Python Nedir?

1) **Özgür ve ücretsiz** bir programlama dilidir.
2) **Guido Van Rossum** adlı Hollandalı bir programcı tarafından 90'lı yılların başında geliştirilmeye başlanmıştır. Guido Van Rossum 2005 ile 2012 yılları arasında Google'da çalışmıştır.
3) Adı **"The Monty Python"** adlı bir İngiliz komedi grubundan esinlenerek konmuştur.
4) Python kelimesi Türkçe **"paytın"** şeklinde telaffuz edilir.

Neden Python?

1) Büyük kuruluşlar **(Google, YouTube ve Yahoo! gibi)** her zaman Python programcılarına ihtiyaç duyuyor.
2) Python ile **masaüstü, oyun, mobil, web ve ağ** alanlarında programlar yazabilirsiniz. Örneğin bir **youtube video indirme** programı veya **fizyde şarkı arayıp dinleme** programı yapabilirsiniz.
3) Python kodları sade, basit ve hızlıdır. Derlenmeye ihtiyaç duymaz.
4) Python farklı işletim sistemleri üzerinde çalışabilir. **Linux, Windows, Mac OS X, MS-DOS, iOS ve Android** vbs…
5)

Python Sürümleri:

1) Piyasada iki çeşit Python sürümü vardır. **Python2 ve Python3**
2) Python3, Python2'ye göre daha güçlüdür ve hatalardan arınmıştır.
3) Python2 ile yazılmış bir program Python3'te çalışmaz. Aynı şekilde Python3 ile yazdığınız bir program Python2'de çalışmaz.

Python'u Kurmak:

1) Python'ı kurmak için **http://www.python.org** adresine tıklayınız.
2) Açılan sayfada **Downloads** linkini tıkladığınızda **'Download the latest versiyon for Windows**' başlığı altında Python'un son sürümü içeren bir buton göreceksiniz. Bu butona tıklayarak Python'ın son sürümünü bilgisayarınıza indirebilirsiniz.
3) Python'ı farklı bir işletim sistemine yüklüyorsanız ya da eski bir Python sürümü kurmak istiyorsanız ilgili sayfada gerekli linkler ve bilgilendirmeler mevcuttur.
4) Bu arada Python'ın hangi dizine kurulduğunu bilmemiz önemlidir. Zira karşılaşacağımız bazı sorunlar bu dizine gitmemizi gerektirebilir.

Python Dosyaları Nerede?

Windows kullanıcısı iseniz ve eğer siz farklı bir yere kaydetmemişseniz Python dosyaları genellikle **C:\Users\Kullanicilar\AppData\Local\Programs\Python** dizini içindedir. Klasörler gözükmüyorsa gizli dosyaları göster seçeneğini aktifleştiriniz.

Editörler: Kod yazmak, saklamak, düzenlemek ve geliştirmek için kullandığımız programlardır.

IDE Nedir?

IDE (Integrated Development Environment), Türkçesiyle Tümleşik Geliştirme Ortamı, bilgisayar yazılımcılarının daha kolay şekilde yazılım geliştirebilmesi için tasarlanan ve yazılım geliştirme aşamasında geliştiriciye birçok kullanışlı araç sunarak daha kolay ve etkili şekilde yazılım geliştirmesine yardımcı olan yazılımlardır.

Python için hangi editörleri kullanabiliriz?

Python için birçok editörü kullanabiliriz. Bu tamamen kullanım alışkanlıkları ve kullanım kolaylığı gibi faktörlere bağlıdır. Bu editörlerden bazıları şunlardır.

Python IDLE: Python.org web sitesinde yer alan ücretsiz bir Python editörüdür. Başlangıç seviyesi için en temel editördür. Küçük kodları IDLE ekranına yazabilirsiniz ancak daha geniş kapsamlı kodlar için sol üst köşede **File [Dosya]** menüsüne ait olan **New Window [Yeni Pencere]** seçeneğine tıklamanız gerekir. Burada beyaz bir ekranla karşılaşacaksınız. İşte asıl kodları buraya yazmanız gerekecektir. Yazdığınız kodları kaydetmek içinse **File [Dosya]** menüsüne ait olan **Save As [Farklı Kaydet]** seçeneğine tıklamanız gerekir. Kodları direk çalıştırmak için ise **Run [Çalıştır]** menüsüne ait olan Run Module seçeneğine ya da klavyeden direk F5 tuşuna tıklayabilirsiniz.

Not: Python'a yeni başlayanların en sık yaptığı hatalardan biri IDLE ekranındaki **>>>** işareti ile komut arasında boşluk bırakmalarıdır. Bu durumda kodlar çalışmayacaktır. Bu yüzden kod yazarken en başta boşluk bırakmamaya özen gösteriniz.

Wing IDE: https://wingware.com/downloads/wingide-101 adresinden indirilebilirsiniz. Wing IDE Professional, Wing IDE Personal ve Wing IDE 101 olmak üzere üç çeşidi vardır. Bunlardan **Wing IDE 101** ücretsizdir.

Canopy: https://store.enthought.com/downloads/#default adresinden ücretsiz indirilebilirsiniz.

Pycharm: Python için en çok tercih edilen editördür. https://www.jetbrains.com/pycharm/download/#section=windows adresinden indirebilirsiniz.

Diğer Editörler: Sublime Text, Eclipse , Visual Studio, Pydev, VIM, Spyder, Komodo, PTVS, Eric Python, Anaconda, Geany, Emacs, Atom, PyScripter, Jupyter Notebook vbs…

PROGLAMLAMA KAVRAMLARI:

Derleyiciler(Compiler) : Kaynak kodları hedef kodlara dönüştürür.

Yorumlayıcılar (interpreter) : Kaynak kodları hedef kodlara dönüştürmekle beraber programınçalışmasını tekrarlar. Python, yorumlanan bir dildir.

Yanaylaçlar(Profiler): Programın çalışmasıyla ilgili istatistiki veri toplar.

Hata Ayıklayıcılar (Debugging): Programdaki hataları bulur.

