Python Programlama Dili

Aşağıdaki linkte Python diline ait resmi sitenin adresi verilmiştir.

The Python Tutorial

https://docs.python.org/3/ (https://docs.python.org/3/)

https://docs.python.org/2.7/tutorial/index.html (https://docs.python.org/2.7/tutorial/index.html)

Python'la çalışma imkanı veren bazı siteler

https://www.dataschool.io/cloud-services-for-jupyter-notebook/ (https://www.dataschool.io/cloud-services-for-jupyter-notebook/)

Değişken:

Yapılacak olan bir matematiksel veya benzeri bir
başka işlemi genelleştirebilmek
için kullanılan araçlardır (sembollerdir).

Örneğin, iki sayıyı toplayacak bir program yazacaksak
bir kullanıcı tarafından verilecek herhangi iki sayıyı
temsil edebilecek a ve b olarak isimlendireceğimiz
iki değişken kullanabiliriz.

Böylece,

toplam = a + b

gibi bir işlem tanımı yapabiliriz. Buradaki a ve b
değişkenleri herhangi iki sayısal değeri alabilir.

In [1]:
```
a = 15
b = 16
############################
toplam = a + b
print(toplam)
```

```
31
```

Değişken adı belirleme kuralları

Aşağıdaki örnekte görüldüğü üzere
sayıyla başlayan bir değişken adı olamaz.

Böyle bir değişken adı yazılırsa aşağıdaki
"SyntaxError: invalid syntax" hatasıyla
karşılaşılır.

Bu hatanın anlamı Python'un yazım kurallarına
uymadığınızdır.

In [2]:
```
3_kilo_elma = "5 TL"
```

```
  File "<ipython-input-2-265f9cc746c5>", line 1
    3_kilo_elma = "5 TL"
               ^
SyntaxError: invalid syntax
```

In [2]:
```
üç_kilo_elma = "5 TL"
print(üç_kilo_elma)
```

```
5 TL
```

In [3]:
```
_deger = 51
```

In [4]:
```
elmanin_kutlesi = 23
```

In [21]:
```
False = 25
```

```
  File "<ipython-input-21-68f64a77f341>", line 1
    False = 25
           ^
SyntaxError: can't assign to keyword
```

In [5]:
```
False38 = 25
False_ = 25
false = 25
false_ = 25
```

In [12]:
```
# Değişken adı olarak Türkçe karakterler kullanılabiliyor.
# Yine de kullanmanızı pek tavsiye etmem...
Türkçe_Harfleri_Tanıyor = "Gerçekten"
print(Türkçe_Harfleri_Tanıyor)
```

```
Gerçekten
```

In [7]:
```
# Değişken adı olarak Çince karakterler bile kullanılabiliyor.
中國佬 = 10**9
print(中國佬)
print(中國佬/中國佬)
```

```
1000000000
1.0
```

Değişken adı olarak kullanılamayacak Python'a ait anahtar kelimeler (keywords)

In [5]:
```
import keyword
keyList = keyword.kwlist
Lkey = len(keyList)

print(keyList[:11])
print(keyList[11:22])
print(keyList[22:])
```

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

Python'da nesne tipleri

Temel 5 nesne tipi vardır:

- Numbers: Sayılar
- String : Metin veya karakter tipinde veri
- List : Küme şeklindeki veriler
- Tuple : Küme şeklindeki veriler ama korunaklı
- Dictionary : Küme şeklinde veri, her bir veri bir anahtar kelimeyle çağırılabilir.
- Function:
- Class:
- Boolean: Mantık tipi

Sayı tipi:

Şunlar örnek verilebilir.

1. reel sayılar: Python'da float tipi olarak adlandırılır.
2. tam sayılar: Python'da int tipi olarak adlandırılır. Integer kelimesinin kısaltmasıdır.
3. karmaşık sayılar: Python'da complex tipi olarak adlandırılır.

type() komutu

Bu komutla bir nesnenin tipi öğrenilebilir.

