1. Değişken tanımlama, type ve print fonksiyonları

Değişken ismi verilirken

- Değişken ismi sayı ile başlayamaz.
- Değişken ismi, isim tamlaması gibi ise kelimeler arasında boşluk bulunamaz.
- :'",<>/?|()!@#$%^&*~-+ simgeleri değişken isminde kullanılamaz. Sadece alt çizgi _ kullanılabilir.
- Python' da tanımlı while, not gibi özel kelimeler değişken ismi olarak kullanılamaz.

```
from termcolor import colored, cprint
print(colored("\nDeğişken ismi verilirken", "red",
attrs=["underline"]))
print(colored("\n-\t Değişken ismi sayı ile başlayamaz.", "blue",
attrs=["bold"]))
print(colored("\n-\t Değişken ismi, isim tamlaması gibi ise kelimeler
arasında boşluk bulunamaz.", "green"))
print(colored("\n-\t : ' ” , < > / ? | \ ( ) ! @ # $ % ^ & * ~ - +
simgeleri değişken isminde kullanılamaz.",
              "magenta", attrs=["dark", "bold"]))
cprint("\n-\tSadece alt çizgi _ simgesi, isim tamlaması şeklindeki
değişken isimleri için kullanılabilir.", "cyan")
cprint("\n-\tPython' da tanımlı ", "blue", end = " ")
cprint("while", 'red', attrs=["bold"], end = " ")
cprint("ve", "blue", end = " ")
cprint("not", "red", attrs=["bold"], end = " ")
cprint("gibi özel kelimeler değişken ismi olarak kullanılamaz. ",
'blue', end = " ")

# end = " " parametresi, print fonksiyonun yeni satıra geçmeden
# metnin yazılmasını sağlar.
```

```
Değişken ismi verilirken

-	Değişken ismi sayı ile başlayamaz.

-	Değişken ismi, isim tamlaması gibi ise kelimeler arasında boşluk
bulunamaz.

-	: ' ” , < > / ? | \ ( ) ! @ # $ % ^ & * ~ - + simgeleri değişken
isminde kullanılamaz.
```

```
-    Sadece alt çizgi _ simgesi, isim tamlaması şeklindeki değişken isimleri için kullanılabilir.

-    Python' da tanımlı  while ve not gibi özel kelimeler değişken ismi olarak kullanılamaz.
```

```
a = 2
print(a) # print fonksiyonu
```

```
2
```

```
pi_sayısı = 3.141592653589793
pi_sayısı
```

```
3.141592653589793
```

Soru: 4 cm yarıçaplı bir dairenin alanı ve çevresi

```
r = 4
çevre = 2 * pi_sayısı * r
alan = pi_sayısı * r ** 2

print(çevre)
print(alan)
```

```
25.132741228718345
50.26548245743669
```

```
print("çevre=", çevre)
print("alan=", alan)
```

```
çevre= 25.132741228718345
alan= 50.26548245743669
```

```
x, y = 40, 64

print(x)
print(y)
```

```
40
64
```

```
x , y = y , x
print("x = ",x)
print("y = ",y)
```

```
x =  64
y =  40
```

```
# yorum satırı

# seri toplama
```

```
i = 1

i = i + 1
i

2

i += 1
i

3

# seri çıkarma

i = 100

i = i - 1
i

99

i -= 1
i

98

# seri çarpma

x = 6

x = x * 4
x

24

x *= 4
x

96

# seri bölme

j = 100

j = j/2
j

50.0

j /= 2
j

25.0
```

tamsayı bölme (//)

```
t, x = 100, 3
t/x
33.333333333333336
t // x
33
```

kalanı bulma (%)

```
t % x
1
40 % 20
0
```

üs bulma (**)

```
2 ** 4
16
```

1/2=0.5 olduğundan bir sayının kare kökünü **0.5 işlemi ile hesaplatabiliriz

```
36 ** 0.5
6.0
```

işlem sırası

- Her zaman için parantez öncelikli işlem sıradır.
- Çarpma ve bölme toplama ve çıkarma işlemine göre önce yapılır.
- İşlemler soldan sağa değerlendirilir.

```
12 + (4 - 7) * 3
3
(12 + (4 - 7)) * 3
27
((10 ** 2) // 3) / 3
```

```
11.0
```

type() fonksiyonu

```
type(4)
```

```
int
```

```
type(3 + 2j)
```

```
complex
```

```
type(3.14)
```

```
float
```

```
type("Mustafa Kemal")
```

```
str
```

```
liste = ["Erol Evgin"]
```

```
type(liste)
```

```
list
```

```
u = (1,2)
```

```
type(u)
```

```
tuple
```

```
A = {"sütun1" : [1,2,3], "B" : ["x1", "x2", "x3"]}
```

```
type(A)
```

```
dict
```

```
def sigma(x):
    return x
```

```
type(sigma)
```

```
function
```

```
type(print)
```

```
builtin_function_or_method
```

Print fonksiyonunun özellikleri

```
print(3.14)
```

```
3.14
```

```
x = "Gökhan"
x
```

```
'Gökhan'
```

```
print(x)
```

```
Gökhan
```

```
print("Vatanını en çok seven, görevini en iyi yapandır. Mustafa Kemal Atatürk")
```

```
Vatanını en çok seven, görevini en iyi yapandır. Mustafa Kemal Atatürk
```

```
print("Vatanını en çok seven, görevini en iyi yapandır.\nMustafa Kemal Atatürk")
```

```
Vatanını en çok seven, görevini en iyi yapandır.
Mustafa Kemal Atatürk
```

```
print("Vatanını en çok seven, görevini en iyi yapandır.\n\t\t\t\t\t\tMustafa Kemal Atatürk")
```

```
Vatanını en çok seven, görevini en iyi yapandır.
                                    Mustafa Kemal Atatürk
```

```
# print içinde birden fazla parametre kullanılabilir
```

```
print(3.14, "çay olsa da içsek", 10**3)
```

```
3.14 çay olsa da içsek 1000
```

```
print(3.14, ",", "çay olsa da içsek",",", 10**3)
```

```
3.14 , çay olsa da içsek , 1000
```

print içinde sep parametresi

```
print(3.14, 56, "FB", 792.14)
```

```
3.14 56 FB 792.14
```

```
print(3.14, 56, "FB", 792.14, sep = ", ")
```

```
3.14, 56, FB, 792.14
```

```
print(3.14, 56, "FB", 792.14, sep = "*****")
```

```
3.14*****56*****FB*****792.14
```

```
print(3.14, 56, "FB", 792.14, sep = " + ")
```

```
3.14 + 56 + FB + 792.14
```

```
print(3.14, 56, "FB", 792.14, sep = "//")
```

```
3.14//56//FB//792.14
```

```
print("Erol","Evgin", sep = "\n")
```

```
Erol
Evgin
```

```
print("Erol","Evgin", sep = "\t")
```

```
Erol	Evgin
```

```
print("19","Mayıs","1919", sep = "/")
```

```
19/Mayıs/1919
```

* Yıldız parametresi

```
print("Python")
```

```
Python
```

```
print(* "Python")
```

```
P y t h o n
```

```
print("T","B","M","M")
print("T","B","M","M", sep =".")
```

```
T B M M
T.B.M.M
```

```
print(*"TBMM ", sep =".")
```

```
T.B.M.M.
```

Çıkış Formatlama

```
# Belirli sayıdaki süslü parantez {} kullanılarak aynı sayıdaki
nicelik
# süslü parantezlerle yer değiştirilerek yazdılır
```

```
"{}, {}, {}".format(3.14, 9, 1001)
```

```
'3.14, 9, 1001'
```

```
"{}{}{}".format(3.14,9,1001)
```

```
'3.1491001'
```

```
"{} {} {}".format(3.14,9,1001)
```

```
'3.14 9 1001'
```

```
x = 9
y = 409

print("{} + {} 'un toplamı {}'dir.".format(x, y, (x+y) ))
```

```
9 + 409 'un toplamı 418'dir.
```

```
x, y = 9, 409

print("{} + {} 'un toplamı {}'dir.".format(x, y, (x+y) ))
```

```
9 + 409 'un toplamı 418'dir.
```

```
# Süslü parantez içine sayılar yazılırsa, bu indeks numaralarına karşı
gelir
# format içindeki dizi, indeks numaralarına göre parantezlerle yer
değişir.

"{2} {1} {0} {3}".format("Çalış","Öğün","Türk","Güven")
```

```
'Türk Öğün Çalış Güven'
```

```
# Sülü parantezin aşağıdaki gibi kullanımında, ondalıklı sayıların
# belirtilen basamak sayısıyla yazılması sağlanır

"{:.2f} {:.4f}
{:.8f}".format(3.14159265358979,3.14159265358979,3.14159265358979,3.14
159265358979)
```

```
'3.14 3.1416 3.14159265'
```

```
# Sülü parantezin aşağıdaki gibi kullanımında, ondalıklı sayıların
# belirtilen basamak sayısıyla yazılması sağlanır

"{:.2f} - {:.4f} -
{:.8f}".format(3.14159265358979,3.14159265358979,3.14159265358979,3.14
159265358979)
```

```
'3.14 - 3.1416 - 3.14159265'
```

f string

f-string, karakter dizilerini biçimlendirmek için kullanılan bir yöntemdir.

f-string kullanmak için, karakter dizisinin başına f veya F harfi eklenir.

Böylece Python, karakter dizisinin normal bir karakter dizisi olmadığını ve içindeki süslü parantezler ({}) arasındaki ifadeleri değerlendirmesi gerektiğini anlar.

f-string, format() fonksiyonuna benzer bir işlev görür, ancak daha kısa ve daha okunaklı bir yazım sunar.

```
x, y = "Ford Mustang", "60 saniye"
print(f"{x} için yapılan en iyi film {y} isimli filmdir.")
```

```
Ford Mustang için yapılan en iyi film 60 saniye isimli filmdir.
```

```
x, y = 8, 16
print(f"x = {x} ve y = {y} olarak tanımlanmıştır.")
```

```
x = 8 ve y = 16 olarak tanımlanmıştır.
```