Sabitler, Değişkenler ve Atama: Değeri değişen ifadelere değişken, değeri değişmeyen ifadelere de ise sabit denir.
. Böylece daha sade, işlevsel ve zaman kazandırıcı işlemler yapılır.

```
#a değişken, "Batman" ise atanan değerdir. = ise atama işlemi yapar.

>>> a = "Batman"

>>> print(a)

Çıktı: Batman
```

Ör:

```
a= 5
b=-4
c=30
print(a*b+c)
Çıktı:10
```

Not: Burada = sembolünün anlamı matematikte kullanıldığı şeklinden farklıdır. Matematikte bu sembol eşitlik sağlar fakat Python 'da simetri olmadığı için 5 = x gibi bir ifade hatalı olacaktır.

```
5=x
print(x)
Çıktı:hata
```

DEĞİŞKEN KURALLARI:

1-Değişken isimleri sayı ile başlayamaz.

3_kilo_elma = “10 tl” ✗
kilo_elma_3 = "10 tl" ✔

2-Değişken adları özel sembol içermez (_ altçizgi hariç)

gelir?= “500 TL” ✗
kullanici_adi= “admin” ✔

3- Değişkenler isimlendirilirken kelimeler arasında boşluk bırakılılamaz.

kullanici adi = “admin” ✗
kulllanici_adi = “admin” ✔

Not: Değişken adlarında Türkçe karakter kullanabiliriz. Ancak uyum sorunu ihtimaline karşı bundan kaçınınız.

4- Değişken adlarında bazı özel anlam ifade eden kelimeler kullanılmaz.

True=5 ✗
true=5 ✔ →Küçük büyük harf duyarlılığından hata oluşmaz.

and=8 ✗
And=8 ✔ →Küçük büyük harf duyarlılığından hata oluşmaz.

Not: Python’da özel anlam ifade eden kelimeleri görmek için aşağıdaki kodları yazın.

```
import keyword
keyword.kwlist
```

Çıktı:

['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

Ör:

```
import keyword
a=keyword.kwlist
len(a)                    #Python’da özel anlam ifade eden kaç adet kelime var?
```

Çıktı: 33

Ör: Birden fazla değişkene aynı değer tek sırada atanabilir.

```
>>> a=b=c=4
>>>print(a*b+c)      #4*4+4

Çıktı: 20
```

Ör: Bir değişkene defalarca farklı değerler atayabiliriz. Son değer geçerlidir.

```
>>> x = 10
>>> x = 20
>>> x = 30
>>> print(x+5)        #30+5

Çıktı: 35
```

Ör: Aynı satırda farklı değişkenlere farklı değerler atanabilir.

```
>>> x, y, z = 4,3,2
>>> print(x*y-z)      #4*3-2

Çıktı:10
```

Ör: Değişken Değerleri birbirleriyle takas edilebilir.

```
>>> x,y,z=4,3,2
>>> x,y,z=y,z,x          #x artık y, y artık z, z artık x oldu.
>>> print(z**y*x)        #4**2*3

Çıktı: 48
```

Ör: Değişken iptali için komut penceresini kapatıp açabiliriz ya da **del** komutunu kullanabiliriz.

```
>>>a=2
>>>print(a)
2
>>>del a          #a değişkeni artık yok yani iptal edildi.
>>>print(a)       #a değişkenini iptal ettiğimiz için kod artık hata verecektir.
```

Not: Python operatör kullanırken kısaltmalar yapabilir. Örneğin x= x + 5 deyimi x + = 5 olarak kısaltılabilir. Bu ifade " x'i 5 arttır." anlamına gelir.

```
>>> x=5
>>> x+=7          #x=x+7 ile aynı anlama gelir.
>>> print(x)
12
>>> x -= 4        #x=x-4 ile aynı anlama gelir.
>>> print(x)
8
>>> x //= 3       #x=x//3 ile aynı anlama gelir.
>>> print(x)
2
>>> x *= 8        #x=x*8 ile aynı anlama gelir.
>>> print(x)
16
>>> x %= 6        #x=x%6 ile aynı anlama gelir.
>>> print(x)
4
>>> x**= 3        #x=x**3 ile aynı anlama gelir.
>>> print(x)
64
```

Yorumlar: Python ’da kodlar zamanla karmaşık hâle gelir ve bu da okumayı zorlaştırır. Bunun için kodlara açıklama eklenmesi gerekebilir. # işareti ile program içerisine yorum yazmak mümkündür. Programlama dili, yorumları göz ardı eder ve kod yapısı bozulmaz. Ör;

'Fırat' , 'Dicle' **#virgül iki ayrı string’i birleştirir.**

Not: Üç tırnak arasına çoklu yorum yazabiliriz.

```
>>>a=5
>>>b=9
“””
a ile b’yi toplayalım
bakalım ne çıkacak

“””
>>>a+b      # a ile b’yı topla
```

14

VERİ TÜRLERİ:

string*:* Tırnak içindeki her türlü karakter dizileridir.

>>>”Merhaba Dünya”
‘Merhaba Dünya’

Not: string’lerde çift tırnak kullanmak şart değildir. Tek tırnak veya üç tırnak da kullanabiliriz.

>>>‘Merhaba Dünya’ **→tek tırnak**
‘Merhaba Dünya’

>>>’’’Merhaba Dünya’’’ **→üç tek tırnak**
‘Merhaba Dünya’

>>>”””Merhaba Dünya””” **→üç çift tırnak**
‘Merhaba Dünya’

Not: Python’da üç tırnak kullanmamızın sebebi alt satıra geçebilmektir.

'''Bilgisayar
Bilimi
Dersi'''

Çıktı:
Bilgisayar
Bilimi
Dersi

#Peki, neden bazen tek tırnak veya çift tırnak kullanmamız gerekiyor? Örnek olarak şöyle bir çıktı elde etmek isteyelim. **İstanbul'un ilçeleri**
Bu ifadede bir kesme işareti var. İçinde kesme işareti olan bir string’i tek tırnak içine alamayız. Bu yüzden diğer bir alternatif olan çift tırnağı ya da üç tırnağı kullanmalıyız.

