NEDEN PYTHON?

Kolay kullanılan bir programlama dilidir
Python programlama dili ile hızlı iş bitirilir
Python programlama dilinin kodlarının okunabilirliği yüksektir
Python programlama dili unicode desteklidir
Python programlama dili herşey içinde bir dildir
Python programlama dili kodları her platformda çalışır

Değişkenler A-Z yada a-z harflerinden biri, _ alt çizgi sembölü ile başlayabilir

Bunu izleyen harf, rakam veya altçizgi sembolü olabilir

Değişken taımlamalarıda özel karakterler, noktalama işaretleri kullanılmaz (@, &... Gibi)

Python büyük ve küçük harf ayrımı yapan bir dildir

Pythonda değişkenlerin türlerini tanımlamaya gerek yoktur

Python Programlama Dilinin Karakter Kümesi

Küçük Harf (a|b|c|d|e|f|g|h|i|||k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z)

Büyük Harf (A|B|C|D|E|F|G|H|I|||K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z)

Sayı (0|1|2|3|4|5|6|7|8|9)

Genel Kullanımlı İşaretler (|(|)| [|] | { | } |+|-|*|/|%|!|&| | |~|^|<|=|>|,|.|:|;|$|?|#)

Özel Kullanımlı İşaretler (' | " | \)

Python programlama dilinde, değişken isimleri, fonksiyon isimleri, nesne isimleri gibi tanıtıcı isimleri, aşağıda görülen saklı isimlerle aynı olamaz

Python Programlama Dilinde Saklı Sözcükler

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

Python programlama dilinde 30 tane saklı sözcük bulunur ve bu saklı sözcüklere Python derleyicisinin komut satırına >>> help('keywords') yazılarak erişilebilir.

Python programlarında tek satırlı yorumlar, # işaretini izleyen karakterlerdir. Python derleyicisi, bu karakterleri okumaz ve değerlendirmez.

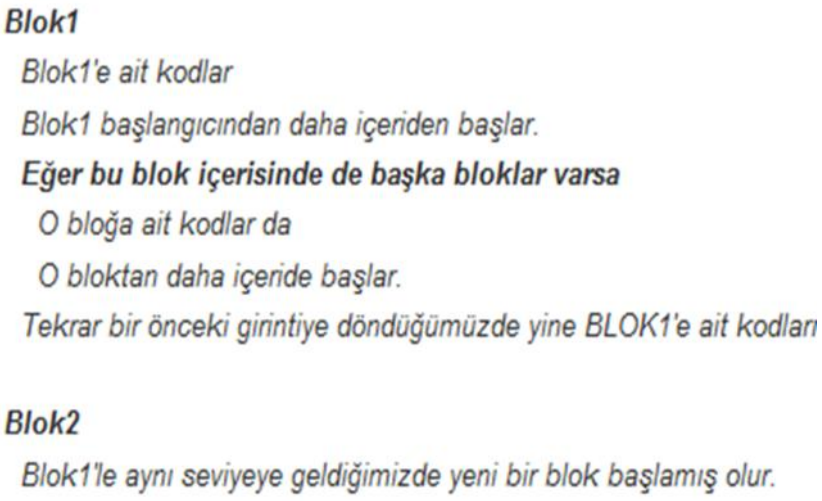
```
program adımı # Program adımıyla ilgili açıklama
```

Python programlarında birden çok satırın Python derleyicisinden saklamak için, bu satırlar """, çok satırlı yorum işaretleri arasına alınır.

```
"""
Yorum Satırları

"""
```

- Python büyük-küçük harf ayrımı yapar.
- Pythonda kod yazarken girintiler kullanılır



- Pythonda değişkenlerin türlerini tanımlamaya gerek yoktur

Klavyeden girilen bir değeri okumak

```
ad=input("Adınızı giriniz")
print(ad)
```

ayberk ismini girdiğimde programın çıktısı:

```
>>>
ayberk
>>>
```

```
sayi=int(input("Tam sayi giriniz"))
print(sayi)
```

sayi olarak 10 girdiğimde programın çıktısı:

```
>>>
10
>>>
```

int , girilen değeri string türünden tamsayi türüne (integer) çevirdi.

```
sayi=input("Tam sayi giriniz")
print(sayi)
```

sayi olarak 10 girdiğimde programın çıktısı:

```
>>>
10
>>>
```

```
sayi=float(input("Tam sayi giriniz"))
print(sayi)
```

sayi olarak 10 girdiğimde programın çıktısı:

```
>>>
10.0
>>>
```

Float , girilen değeri string türünden ondalık sayı türüne(float) çevirdi.