In [8]:
```
# 5 sayısının tipi int yani tam sayıdır.
type(5)
```

Out[8]:
```
int
```

In [9]:
```
# 5. sayısının tipi float yani reel sayıdır.
type(5.)
```

Out[9]:
```
float
```

In [10]:
```
# 5j sayısın tipi complex yani karmaşık sayı tipidir.
type(5j)
```

Out[10]:
```
complex
```

In [35]:
```
type(5+5j)
```

Out[35]:
```
complex
```

In [11]:
```
j
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-11-3eedd8854d1e> in <module>()
----> 1 j

NameError: name 'j' is not defined
```

In [16]:
```
# matematikteki "i" sanal sayısı
1j # 'dir
```

Out[16]:
```
1j
```

In [17]:
```
1j**2
```

Out[17]:
```
(-1+0j)
```

In [13]:
```
type(0 + 0j)
```

Out[13]:
```
complex
```

In [37]:
```
# en büyük sayı kümesi işlem sonucunda elde edilen
# sayının tipini verir
type((5+5j)*5.)
```

Out[37]:
```
complex
```

In [38]:
```
type(5*13)
```

Out[38]:
```
int
```

In [39]:
```
type(5*13.0)
```

Out[39]:
```
float
```

String tipi

Sayısal olmayan verileri temsil edebilen bir tiptir. Örneğin, bir telefon rehberi programı yazmak istiyoruz. Burada şahıs adları,
adres bilgileri, email bilgileri v.b. yukarıdaki üç tiple temsil edilemez
veya temsil zordur.

Bu tipteki veri, verinin etrafına tırnak işaretleri konularak oluşturulur.

In [42]:
```
type("mesut")
```

Out[42]:
```
str
```

In [46]:
```
type('mesut')
```

Out[46]:
```
str
```

In [51]:
```
type(
"""
MEY BİTER SAKİ KALIR,
HER RENK SOLAR HAKİ KALIR,
DİPLOMA İNSANIN CEHLİNİ ALSA DA,
MAYASINDA VARSA EŞEKLİK BAKİ KALIR
""")
```

Out[51]: str

In [55]:
```
print("4" + "5")
type("4" + "5")
```

```
45
```

Out[55]: str

In [3]:
```
ad = "mesut"
soyad = "karakoç"
ad
print(ad + " " + soyad)
```

```
mesut karakoç
```

In [58]:
```
print(32*"*")
type(32*"*")
```

```
********************************
```

Out[58]: str

Liste tipi (List type)

Küme veya dizi şeklindeki verilerdir.

In [59]:
```
# Türkçenin ilk 6 harfini içeren bir liste tipi veri
["A", "B", "C", "Ç", "D", "E"]
```

Out[59]: ['A', 'B', 'Ç', 'D', 'E']

In [60]:
```
type(["A", "B", "C", "Ç", "D", "E"])
```

Out[60]: list

In [61]:
```
# boş liste
[]
```

Out[61]: []

In [62]:
```
type([])
```

Out[62]: list

Mantık (bool) Tipi

Mantık işlemleri sonucu ortaya çıkan verilerin tipidir.

Buna iki örnek Python'da ön tanımlı olarak varolan True ve False ifadeleridir.

```
>>> type(True)
<class 'bool'>
```

ARİTMETİK İŞLEMCİLER

+, -, *, /, %, **
toplama, çıkarma, çarpma, bölme, mod, kuvvet

işlem önceliği:

- * ve / işlemcileri +,- işlemcilerinden önceliklidir.
- ** ise *,/,+,- şlemcilerinden önceliklidir.
- işlem önceliğini parantezler "()" kullanarak değiştirebiliriz.

```
>>> 2+3/4
2.75
>>> (2+3)/4
1.25
```

MANTIK İŞLEMCİLERİ:

and, or, not
ve, veya, değil

ve (and) işlemcisi:

```
>>> True and False
False
>>> True and True
True
>>> False and True
False
>>> False and False
False
```

veya (or) işlemcisi:

```
>>> True or False
True
>>> False or True
True
>>> True or True
True
>>> False or False
False
```

değilleme (not) işlemcisi:

```
>>> not True
False
>>> not False
True
```

```
A B|A ve B |A veya B | değil A
-----------|---------|--------
0 0|  0    |    0    |    1
0 1|  0    |    1    |    1
1 0|  0    |    1    |    0
1 1|  1    |    1    |    0
```

```
0: False (Yanlış)
1: True  (Doğru)
```

Karşılaştırma işlemcileri:

- == eşit midir?
- > büyük müdür?
- < küçük müdür?
- >= büyük eşit midir?
- <= küçük eşit midir?
- != eşit değil midir?

```
>>> 5 == 5
True
>>> 5 > 5
False
>>> 5 < 5
False
>>> 5 >= 5
True
>>> 5 > 5 or 5==5
True
>>> 5 <= 5
True
>>> 5 != 5
False
```

tipleri farklı nesnelerin karşılaştırılması

```
>>> 5 == '5'
False
>>> 5 == int('5')
True
>>> str(5) == '5'
True
```

Tip dönüşümleri:

Daha önce bahsi geçen tipleri birbirine dönüştürmek mümkündür.

reel sayıyı tam sayıya dönüştürme: int() ifadesiyle yapılır.

```
>>> int(123.45)
123
>>> type(int(123.45))
<class 'int'>
```

tam sayıyı reel sayıya dönüştürme: float() ifadesiyle yapılır.

```
>>> float(123)
123.0
>>> type(float(123))
<class 'float'>
```

string tipini sayıya dönüştürme: int() veya float() kullanılabilir. Yeterki string verisi sayıya dönüştürülebilecek türde olsun.

```
>>> type(float("3.14159"))
<class 'float'>
```

sayıya çevrilmesi mümkün olmayan string:

```
>>> float("mesut")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: could not convert string to float: 'mesut'
```

sayısal ifadeleri string'e dönüştürme: str() ifadesiyle gerçekleştirilebilir.

```
>>> a = 3.14159
>>> b = str(a)
>>> a
3.14159
>>> b
'3.14159'
>>> type(a)
<class 'float'>
>>> type(b)
<class 'str'>
```

Python programları ve kullanılan komutların tanımları

İki sayıyı toplama

In [18]:
```
a = float(input('a = ?'))
b = float(input('b = ?'))

Toplam = a + b

print(Toplam)
```

```
a = ? 3b = ? 47.0
```

In [20]:
```
a = input('a = ?')
b = input('b = ?')

af = float(a)
bf = float(b)

Toplam = af + bf

print(Toplam)
```

```
a = ? 12b = ? 3446.0
```

Bu program da print, input ve float Python komutları kullanılmıştır.

input:

kullanıcıdan veri almak için kullanılır.

kullanımı:

```
a = input('veri = ?')
```

kullanıcıdan alınan veri a değişkenine string tipi bir veri olarak aktarılır.

In [6]:
```
a = input('veri = ?')
type(a)
```

```
veri = ? 3.14159
```

Out[6]:
```
str
```

print:

verilerin ekrana yazılmasını sağlar.

kullanımları:

In [7]:
```
print("Mesut")
```

```
Mesut
```

In [8]:
```
ad = "Mesut"
print(ad)
```

```
Mesut
```

format metodu

format string tipi verilerin düzenlenmesini sağlar.

https://pyformat.info/ (https://pyformat.info/)

In [15]:
```
alan = 50 #m^2
print("{} m^2".format(alan))
```

```
50 m^2
```

In [16]:
```
pi = 3.14159265 #m^2
print("{:.2f} m^2".format(pi))
```

```
3.14 m^2
```

Büyük olan sayıyı bulan program

In [5]:
```
a = float(input('a = ?'))
b = float(input('b = ?'))

if a>b:
    buyuk = a
else:
    buyuk = b

print(buyuk)
```

```
a = ? 6b = ? 46.0
```

if ... then ... else ...

bir programdaki verilerin karşılaştırılmasına imkan verir ve
bu sayede programın takip edecepi yol belirlenir.
Burada "if" kelimesinin türkçe karşılığı "eğer", "else" kelimesinin karşılığı da "değilse" dir.

Python'da blok yapısı

```
Blok:
#234 açıklama
    komut 1
    komut 2
komut 3
```

Yukarıdaki gösterimdeki "Blok" yerine if, while, for, try vb. komutlar
gelebilir. Bunlar blok yapısına sahip komutlardır. Yukarıdaki gösterimde
komut 1 ve komut 2 "Blok:"'a aittir. komut 3 "Blok:"'tan bağımsızdır.