"İstanbul'un illeri" **İstanbul'un illeri**
"””İstanbul'un illeri”"" **İstanbul'un illeri**

integer: Tam sayılardır. Pozitif, negatif ya da sıfır değeri alabilir. Kesirli değer içermez. **Ör:** 10, 0, -10

float: Reel sayılardır. Kayan noktalı sayılar da denir. Ancak burada virgül yerine nokta kullanmamız gerekir. **Ör:** 12.6, 12.0, -12.0, 0.0

bool: Herhangi bir ifadenin doğruluğunu veya yanlışlığını sorgular. Bool iki değer alabilir. Bunlar True ve False 'tur.

Dönüştürme Fonksiyonları

str(): Verileri string'e (karakter dizisi) çevirir.

```
>>> str(4)       Çıktı: '4'
>>> str(4.0)     Çıktı: '4.0'
>>> str(10+2)    Çıktı: '12'
>>> str(10/2)    Çıktı: '5'
```

int(): Verileri integer'a (tamsayı) çevirir.

```
>>> int(28.9) Çıktı: 28
>>> int(5+4)  Çıktı: 9
>>> int(5/3)  Çıktı: 1
>>> int(5*0.4) Çıktı: 2
>>> int(0.8+0.9)        Çıktı: 1
>>> int("5")  Çıktı:  5
```

Not: Tırnak içindeki verilerin tamsayıya çevrilmesi için tırnak içindeki ifadenin tamsayı olması gerekmektedir. Aksi halde çevirme işlemi yapılamaz.

```
>>> int("28.9") #tırnak içerisindeki ifade tamsayı değildir. Hata verir.
>>> int("5+4")  #hata, tırnak içerisindeki ifade tamsayı değildir. Hata verir.
>>> int("bilgisayar")  #hata, tırnak içerisindeki ifade tamsayı değildir. Hata
verir.
```

float(): Verileri float'a (reel sayı) çevirir.

```
>>> float(5) Çıktı: 5.0
>>> float("5") Çıktı: 5.0
>>> float("5.7")        Çıktı: 5.7
>>> float(-5)   Çıktı: -5.0
>>> float(0) Çıktı: 0.0
>>> float(3+4) Çıktı: 7.0
>>> float(3.5+4)  Çıktı: 7.5
```

Not: Tırnak içindeki verilerin float'a çevrilmesi için tırnak içindeki ifadenin tamsayı veya float olması gerekmektedir. Aksi halde çevirme işlemi yapılamaz.

float("3+4") **#tırnak içerisindeki ifade tamsayı veya float değildir. Hata verir.**

print() fonksiyonu: Ekrana yazdırma fonksiyonudur.

Kural: Stringleri tırnak içerisine almamız gerekir.

```
>>> print("Merhaba Dünya")    Çıktı: 'Merhaba Dünya'
```

Kural: Stringleri tek tırnak içerisine de alabiliriz.

```
>>> print('Merhaba Dünya')    Çıktı: 'Merhaba Dünya'
```

Kural: Stringleri üç tek tırnak içerisine de alabiliriz.

```
>>> print(''''Merhaba Dünya''')    Çıktı: 'Merhaba Dünya'
```

Kural: Stringleri üç çift tırnak içerisine de alabiliriz.

```
>>> print(,''Merhaba Dünya''')    Çıktı: 'Merhaba Dünya'
```

Kural: Stringleri tırnak içerisine almazsak kod hata verir.

```
>>> print(Merhaba Dünya)      Çıktı: Hata
```

Kural: Stringlerde tırnaklar eksik olursa kod hata verir.

```
>>> print(,Merhaba Dünya)     Çıktı: Hata
```

Kural: Sayıları tırnak içerisine almamıza gerek yoktur.

```
>>> print(5)                  Çıktı: 5
```

Kural: Sayıları tırnak içerisine alırsak o artık bir string'tir. Dolayısıyla matematiksel olarak bir anlam ifade etmez.

```
>>> print('5')                 Çıktı: '5'
```

Kural: Parantezin içinde matematiksel işlem varsa sonuç yazılır.

```
>>> print(6+3)                Çıktı: 9
```

Kural: Matematiksel işlem tırnak içinde yazıldıysa veri artık string olacağı için işlem yapılmaz ve aynen yazılır.

```
>>> print(“5+4”)              Çıktı: ‘5+4’
```

Kural: Boş bir string ekrana yazdırılabilir.

```
>>> print(“”)                 Çıktı:
```

Kural: Virgüller ifadelerin arasına birer boşluk bırakır.

```
>>>print(“Bilgisayar”,“Bilimi”,“Kodlama”)
```

```
Çıktı: Bilgisayar Bilimi Kodlama
```

Kural: + işareti ifadeleri birleştirir.

```
>>> print(“bilgi” + “sayar”)  Çıktı: bilgisayar
```

```
>>> print( “bilgi"+"  "+"sayar”)
```

```
Çıktı: bilgi sayar
```

Açıklaması: Ortadaki çift tırnağın içinde bir boşluk karakteri var. Dolayısıyla araya bir boşluk karakteri ekledik.