Ekrana yazdırma

python'da ekrana yazdırma işlemleri print() komutu ile gerçekleşir.

```
>>> a = 1234.56789
>>> b = [2, 4, 6, 8]
>>> print (a,b)
1234.56789 [2, 4, 6, 8]
>>> print ('a =',a, '\nb =',b)
a = 1234.56789
b = [2, 4, 6, 8]


>>> b = 2 # b tamsayi türünde
>>> print (b)
2
>>> b = b*2.0 # b ondalık sayı türünde
>>> print (b)
4.0
```

String

```
>>> string1 = 'Cikmak için entere basiniz.'
>>> string2 = 'programdan.'
>>> print (string1 + ' ' + string2) # iki string türünün toplanması
'Cikmak için entere basiniz. Programdan.
>>> print (string1[0:12])  # Bastan 11. Karaktere kadar yazar
Cikmak için

>>> s = 'Press return to exit'
>>> s[0] = 'p'
Traceback (most recent call last):
File ''<pyshell#1>'', line 1, in ?
s[0] = 'p'
TypeError: object doesn't support item assignment
```

Stringler

Matematiksel işlemlere sokulmayan karakterlerden oluşan sözel ifadelerdir.

‘ tek tırnak, ‘’ tırnak yada ‘’’ veya ‘’’’’ arasında yazılırlar

Stringler + operatörü ile birleştirilirler.

Stringlerin : operatörü ile belirli bir kısmı ayrılabilir.

```
>>> string1 = ’Ayberk.’
>>> string2 = ’YILMAZ
>>> print (string1 + ’ ’ + string2) # iki string türünün toplanması
Ayberk YILMAZ

>>> print (string1[0:4]) # Bastan 4 Karakter kadar yazar
Aybe

>>> len(string1)
6

>>> s = ’Papatya’
>>> s[0] = ’p’
Traceback (most recent call last): File ’’’’, line 1, in ? s[0] = ’p’ TypeError: object doesn’t
support item assignment
```

Stringler değiştirilemez objelerdir ve belirli bir uzunluğa sahiptirler.

Tamsayılar (Integers)

Ondalık nokta içermeyen nümerik değerlerdir.

```
>>> 35+40

75

>>> 13/3
4.333333333333333

>>> divmod(10,7)
(1, 3)
```

```
>>> 6/4
1.5

>>> int(6/4)
1

>>> 6//4
1
```

Kayan Noktalı Sabitler

Ondalık sayı veya üstel sayılardır.

```
>>> 10.0/2.5
4

>>> 1.2e3/3
400.0

>>> 2*3.0
6.0
```

Karmaşık sayılar (Complex)

Matematikte a+bi şeklinde ifade edilen sayılardır. Burada a sayının gerçel (real) kısmı b ise sanal (imaginary) kısmını ifade etmektedir.

```
>>> 5+3j
(5+3j)

>>> complex(5,3)
(5+3j)

>>> 6j*6j
(-36+0j)

>>>(1+2j)/(1+1j)
(1.5+0.5j)
```

```
>>> a=4-3j
>>> print(a)
(4-3j)
>>> print(a.real)
4.0
>>> print(a.imag)
-3.0
>>> print(ab(a))
5.0
>>> b=a.conjugate()
(4+3j)
```

Aritmetik operatörler

Operatör	İşlem	Örnek
=	Atama Operatörü	A=4, b=’Oku’
+	Toplama	A=4+86
-	Çıkarma	A=86-52
*	Çarpma	A=9*86
/	Bölme	A=86/12
**	Kuvvet alma	A=8**3 (8′in 3.kuvveti)

```
>>> s = ’Merhaba ’
>>> t = ’sana’
>>> a = [1, 2, 3]
>>> print (3*s) # Tekrarlamalı yazım
Merhaba Merhaba Merhaba
>>> print (3*a)
[1, 2, 3, 1, 2, 3, 1, 2, 3]
>>> print (a + [4, 5]) # Eleman ekleme
[1, 2, 3, 4, 5]
>>> print (s + t) # İki string türünü ekleme
Merhaba sana
>>> print (3 + s) # Farklı değişken türleri (integer ve string)
Traceback (most recent call last):
File ’’<pyshell#9>’’, line 1, in ?
print n + s
TypeError: unsupported operand types for +: ’int’ and ’str’
```