While döngüsü ile 1'den 10'a sayan program

In [6]:
```
i=0
while i<10:
    i = i + 1
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

while

Aşağıdaki örnekteki gibi kullanılır. Örnekteki durumda (şart) doğru olduğu sürece while bloku içindeki komutları tekrarlayarak gerçekleştirir.

kullanımı:

```
while (şart):
    komutlar
```

Sonsuz döngü

In [0]:
```
while True:
    print("Sonsuzluk")
```

range()

bir tam sayılar listesi oluşturur.

kullanımı:

In [7]:
```
# bu haliyle 0'dan 4'e kadar
# olan sayıları üretebilir.
range?
```

Out[7]: range(0, 5)

In [9]:
```
range(5)
```

Out[9]: range(0, 5)

In [8]:
```
range(0, 5, 1)
```

Out[8]: range(0, 5)

In [7]:
```
list(range(5))
```

Out[7]: [0, 1, 2, 3, 4]

In [8]:
```
list(range(1, 6))
```

Out[8]:
```
[1, 2, 3, 4, 5]
```

for

bir döngü komutudur. kendisine verilen liste tipi veriyi okuyarak döngü oluşturur.

kullanımı:

In [33]:
```
for i in range(1, 11):
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

In [11]:
```
for i in "mesut":
    print(i)
```

```
m
e
s
u
t
```

In [35]:
```
for i in ["Ankara", "İzmir", "Kayseri", "Samsun", "Tokat"]:
    print(i)
```

```
Ankara
İzmir
Kayseri
Samsun
Tokat
```

In [36]:
```
j = 0
for i in ["Ankara", "İzmir", "Kayseri", "Samsun", "Tokat"]:
    j = j + 1
    print(j, i)
```

```
1 Ankara
2 İzmir
3 Kayseri
4 Samsun
5 Tokat
```

enumerate

for ile beraber kullanıldığında bir listedeki elamanın sıra numarasını elde etmeyi sağlar.

In [12]:
```
for j, i in enumerate(["Ankara", "İzmir", "Kayseri", "Samsun", "Tokat"]):
    print(j+1, i)
```

```
1 Ankara
2 İzmir
3 Kayseri
4 Samsun
5 Tokat
```

len

bu komutla liste, string, tuple (demet), dictionary (sözlük) vb. tipteki nesnelerin uzunluk veya elaman sayısı bilgileri elde edilebilir.

In [13]:
```
sehirler = ["Ankara", "İzmir", "Kayseri", "Samsun", "Tokat"]
len(sehirler)
```

Out[13]: 5

In [18]:
```
print(sehirler[0])
print(len(sehirler[0]))
```

```
Ankara
6
```

In [14]:
```
len("Mesut Karakoç")
```

Out[14]: 13

In [15]:
```
"Mesut Karakoç"
```

Out[15]: 'Mesut Karakoç'

In [0]:
```
```

Oyuncunun tuttuğu sayıyı bulabilen program

In [35]:
```
hak = 11
deneme = 0
enk = 1
enb = 100

print("[{}, {}] arasındaki tuttuğun herhangi bir tam sayıyı \
      {} defa da bulabilirim.".format(enk, enb, hak))

while deneme<hak:
    deneme += 1 # deneme = deneme + 1
    print("{}. deneme".format(deneme))
    tahmin = round((enk + enb)/2)
    print("Tuttuğun sayı {} mi?".format(tahmin))
    buldummu = input()
    if buldummu.upper() == 'E': # Evet buldun
        print("Aferin bana")
        break
    elif buldummu.upper() == 'B': # Tahmin ettiğim sayı daha büyük
        enk = tahmin
    elif buldummu.upper() == 'K': # Tahmin ettiğim sayı daha küçük
        enb = tahmin
    else:
        print('Lütfen anlamlı bir cevap ver? (E/B/K)')

if deneme == hak and buldummu.upper() != 'E':
    print("{} denemede başarısız oldum!".format(deneme))
```

```
[1, 100] arasındaki tuttuğun herhangi bir tam sayıyı      11 defa da bulabilirim.
1. deneme
Tuttuğun sayı 50 mi?
 k2. deneme
Tuttuğun sayı 26 mi?
 k3. deneme
Tuttuğun sayı 14 mi?
 k4. deneme
Tuttuğun sayı 8 mi?
 k5. deneme
Tuttuğun sayı 4 mi?
 k6. deneme
Tuttuğun sayı 2 mi?
 k7. deneme
Tuttuğun sayı 2 mi?
 k8. deneme
Tuttuğun sayı 2 mi?
 k9. deneme
Tuttuğun sayı 2 mi?
 k 10. deneme
Tuttuğun sayı 2 mi?
11. deneme
Tuttuğun sayı 2 mi?
 k11 denemede başarısız oldum!
```

Yukarıdaki uygulamada öğrendiğimiz yeni metod ve komutlar:

break

for veya while döngülerinden döngü bitmeden çıkmak için kullanılır.

upper (metodu)

aşağıdaki örnekteki gibi harflerin küçük harften büyük harfe dönüştürülmesini sağlar.

```
"mesut".upper()
MESUT
```

lower (metodu)

aşağıdaki örnekteki gibi harflerin büyük harften küçük harfe dönüştürülmesini sağlar.

```
"MESUT".lower()
mesut
```

help

Python ifadeleri hakkında bilgi edinmeyi sağlar.