Kural: + işareti koymadan da ifadeler birleştirilebilir.

```
>>> print(“bilgi” “sayar”)  Çıktı: bilgisayar
```

```
Ör: >>> print(999 + “9”) #integer ve string toplanmaz
Çıktı: Hata
```

len() fonksiyonu: Stringlerin uzunluğunu ölçer.

```
>>> len(“Türkiye”)                    Çıktı: 7
```

```
>>> len("Bilgisayar Bilimi")          Çıktı: 17
```

“Bilgisayar Bilimi” stringinin uzunluğu 17 karakterdir. Burada boşluk karakterinin de sayıldığına dikkat edelim.

```
>>> len("Haydi, Kodla!")              Çıktı: 13
```

Noktalama işaretleri de sayılır.

```
>>> len("Bilgisayar Bilimi") + len("Dersi") Çıktı: 22
```

“Bilgisayar Bilimi” stringi ile *“Dersi”* stringinin uzunluğu sayısal olarak toplandı.

```
>>> len("Bilgisayar Bilimi") - len("Dersi") Çıktı: 12
```

Örnekler:

```
>>> len("len")  Çıktı: 3
>>> len(“0”)    Çıktı: 1
>>> len(“-4”)   Çıktı: 2
>>> len("4.5")  Çıktı: 3
>>> len(“-1.5”) Çıktı: 4
>>> len("12+42”)Çıktı: 5
>>> len(“100-20”)Çıktı: 6
>>> len(str(10))Çıktı: 2
>>> len(int(“10”)Çıktı: hata (string değil)
>>> len(563)    Çıktı: hata (string değil)
```

type() fonksiyonu: Verilerin türünü sorgular.

```
>>> type("Merhaba")    Çıktı: <class 'str'>

>>> type(“”)       Çıktı: <class 'str'>

>>> type("4")      Çıktı: <class "str">

>>> type(4)        Çıktı: <class "int">

>>> type(4.2)      Çıktı: <class 'float'>

>>> type(4.0)      Çıktı: <class 'float'>

>>> type(Merhaba) Çıktı: tırnak işareti olmadığı için hata verir.
```

\n parametresi: Bu parametreye newline adı verilir. print() fonksiyonu içerisinde kullanıldığında ilgili yerden bir alt satıra geçiş yapar.

```
>>> print("bilgisayar bilimi")
Çıktı: bilgisayar bilimi
```

\n karakterini araya koyduğumuzda ilgili yerden bir alt satıra geçiş yapacaktır. Yukarıdaki örneği bir de şu şekilde yazalım.

Ör:

```
>>> print("bilgisayar\nbilimi")
Çıktı:
bilgisayar
bilimi
```

#Görüldüğü üzere **\n** parametresini bilgisayar ifadesinin hemen sonuna koyduk ve böylece tam da oradan satır başına geçiş yapmış olduk.

\t parametresi: print() fonksiyonu içerisinde kullanıldığında ilgili yerden bir tab kadar boşluk bırakır.

Ör:

```
>>> print("Ocak\tŞubat\tMart")
Çıktı: Ocak     Şubat   Mart
```

*** parametresi:** stringi parçalara böler.

Ör: `>>> print(*"Linux")`
Çıktı: `L i n u x`

sep parametresi: İngilizcede separator (ayırıcı, ayraç) kelimesinin kısaltmasıdır.

>>> print("www.", "google.", "com")
Çıktı: www. google. com

sep=" " ifadesi görünmezdir, yani aslında o arka planda çalışır ve default olarak tırnak içindeki ifadelerin arasında boşluk bırakır. Ancak tırnak içindeki ifadelerin arasına boşluk değil de başka bir karakter koymak istersek o zaman iş değişir. O halde yapmamız gereken sep parametresinin içine istediğimiz karakteri koymaktır. O halde sep parametresine (yani sep= " " deki çift tırnak arasına) hiçbir şey yazmaz isek tırnak içindeki ifadelerin arasında hiç boşluk oluşmaz. O halde doğru kodumuzu yazalım.

>>> print("www.", "google.", "com", sep="") #Bir boşluk nelere kadirÇıktı: www.google.com

Not: sep=" " ile sep=None aynı anlama gelir.

end parametresi: print() içerisinde kullanılır. Yazdırılmak istenen ifadelerin sonuna hangi karakterin geleceğini belirler. Varsayılan olarak "\n" parametresi ile birlikte gelir. Yani yazılan ifade bitince bir alt satıra geçer.

Ör:
>>> print("Bugün günlerden Salı")
Çıktı: Bugün günlerden Salı

#Burada herhangi bir end parametresi göremiyoruz. Ancak Python yukarıdaki kodu aslında şu şekilde algılar:

>>> print("Bugün günlerden Salı", end="\n")

#end parametresinin değerini değiştirelim.

>>> print("Bugün günlerden Salı",end=".")
Çıktı: Bugün günlerden Salı.

#Böylece end fonksiyonu ile string'in sonuna bir nokta koymuş olduk.

Ör:
print(*"Linux", sep=".")
Çıktı: L.i.n.u.x

Ör:

```
print(*"Linux", sep="\n")
```

Çıktı:

```
L
i
n
u
x
```

Açıklama: * işaretini kullanarak “Linux” stringini parçalara böldük ve bu parçaların arasında sep parametresini kullanarak satır başı (\n) yaptık.

string indeksleme ve parçalama: stringlerde her bir karakterin kendine has bir konumu vardır. Bu konumlara indeks adı verilir. Python'da ve çoğu programlama dilinde indeksleme "0" dan başlar. Aşağıdaki örnekleri inceleyelim.

```
>>> a="Kodlama"
>>> a[0]
'K'
>>> a[2]
'd'
>>> a[-1]           #sondan 1.karakter
'a'
>>> a[-3]
'a'
>>> a[2:6]
'dlam'
>>> a[:6]          #baştan 6.karaktere kadar tüm karakterler
'Kodlam'
>>> a[2:]          #2.karakterden sona kadar tüm karakterler
'dlama'
>>> a[:]           #tüm karakterler
'Kodlama'
>>> a[2:-2]
'dla'
>>> a[:-2]
'Kodla'
>>> a="Kodlama Eğitimi"
>>> a[2:13:2]    #2.karakterden 13.karaktere kadar 2 sıra atlayarak
'daaEii'
>>> a[::2]       #Baştan sona kadar 2 sıra atlayarak
'KdaaEiii'
>>> a[::-1]      #Sondan başa kadar 1 sıra atlayarak
'imitiğE amaldoK'
```

Ör: İsminizi tersten yazdırın.

```
isim = input ("İsminiz :")

print("İsminizin tersten yazılışı :", isim[::-1])
```

Formatlama: Bir stringin içine önceden tanımlanan bir veri veya değişken yerleştirebiliriz. Bu işleme formatlama denir. Bu işlem için süslü parantez kullanmalıyız.

