç olarak iki koşuldan da True and True bilgisi geldiği için sonuç da True olur ve kişi ehliyet alabilir diyebiliriz.

** or operatöründen sonra ekstra bir parantez daha kullanıldığına dikkat ediniz. Sonuçta and operatörünün solunda ve sağında birer koşul var ve koşullarda kendi içlerinde guruplanabilir.

Python Not (değil) Operatörü

Python not operatörü ile koşulların tersi alınır. Örneğin bir koşul False üretiyorsa not operatörü ile True bilgisine çevrilir.

Örnek

```
x = 5
sonuc = not(x > 3 and x < 10)
```

Burada x, 3' den büyük ve 10' dan küçük bir sayı ve and operatörü ile birleştirildiğinden dolayı bize True bilgisi gelir ancak not(True) dediğimizden dolayı sonuc değişkenine False bilgisi atanır.

Pythonda String Tanımlama

Pythonda Karakter Dizileri yani string veri tipleri **tek tırnak (' ')** ya da **çift tırnak (" ")** ile oluşturulur.

'Hello World' ile "Hello World" tanımlaması aynıdır. Ancak bazen karakter dizileri içerisinde tek tırnak karakterini karakter dizisinin bir elemanı gibi göstermek isteriz.

Örnek

"I'm from Turkey" şeklinde tek tırnak karakterini tanımlayabilmek için mecburen çift tırnak kullanmamız gerekir çünkü **'I'm from Turkey'** bu şekilde bir kullanım hata verecektir.

Birden Fazla Satırda String Tanımlama

3 tırnak ile birden fazla satırda string tanımlaması yapabilirsiniz. 3 tane tek tırnak ya da 3 tane çift tırnak kullanabilirsiniz.

Örnek

```
text = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua."""
print(text)
```

String Birleştirme (String Concatenation)

Tanımladığımız string ifadeleri '+' operatörü ile birleştirebiliriz. Ancak string birleştirmede number değişkenleri str() ile string'e çevirmemiz gerekiyor.

Örnek

```
name = 'Sadık'
surname = 'Turan'
age = 36

greeting = 'My name is '+ name + ' '+ surname + ' and \nI am '+ str(age) + ' years old.'
print(greeting)  # My name is Sadık turan and I am 36 years old.
```

Bu şekilde değişken içeriklerine göre bir string ifadeyi + operatörü ile oluşturmuş olduk ve age değişkeni number türünde olduğundan dolayı str(age) şeklinde string' e çevirdik.

String Formatlama

'+' operatörünü kullanarak string birleştirme işlemi bazen zor olabiliyor. Dolayısıyla kullanabileceğimiz **format()** metodu ile **f-string** isminde iki farklı alternatifimiz mevcut.

String format() Metodu

```
name = 'Çınar'
surname = 'Turan'
age = 36
```

.format() metoduna vereceğimiz her parametre sırasıyla { }' lerin yerine kopyalanır.

```
print('My name is {} {}'.format(name, surname))
# My name is Çınar Turan
```

Gördüğünüz gibi birinci { }' in yerine Çınar ikinci { }' in yerine ise Turan yazdırılır.

```
print('My name is {1} {0}'.format(name, surname))  # My name is Turan Çınar
```

{ }' ler için indeks numarası da verebiliriz. Yani format metodu içindeki name, 0 ve surname, 1 değerlerini alır.

```
print('My name is {s} {n}'.format(n=name, s=surname))  # My name is Turan Çınar
```

format() metodu içindeki değişkenlere takma isim de verebiliriz.

```
print("My name is {} {} and I'm {} years old.".format(name, surname, age))
```

age 3. süslü parantezin yerine gelir ayrıca age number olduğundan dolayı format() metodunda str() fonksiyonunu kullanmamız gerekmez.

```
print("My name is {} {} and I'm {} years old.".format(name, name, name))
# My name is Çınar Çınar Çınar
```

forma içine eklediğimiz 3 tane name değişkeni sırasıya süslü parantezler yerine gelir. Eğer ki tek name değerini 3 kere yazdırmak istersek, {0} şeklinde kullanabiliriz.

```
print("My name is {0} {0} and I'm {0} years old.".format(name))
# My name is Çınar Çınar Çınar
```

f-String ile String Formatlama

f-string ile kolaylıkla string birleştirme işlemi yapabiliriz. Çünkü string ifadenin içinde süslü parantezler içinde değişkenleri yazabiliriz.

```
print(f"My name is {name} {surname} and I'm {age} years old.")
```