Krşılaştırma Operatörleri

a+=b	a=a+b
a-=b	a=a-b
a*=b	a = a*b
a/=b	a = a/b
a **= b	a = a**b
a%=b	a = a%b

>	Büyüklük	A>B (A, B' den büyüktür)
>=	Büyüklük ya da eşitlik	A>=B (A, B' ye eşit veya büyük)
<	Küçüklük	A<B (A, B' den küçüktür)
<=	Küçüklük ya da eşitlik	A<=B (A, B' den küçük veya eşit)
==	Eşitlik	A=B (A, B' ye eşit)
!=	Eşit değil	A**!=**B (A, B' ye eşit değil)

and	Ve işlemi	a == 3 and b == 12
or	Veya İşlemi	a == 5 or b == 43
not	Değil İşlemi	a % 2 == 0 değilse
//	Bölümü verir	20//2 Sonuç=10
Pow(x,y)	Kuvvet alır	Pow(3,4) Sonuç=81
abs(x)	Mutlak Değer	Abs(-10) Sonuç=10

```
>>> a = 2 # Integer türünde
>>> b = 1.99 # Float türünde
>>> c = '2' # String
>>> print (a > b)
1
>>> print (a == c)
0
>>> print ((a > b) and (a != c))
1
>>> print ((a > b) or (a == b))
1
```

Koşul İfadeleri

if-else yapısı

```
if <şart>:
    doğruysa işletilecek blok
elif <şart>:
    doğruysa  işletilecek blok
else:
    koşul sağlanmıyorsa işletilecek blok
```

```
a=5
if a < 0.0:
    print( ’negatif’)
elif a > 0.0:
    print(’pozitif’)
else:
    print(’sıfır’)

>>>
a is pozitif
>>>
```

Döngüler

```
while <şart>:
    doğruysa  işletilecek blok
else:
   koşul sağlanmıyorsa işletilecek blok
```

```
nMax = 5
n=1
a = [] # Boş bir liste oluşturduk
while n < nMax:
     a.append(1.0/n) # listeye eleman ekleniyor
     n=n+1
print (a)

>>>

[1.0, 0.5, 0.3333333333333331, 0.25]
>>>
```

```
for i in range (başlangıç,bitiş,artışmiktari):
                    bitiş sayısına veya daha büyük bir sayıya ulaşana kadar işletilecek blok

nMax = 5
a=[]
for n in range(1,nMax):
        a.append(1.0/n)
print (a)
>>>
[1.0, 0.5, 0.33333333333333331, 0.25]
>>>
```

pass bildirimi
pass Bildirimi, program kontrolüne, döngüye girmeden pas geçmesini bildirir. Bu bildirim ile program kontrolü, döngüden bir sonraki program adımını çalıştırır.

continue bildirimi
bu bildirim, program kontrolüne çalışılmakta olan iterasyondan çıkılarak, döngüye bir sonraki iterasyondan devam edilmesini bildirir.

break bildirimi
Bu biildirim, program kontrolüne döngüden derhal çıkılarak, programa döngüden bir sonraki program adımından itibaren devam edilmesini bildirir. İç içe döngüler ile çalışıldığında, breakbildirimine rastlandığı anda, iç döngüden çıkılır ve dış döngü çalıştırılmaya devam edilir.

Bazen bir döngü else bildirimini de içerebilir. Bu durumda döngü çıkışında bu else bloğunun içeriği çalıştırılır. Fakat break bildirimi bunu da önler ve program kontrolüne döngüden elsebloğu dahil derhal çıkılarak, programa döngüden bir sonraki program adımından itibaren devam edilmesini bildirir.

```
for i in ("Ocak" , "Mart" , "Nisan"):
     if i == "Mart":
           continue;
print(i);
```

```
for j in range(1,10):
       if j == 3:
           break
print(j)
```

Tip dönüşümleri

int(a)	a yi integer türüne dönüştürür
float(a)	a yi float türüne dönüştürür
complex(a,b)	a ve b yi kopleks sayıya dönüştürür *a +bj*

```
>>>a=5
>>> b = -3.6
>>> d = ’4.0’
>>> print (a + b)
1.4
>>> print int(b)
-3
>>> print complex(a,b)
(5-3.6j)
>>> print float(d)
4.0
>>> print int(d) # This fails: d is not Int type
Traceback (most recent call last):
File ’’<pyshell#7>’’, line 1, in ?
print int(d)
ValueError: invalid literal for int(): 4.0
```