Ör:

```
"{}{}{}".format(5,6,7)
```

'567'

→**Açıklaması:** Tırnak işaretlerinin arasına üç adet açılıp kapanan süslü parantez yerleştirdik. Sonra .format() fonksiyonunun içine sırasıyla 5,6,7 yazdık. Böylece ilk süslü parantezin içine 5, ikinci süslü parantezin içine 6, üçüncü süslü parantezin içine de 7 koymuş ve bunları yazdırmış olduk.

Ör: Şimdi de değişken kullanarak formatlama işlemi yapalım.

```
a=3
b=4
"{}+{}'ün toplamı {}'dir.".format(a,b,a+b)
```

Çıktı: '3+4'ün toplamı 7'dir.'

Ör:

```
a = "{} ve {} çok iyi arkadaştır."
a.format("Ali", "Veli")
```

Çıktı: 'Ali ve Veli çok iyi arkadaştır.'

Ör: Süslü parantezlerin içine sayı koyarak yazdırma sıralamasını belirleyebiliriz.

```
"{2} {0} {1}".format("Ali","Veli","Murat")
```

Çıktı: 'Murat Ali Veli'

Açıklaması: Tırnak işaretlerinin arasına üç adet açılıp kapanan süslü parantez yerleştirdik. Bu süslü parantezlerin içine de verilerin hangi sırada olacağını gösteren sayılar koyduk. Yani burada {2} 'nin anlamı ilk sıraya 2.verinin yani "Murat" ın geleceğidir. Bu arada Python'da veri sıralaması 1'den değil 0'dan başlamaktadır. O yüzden "Ali" 1.değil 0.sıradadır. Dolayısıyla "Veli" de 1.sıradadır. Böylece ilk süslü parantezin içine "Murat", ikinci süslü parantezin içine "Ali", üçüncü süslü parantezin içine de "Veli" koymuş ve bunları yazdırmış olduk.

input() fonksiyonu : Kullanıcıdan bilgi almak için kullanılır.

Ör: Kullanıcıya ismini sorup "Merhaba İsim" yazdıran kodlar.

```
isim = input("İsminiz nedir? ")
print("Merhaba", isim)
```

Çıktı:

İsminiz nedir? Niyazi
Merhaba Niyazi

Ör: Kullanıcın girdiği iki sayıyı toplayan kodlar: (Öğrenciler daha sonra üç veya dört adet sayıyı toplasın)

```
deger1 = int(input("İlk sayıyı giriniz: "))
deger2 = int(input("İkinci sayıyı giriniz: "))
toplam = deger1 + deger2
print(deger1, "+", deger2, "=", toplam)
```

Çıktı:

İlk sayıyı giriniz: 5
İkinci sayıyı giriniz: 52
5 + 52 = 57

Ör: Girilen yaşa yorum yazan kodlar:

```
yas = input("Yaşınız: ")
print("Demek", yas, "yaşındasın.")
print("Daha çok gençsin")
```

Ör: Saniyeyi dakikaya çeviren kodlar:

```
saniye=int(input("Saniye sayısını giriniz: "))
dakika=saniye/60
print(saniye,"saniye ",dakika,"dakika eder")
```

Ör: Dairenin alanını bulan kodlar:

```
r = int(input("Yarıçapı giriniz:"))
pi = 3.14
alan = pi * (r **2)
print("Dairenin alanı:", alan)
```

Ör: Kullanıcının girdiği saniyeleri, saat, dakika ve saniye olarak parçalara ayıran program:

```
saniye = int (input ("saniye sayısını girin:"))
saat = saniye // 3600      #3600 saniye = 1 saat
saniye = saniye% 3600
dakika = saniye // 60      #60 saniye = 1 dakika
saniye = saniye% 60
print(saat, "sa", dakika, "dk", saniye, "sn")
```

OPERATÖRLER (İŞLEÇLER):

Aritmetik Operatörler:

Toplama : +
Çıkarma : -
Çarpma : *
Bölme : /
Üs Alma : **

+ operatörü: Toplama ve birleştirme için kullanılır.

```
>>>10+20        30
>>>5.5+3.8      9.3
>>>9+4.0        13.0
```

Not: Sayıların çift tırnak içine alınmadığına dikkat edin. Eğer çift tırnak içine alınsaydı veri türü bir integer değil string olacaktı. Bu durumda matematiksel işlem değil birleştirme işlemi yapılacaktır.

Örnekler:

```
>>>"10"+”20”       ‘1020’
>>>"10" + 20       →hata (ikisinin de aynı tür veri olması gerekir)
>>>"5" + str(10)   ‘510’
>>>5 + int("10")   15
```

- operatörü:

```
>>>50-30          20
>>>-7- - 9        2
>>>1.5-0.5        1.0
>>>4.0-1          3.0
```

*** operatörü: Çarpma ve stringleri belli sayıda tekrar etmek için kullanılır.**

```
>>>10*5           50
>>>-6*-8          -48
>>>1.5*1.5        2.25
```

Ör:

```
>>>x=4
>>>y=3
>>>3*x+2*y-5
13
```

Ör:

>>>"w" * 3 →"w" stringinin 3 defa tekrar ederek yazdırdı.
www

Ör:

>>>"aheste " * 2 →"aheste " tırnak kapanmadan boşluk bırakıldığına dikkat edin.
aheste aheste

Ör:

>>>"-" * 10 →"-" işaretini 10 kere çoğalttık.

/ operatörü: Bölme işlemi gerçekleştirir. Sonuç daima float veri türüdür.

>>>21/3 **7.0**
>>>int(21/3) **7** → burada veri integer'a dönüştürüldü.
>>>21/0 →hata

// operatörü: Bölme işleminde kalan sayı göz ardı edilir.(Taban Bölme)

>>>25//6 **4**
>>>6//25 **0**
>>>4.5//1.2 **3.0**
>>>2.1//1 **2.0**

Not: - eksi ve / bölü işleçlerini karakter dizileri ile birlikte kullanamayız.

Üs Alma Birinci Yol:

5**2 **25**
5**-1 **0.2**
5**0 **1**

Üs Alma İkinci Yol:

pow(5,2) **25**
pow(5,-1) **0.2**
pow(5,0) **1**
pow(11,3,4) **3** →11'in 3. kuvveti olan 1331'in 4'e bölümünden kalan sayı

Not: **$5*10^3$ gibi bir ifade şöyle yazılabilir:**

5e2 **5000**
5E2 **5000**

Mod İşlemi: **Bölme sonucunda kalan sayıyı verir.**

25%7 **4**
22%11 **0**
6%25 **6**
0%25 **0**
25%0 **→hata**

Karekök Alma: **Bir sayının 0.5. kuvveti o sayının karekökü̈dür.**

>>>144**0.5
12

Yuvarlama:

round(28.71) **29**
round(28.47) **28**

Not: **Yuvarlama işleminde eğer 5 değeri ile karşılaşırsak en yakın çift sayıya yuvarlama yapmamız gerekir.**

round(27.5)
28 **→en yakın çift sayı olan 28'e yuvarlıyor.**

round(28.5)
28

→Açıklaması: En yakın çift sayı kuralından dolayı sonuç 28 oluyor.

Ör:

x = 793.748
round(x) **794**
round(x, 1) **793.7**
round(x, 2) **793.75**
round(x, 0) **794.0**
round(x, -1) **790.0**
round(x, -2) **800.**

_ (alt çizgi işareti): Son verinin değerini hafızada tutar.

Ör:

```
>>>10+5
15
>>>_+5                    →altçizgi ile 5'i topladık.
20                        →artık son öğe 15 değil 20 oldu.
>>>_/10
2.0
```

ARİTMETİK ALIŞTIRMALAR:

```
5+4*3/3-9
Çıktı: 0.0

15-2**4/2+(-2-2)
Çıktı: 3.0

0-9**0
Çıktı: -1

100**0.5/10*2
Çıktı: 2.0

10%6-1
Çıktı: 3

39//12+12/2
Çıktı: 9.0

“10+20+30”
Çıktı: 10+20+30

“55”+”55”
Çıktı: 5555

5+”5”
Çıktı: hata

str(12+8)+"0"
Çıktı: 200

int("56")+int("12")
Çıktı: 68

“k”*3+str(5)
Çıktı: kkk5

2*"6",66)
Çıktı: 66 66

5+4*10/(4+5-9)
hata
```

KARŞILAŞTIRMA OPERATÖRLERİ:

Eşittir	**==**
Eşit değildir	**!=**
>	**büyüktür**
<	**küçüktür**
>=	**büyük eşittir**
<=	**küçük eşittir**

Bool Kavramı: Bool herhangi bir ifadenin doğruluğunu veya yanlışlığını sorgular. Eğer bir sorgulamanın sonucu doğru ise True, yanlış ise False çıktısı alıyoruz.

a=1	
a==1	**True**
a==2	**False**
a!=5	**True**
a!=1	**False**
a>2	**False**
8>a	**True**
a>=1	**True**
a>=2	**False**
“Bilim”<”Kodlama”	**True** →**alfabetik sıraya göre hareket eder.**

Not: Bool işleçleri sadece doğruluk-yanlışlık sorgulamaya yarayan araçlar değildir. Bilgisayar biliminde her şeyin bir bool değeri vardır. 0 değeri ve boş veri tipleri False ‘tur. Bunlar dışında kalan her şey ise True ‘dur.

bool(5)	**True**
bool(5.8)	**True**
bool(-5)	**True**
bool(“Steve Jobs”)	**True**
bool(“0”)	**True**
bool(“ “)	**True**
bool()	**False**
bool(“”)	**False**
bool(0)	**False**
bool(0.0)	**False**

and, or, not operatörleri

```
>>>a = 23
>>>b = 10
>>>a == 23 and b == 10
True
>>>a == 23 and b == 56
False
>>>a == 23 or b == 56
True
```

not: Değil anlamı taşır. Kullanıcı tarafından bir değişkene veri girilip girilmediğini denetlemek için kullanılabilir.

Ör:

```
>>>a = 23
>>>not a
False
>>>a = ""

>>>not a
True
>>>a=0
>>>not a
True
```

Örnek Program: Parola girilip girilmediğini denetleyen kodlar.

```
parola = input("parola: ")
if not parola:
    print("Parola boş bırakılamaz!")
else:
    print("İşlem tamam")
```

Aitlik Operatörü: Aitlik işleçleri, bir karakter dizisi ya da sayının, herhangi bir veri içinde bulunup bulunmadığını sorgular. Python 'da bir tane aitlik işleci bulunur. Bu işleç de **in** işlecidir.

```
>>>a = "abcd"
>>>"b" in a          →"b" ifadesi a değişkeninin içinde mi?
True
>>>"f" in a         →"f" ifadesi a değişkeninin içinde mi?
False
```

Kimlik Operatörü: **Python ‘da her şeyin bir kimlik numarası vardır. id() fonksiyonu ile bu kimlik numarasını bulabiliriz.**

```
>>>a = 100
>>>id(a)
137990748
```

→a değişkeninin temsil ettiği 100 sayısının kimlik numarası

KOŞULLU DURUMLAR: if, else, elif

if : "eğer" anlamı taşır. Bir koşulun sağlanması durumunda kullanılır.

```
#Yaşa göre yorum yazan program:
yas = int(input("Yaşınızı giriniz: "))
if yas < 14:
  print("Mrb çocuk!")
if 13 < yas < 19:
  print("Mrb Liseli!")
if 18 < yas < 24:
  print("Mrb üniversiteli!")
if 23 < yas < 30:
  print("Evlenme yaşın gelmiş")
if 29 < yas < 55:
  print("Çoluk çocuk nasıl?")
if yas > 54:
  print("Hayatın son demleri!")
```

else: "değilse" anlamı taşır. Bir koşulun sağlanmaması durumunda kullanılır.

```
#Bir sayının tek mi çift mi olduğunu tespit eden program:

sayı = int(input("Bir sayı girin: "))
     if sayı % 2 == 0:
        print("Girdiğiniz sayı çift sayıdır.")
     else:
        print("Girdiğiniz sayı tek sayıdır.")
```

```
#Kullanıcı adı yazılmamışsa uyarı veren program:

kullanıcı = input("Kullanıcı adınız: ")
if kullanıcı:
     print("Teşekkürler!")
else:
     print("Kullanıcı adı alanı boş bırakılamaz!")
```

```
#Kullanıcı adı veya parola yanlışsa uyarı veren program:

kullanıcı_adı = input("Kullanıcı adınız: ")
parola = input("Parolanız: ")
if kullanıcı_adı == "aliveli" and parola == "1234":
    print("Programa hoşgeldiniz")
else:
    print("Yanlış kullanıcı adı veya parola!")
```

elif: "eğer değilse" anlamı taşır. Eğer kodlarda sürekli if kullanırsak program tüm koşulları gözden geçirip ona göre karar verir. Ancak else sadece bir önceki if bloğuna göre hareket eder.

```
#boya göre yorum yazan program

boy = int(input("boyunuz kaç cm?"))
if boy < 160:
    print("boyunuz kısa")
elif boy < 180:
    print("boyunuz normal")
elif boy < 200:
    print("boyunuz uzun")
else:
    print("boyunuz çok uzun")
```

Not: "if" ile "elif" arasındaki farkı daha iyi anlamak için yukarıdaki programda elif yazan yerleri silip yerine if yazın. Bu durumda boyunuz kaç cm sorusuna 165 olarak cevap verseydik çıktı aşağıdaki gibi olacaktı. Çünkü "if" tüm koşulları değerlendirmiş olacaktı.

Çıktı:

boyunuz kısa
boyunuz normal
boyunuz uzun

```
#Parola uzunluğuna göre işlem yapan program:

parola=input("Parolanız: ")

toplam_uzunluk=len(parola)

mesaj="Parolanız toplam {} karakterden oluşuyor!"

if toplam_uzunluk > 12:

     print(mesaj.format(toplam_uzunluk)

     print("Parolanızın toplam uzunluğu 12 karakteri geçmemeli!")

else:

     print("Sisteme hoşgeldiniz!")
```

```
#İlk sayının ikinci sayıya tam bölünüp bölünmediğini hesaplayan ve
sonuca göre kullanıcıyı bilgilendiren program:

bölünen = int(input("Bir sayı girin: "))

bölen = int(input("Bir sayı daha girin: "))

şablon = "{} sayısı {} sayısına tam".format(bölünen, bölen)

if bölünen % bölen == 0:

    print(şablon, "bölünüyor!")

else:

    print(şablon, "bölünmüyor!")
```

for döngüsü: Başlangıç, bitiş ve artış değerleri belirtilen işlemleri tek tek tekrar eder.

Ör: 1'den 5' kadar olan sayıları yazdıralım.

```
print(1)
print(2)
print(3)
print(4)
print(5)
```

Çıktı:

1
2
3
4
5

Ancak, 1'den 100'e kadar yazmak gerekirse böyle bir çözüm yolu doğru olmayacaktır! Bu durumda döngü yapıları tercih edilmelidir. Python dilinde döngü için while ve for yapıları kullanılır.

Ör: 1'den 100'e kadar olan tüm sayıları yazdıralım. Değişkenimiz n olsun;

```
for n in range(1,100):
     print(n, end=" ")
```

Çıktı: 1 2 3..... 97 98 99

Ör: Şimdi de 1'den 100'e kadar tüm tek sayıları yazdıralım. Değişkenimiz yine n olsun;

```
for n in range(1,100,2):
     print(n, end=" ")
```

Çıktı: 1 3 5..... 95 97 99

Açıklama: range(1, 100, 2) ifadesindeki 1 başlangıç sayısıdır. Eğer burası boş bırakılırsa sayı otomatik olarak sıfırdan başlar. 100 ise yazılacak sayıların sınırıdır. 100 çıktıya dâhil değildir. 2 ise artış miktarını gösterir. Yani sayıyı 2'şer arttırır.

Örnekler:

range(10) → **0, 1,2,3,4,5,6,7,8,9** →**başlangıç ve artış değeri yok. Sadece bitiş değeri var.**

range(1, 10) → **1,2,3,4,5,6,7,8,9** → **artış değeri yok. Sadece başlangıç ve bitiş değeri var.**

range(1, 10, 2) → **1,3,5,7,9** → **başlangıç, bitiş ve artış değeri var.**

range(10, 0, -1) → **10,9,8,7,6,5,4,3,2,1** → **buradaki artış değeri eksiye doğru gitmektedir.**

range(10, 0, -2) → **10,8,6,4,2**

range(2, 11, 2) → **2,4,6,8,10**

range(-5, 5) → **−5,−4,−3,−2,−1,0,1,2,3,4**

range(1, 2) → **1**

range(1, -1, -1) → **1,0**

range(0) → **()**

Ör: 21'den 0'a kadar olan sayıları 3'er 3'er azaltarak yazdıralım.

```
for n in range(21,0,-3):
     print(n, end=" ")
```

Çıktı: 21 18 15 12 9 6 3

Ör: 1'den 100' kadar olan sayıların toplamı

```
top= 0
for i in range(1,100):
     top+= i
print(top)
```

Çıktı: 4950

Ör: 10 ve 10'un üstleri yazdıran program

```
for i in range(5):
    print("{}".format(10**i))
```

Çıktı:

```
1
10
100
1000
10000
```

Ör: Bir string değişkeninin her bir karakterini ayrı ayrı işleme ve yazdırma

```
a = "kodlama"
for harf in a:
     print(harf, end=" ")
```

Çıktı: k o d l a m a

Ör:

```
sayılar = "12345"
for sayı in sayılar:
     print(int(sayı) * 2, end=" ")
```

Çıktı: 2 4 6 8 10

Ör:

```
sayılar = "1234567"

for i in sayılar:
    if int(i) > 3:
        print(i, end=" ")
```

Çıktı: 4 5 6 7

Açıklaması: **sayılar** değişkeni oluşturduk. **"1234567"** ifadesindeki her bir karakteri ayırdık. Yani tüm sayılar artık bağımsızlığını ilan etmiş durumda. Ayrıca her bir karakteri **i** değişkene atadık. Ancak biz bu karakterleri **int(i)** kodunu yazarak tamsayıya çevirdik. Çünkü matematiksel işlemler yapmak için verileri sayıya çevirmemiz gerekiyor. **if** komutuyla da 3'ten büyük olanları yazdırdık.