(Python ifadesi)?

Help'e benzer şekilde Python ifadeleri veya çağrıldıysa
bir kütüphaneye ait ifade hakkında bilgi edinmeyi sağlar.

Bu kullanım Jupyter Notebook için geçerlidir.

```
In [26]: help('print')
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

```
In [0]: print?
```

Asal sayıları hesaplayan bir program

In [10]:
```
n=1
while n<100:
    n += 1
    asal = True
    m = 1
    while m<n-1:
        m += 1
        #print("m=",m)
        if n % m == 0:
            asal = False
            break
    if asal == True:
        print("n=", n)
```

```
n= 2
n= 3
n= 5
n= 7
n= 11
n= 13
n= 17
n= 19
n= 23
n= 29
n= 31
n= 37
n= 41
n= 43
n= 47
n= 53
n= 59
n= 61
n= 67
n= 71
n= 73
n= 79
n= 83
n= 89
n= 97
```

1000 tam sayısından önce kaç asal sayı var

In [24]:
```
n=1
t = 1
while n<1000:
    n += 2
    asal = True
    m = 1
    while m<n-2:
        m += 2
        #print("m=",m)
        if n % m == 0:
            asal = False
            break
    if asal == True:
        t += 1
        #print("n=", n)

print("{} sayısına kadar toplam {} tane asal sayı vardır.".format(n-1, t))
```

```
1000 sayısına kadar toplam 168 tane asal sayı vardır.
```

Sayı tahmin etme oyunu

In [17]:
```
hak = 7
rastgele = 35

print ("[1, 100] aralığındaki tuttuğum bir rastgele")
print ("tam sayıyı bulmak için {} hakkın var.".format(hak))

Bildinmi = False
i = 0
while i<hak and not Bildinmi:
    i += 1
    print("{}. hakkın".format(i))
    cevap = int(input('Tahminin nedir?'))
    if cevap == rastgele:
        print("{}. denemede buldun. Tebrikler!".format(i))
        Bildinmi = True
    elif cevap>rastgele:
        print("Tahminini küçült!")
    else:
        print("Tahminini büyült!")

if Bildinmi == False:
    print("Çok çalışman lazım!")
```

```
[1, 100] aralığındaki tuttuğum bir rastgele
tam sayıyı bulmak için 7 hakkın var.
1. hakkın
Tahminin nedir? 351. denemede buldun. Tebrikler!
```

Sayı tahmin etme oyunu (rastegele sayı eklendi)

In [28]:
```
from random import randint

hak = 7
rastgele = randint(1, 100)

print ("[1, 100] aralığındaki tuttuğum bir rastgele")
print ("tam sayıyı bulmak için {} hakkın var.".format(hak))

Bildinmi = False
i = 0
while i<hak and not Bildinmi:
    i += 1
    print("{}. hakkın".format(i))
    cevap = int(input('Tahminin nedir?'))
    if cevap == rastgele:
        print("{}. denemede buldun. Tebrikler!".format(i))
        Bildinmi = True
    elif cevap>rastgele:
        print("Tahminini küçült!")
    else:
        print("Tahminini büyült!")

if Bildinmi == False:
    print("Çok çalışman lazım!")
```

```
[1, 100] aralığındaki tuttuğum bir rastgele
tam sayıyı bulmak için 7 hakkın var.
1. hakkın
Tahminin nedir? 50Tahminini büyült!
2. hakkın
Tahminin nedir? 75Tahminini küçült!
3. hakkın
Tahminin nedir? 62Tahminini küçült!
4. hakkın
Tahminin nedir? 56Tahminini küçült!
5. hakkın
Tahminin nedir? 53Tahminini küçült!
6. hakkın
Tahminin nedir? 52Tahminini küçült!
7. hakkın
Tahminin nedir? 517. denemede buldun. Tebrikler!
```

In [0]: