while döngüsü

```
i = 0

while (i < 10):
    print("i' nin değeri", i)
    i += 1

i = 1

while (i <= 20):
    print("i' nin değeri", i)
    i += 2

u = 0

while (u <= 10):
    print("Python Çalışıyoruz...")
    u += 1

liste = [1,2,3,4,5, "python","kağan"]

for i in liste:
    print(i)

len(liste)

liste[2]

liste

i = 0

while (i < len(liste)):
    print("index :", i, "Liste elemanı", liste[i])
    i += 1
```

range() fonksiyonu

```
range(0,20)

print(range(0,20))

print(*range(0,20))

print(*range(0,101))

print(*range(15))
print(*range(0,100,2))

print(*range(0,100,3))

print(*range(20,0,-1))

liste = [1, 2, 3, 4, 5, 6, 7, 8, 9]

for i in liste:
    print(i)

for i in range(1,21):
    print(i)

for i in range(0,21):
    print(* "*" * i)
```

Break ve Continue

break ifadesi

```
i = 0

while (i < 10):
    print("i:",i)
    i += 1

i = 0

while (i < 10):

    if (i == 5):
        break
    print("i:",i)
    i += 1

liste = [1,2,3,4,5]

for i in liste:
    if(i == 3):
        break
    print("i:", i)

while True:
    isim = input("isim (çıkmak için 'q' ya basınız: )")
    if (isim == 'q'):
        print("programdan çıkılıyor...")
        break
    print("isminiz:", isim)
```

continue ifadesi

```
liste = list(range(0,11))
print(liste)

for i in liste:
    print("i:", i )

for i in liste:
    if( i == 3 or i == 5):
        print("i:", i)
        print("****************************")
        print("i:", i)

liste

for i in liste:
    if( i == 3 or i == 5):
        print("i:", i)

for i in liste:
    if( i == 3 or i == 5):
        print("xxxxxxxxxxx=",i)
        continue
    print("i:", i)

i = 0

while (i < 10):
    print("zzzz")
    if (i == 2):
        print("xxxxxxxx=",i)
        i += 1
        continue

    print ("i:",i)
    i += 1
```

```
zzzz
i: 0
zzzz
i: 1
zzzz
xxxxxxxx= 2
zzzz
i: 3
```

```
zzzz
i: 4
zzzz
i: 5
zzzz
i: 6
zzzz
i: 7
zzzz
i: 8
zzzz
i: 9
```

```
i = 0

while (i < 10):
    if (i == 2):
        i+=1
        continue

    print ("i:",i)
    i += 1
```

```
i: 0
i: 1
i: 3
i: 4
i: 5
i: 6
i: 7
i: 8
i: 9
```

Mantıksal Değerler ve Karşılaştırma Operatörleri

```
a = True
print(type(a))
```

```
<class 'bool'>
```

```
b = False
print(type(b))
```

```
<class 'bool'>
```

```
bool(3), bool(-1.71), bool(0), bool(0.0)
```

```
(True, True, False, False)

1 == 1

True

1 == 2

False

c = None
print(c)
print(type(c))

None
<class 'NoneType'>

c = 4
c
```

Karşılaştırma Operatörleri

```
"Apple" == "Apple"

True

"Apple" != "Samsung"

True

[1,2,3] == [1,2,3]

True

4 > -20

True

"Araba" > "Araba"

False

"Uçak" <= "Uçal"

True
```

Mantıksal Bağlaçlar

and

```
1 < 2 and "Western Digital" == "Western Digital"
```

```
True
```

```
2 > 9 and 7 > 6
```

```
False
```

```
help("and")
```

```
Boolean operations
******************

   or_test  ::= and_test | or_test "or" and_test
   and_test ::= not_test | and_test "and" not_test
   not_test ::= comparison | "not" not_test

In the context of Boolean operations, and also when expressions are
used by control flow statements, the following values are interpreted
as false: "False", "None", numeric zero of all types, and empty
strings and containers (including strings, tuples, lists,
dictionaries, sets and frozensets).  All other values are interpreted
as true.  User-defined objects can customize their truth value by
providing a "__bool__()" method.

The operator "not" yields "True" if its argument is false, "False"
otherwise.

The expression "x and y" first evaluates *x*; if *x* is false, its
value is returned; otherwise, *y* is evaluated and the resulting value
is returned.

The expression "x or y" first evaluates *x*; if *x* is true, its value
is returned; otherwise, *y* is evaluated and the resulting value is
returned.

Note that neither "and" nor "or" restrict the value and type they
return to "False" and "True", but rather return the last evaluated
argument.  This is sometimes useful, e.g., if "s" is a string that
should be replaced by a default value if it is empty, the expression
"s or 'foo'" yields the desired value.  Because "not" has to create a
new value, it returns a boolean value regardless of the type of its
argument (for example, "not 'foo'" produces "False" rather than "''".)
```

```
Related help topics: EXPRESSIONS, TRUTHVALUE
```

```
3 > 9 and 1 > 0.1 and "JPhone" == "JPhone" and "KBookPro" ==
"KBookPro"
```

```
False
```

or

```
1 < 2 or "Western Digital" == "Toshiba"
```

```
4 < 2 or "Western Digital" == "Toshiba"
```

not

```
not (2 == 2)
```

```
False
```

```
not (3 > 2 and 1 > 0.1 and "JPhone" == "JPhone" and "KBookPro" ==
"KBookPro")
```