Ör: Parola girilirken Türkçe karakter uyarısı veren program

```
tr_harfler = "ŞşÇçtiğÖöÜüİı"

parola = input("Parolanız: ")

for karakter in parola:
    if karakter in tr_harfler:
        print("Parolada Türkçe karakter kullanılamaz")
```

Ör: İlk_metin'de olan ama ikinci_metin'de olmayan öğeleri yazdırmak

```
ilk_metin = "Bilgisayar"
ikinci_metin = "Bilişim"
for s in ilk_metin:              #ilk_metin'deki her öğeye s diyoruz
    if not s in ikinci_metin:    #eğer bu öğeler ikinci_metinde yoksa
        print(s, end=" ")        #bu olmayan s'leri yazdırıyoruz
```

Çıktı: g s a y a r

while döngüsü: Bir koşul sağlanmaya devam ettiği sürece işlemleri tekrarlar. İngilizce bir kelime olan while, Türkçede '... iken, ... olduğu sürece' gibi anlamlarına gelir.

Ör:

```
while a == 1:                          # a, 1 olduğu sürece
```

Ör:

```
a = 1
while a < 10:                          # a 10'dan küçük olduğu sürece
       print("Ali")                    # ekrana "Ali" yazdır.
```

Açıklaması: Burada programımız diyor ki a, 10'dan küçük olduğu sürece ekrana "Ali" yazdır. Ancak a=1 olduğu için a her zaman 10'dan küçük olacak. Bu da ekrana sürekli "Ali" yazılmasına neden olacak. Buna sonsuz döngü diyoruz. (infinite loop) Buna son vermek için klavyenizde Ctrl+C veya Ctrl+Z tuşlarına basarak programı durmaya zorlayabilirsiniz.

#Bu durumu düzeltmek için programı şu şekilde yazalım;

```
a = 1
while a < 10:
    a += 1
    print("Ali", end=" ")
```

Çıktı: Ali Ali Ali Ali Ali Ali Ali Ali Ali

Açıklaması: Python öncelikle a = 1 satırını görüyor ve a'nın değerini 1 yapıyor. Daha sonra (while a < 10) satırını görüyor. Ardından a'nın değerini 1 artırıyor (a += 1) ve a'nın değeri 2 oluyor. a'nın değeri (yani 2) 10'dan küçük olduğu için Python ekrana ilgili çıktıyı veriyor. İlk döngüyü bitiren Python başa dönüyor ve a'nın değerinin 2 olduğunu görüyor. a'nın değerini yine 1 artırıyor ve a'yı 3 yapıyor. a'nın değeri hâlâ 10'dan küçük olduğu için ekrana yine ilgili çıktıyı veriyor. İkinci döngüyü de bitiren Python yine başa dönüyor ve a'nın değerinin 3 olduğunu görüyor. Yukarıdaki adımları tekrar eden Python, a'nın değeri 9 olana kadar dönmeye devam ediyor. a'nın değeri 9'a ulaştığında Python a'nın değerini bir kez daha artırınca bu değer 10'a ulaşıyor. Python a'nın değerinin artık 10'dan küçük olmadığını görüyor ve programı sona erdiriyor.

Ör:

```
a=1
while a<5:                          # a 5'ten küçük olduğu sürece
     print(a, end=" ")              # a'yı yazdır.
     a+=1                           # a'yı 1 arttır.
```

Çıktı: 1 2 3 4

Açıklaması: İlk satırda a değişkene 1 atadık. İkinci satıra geldiğimizde ise değişkenin 5'ten küçük olup olmama durumuna baktık, eğer küçükse kodumuz alt satıra geçecek ve böylece değişken ekrana yazdırılacak. Son koda geldiğimizde a değişkeni 1 değer artıp 2 olacak ve döngüye girecek. Sonra döngü devam edip ekrana 2 yazdırılacak. Bu durum a'nın 5'ten küçük olmaması şartına kadar sağlanacak. Yani a artık 5 olduğunda yani **while a<5:** koşulu sağlanmadığında döngü duracak ve ekrana sadece 1 2 3 4 yazdırılmış olacak

Ör: 1'den 16'ya kadar çift sayıları yazdırma:

```
a = 0
while a < 16:
    a += 1
    if a % 2 == 0:
        print(a, end=" ")
```

Çıktı: 2 4 6 8 10 12 14 16

Belirsiz Döngü: Bu döngüde döngünün kaç defa döneceği belirsizdir. Döngünün kaç defa döneceğine şartlar ve kullanıcı karar verir.

```
n = 1
karar= int(input("sayılar kaça kadar sıralansın?" ))
while n <= karar:
     print(n)
     n += 1
```

```
sayılar kaça kadar sıralansın? 7
```

Çıktı: 1 2 3 4 5 6 7

Açıklama: Bu programda döngünün kaç defa döneceği belli değildir. Kullanıcının girdiği sayı 1 veya 1'den büyük olduğu sürece döngü tekrar edecektir. Örneğin kullanıcı 7 sayısını girdiğinde 1'den 7'ye kadar olan sayılar ekrana yazdırılacaktır. Ancak kullanıcı 0 veya eksi değerde bir sayı girdiğinde while n <= karar: şartı oluşmamış olacağından program tepki vermeyecektir.