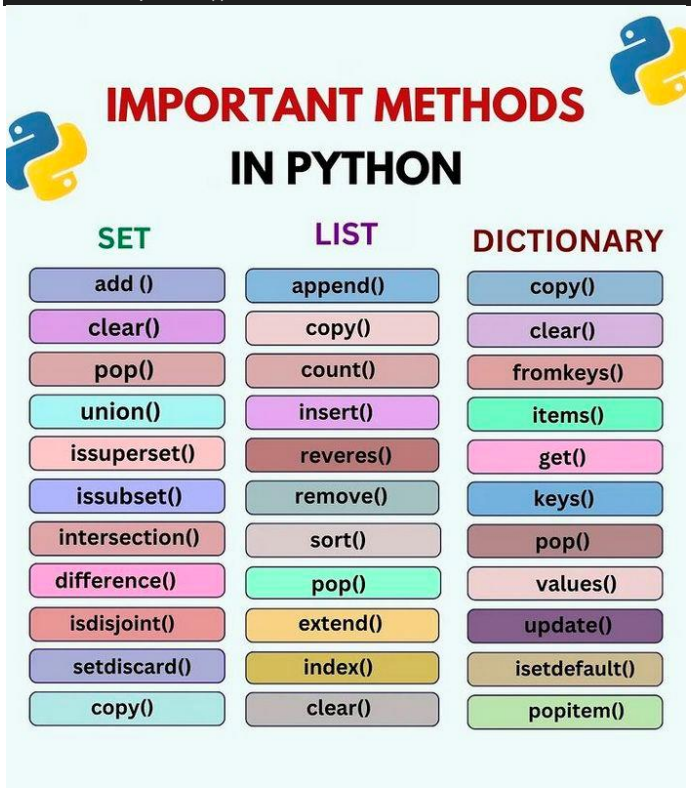
DS PYTHON NOTES (Feb 2024)

GIRIS

```
Python'da en çok kullanılan veri yapıları ve metodları şunlardır:
Listeler (Lists):
        append(): Listenin sonuna bir öğe ekler.
        extend(): Bir listenin sonuna başka bir listenin tüm öğelerini ekler.
        insert(): Belirtilen konuma bir öğe ekler.
        remove(): Belirtilen değere sahip ilk öğeyi listeden kaldırır.
        pop()   : Belirtilen indeksteki öğeyi kaldırır ve geri döndürür.
        index(): Belirtilen değerin indeksini döndürür.
        count(): Belirtilen değerin listede kaç kez bulunduğunu sayar.
        sort(): Listeyi sıralar.
        reverse(): Listeyi tersine çevirir.

Demetler (Tuples):
        index(): Belirtilen değerin indeksini döndürür.
        count(): Belirtilen değerin demette kaç kez bulunduğunu sayar.

Sözlükler (Dictionaries):
        keys(): Sözlükteki tüm anahtarları döndürür.
        values(): Sözlükteki tüm değerleri döndürür.
        items(): Sözlükteki tüm çiftleri (anahtar-değer) döndürür.
        get(): Belirtilen anahtara karşılık gelen değeri döndürür.
        update(): Bir sözlüğü başka bir sözlükle günceller.

Kümeler (Sets):
        add(): Bir öğeyi kümeye ekler.
        remove(): Belirtilen öğeyi kümeden kaldırır.
        discard(): Belirtilen öğeyi kümeden kaldırır (eğer öğe yoksa hata vermez).
        pop(): Bir öğeyi kümeden kaldırır ve döndürür.
        clear(): Kümedeki tüm öğeleri kaldırır.

Dize (Strings):
        split(): Diziyi belirli bir ayraç kullanarak parçalara böler.
        join(): Bir liste veya demetin öğelerini bir dizeye birleştirir.
        strip(): Dizenin başındaki ve sonundaki boşlukları kaldırır.
        upper(), lower(): Diziyi büyük veya küçük harfe dönüştürür.
        replace(): Belirli bir alt dizenin tüm örneklerini başka bir alt dize ile değiştirir.
```



Pep8 Kuralı

```
# bir satırda 79 karakterden fazlası olmasın
# her code için gereksiz yorum satiri kullanılmaz
# clean code
# okunabilirlik onemlidir…
```

print() fonksiyonu

```
print("Techpro education") # Techpro education

print("Mikail", "Serhan", "Mustafa")                    # Mikail Serhan Mustafa
print("Mikail", "Serhan", "Mustafa", sep="**")    # Mikail**Serhan**Mustafa
print("Mikail", "Serhan", "Mustafa", sep="-")     # Mikail-Serhan-Mustafa

help(print)

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.

print("Betül", end="!")
print("Zeynep", end= "*")
print("Mustafa", "Ali", sep="--", end= "%%")      # Betül!Zeynep*Mustafa--Ali%%
```

Escape Sequences: Code'un calismasini engelleyen durumlari onleme

```
print('I'm a python teacher')       # ERROR
print("I'm a python teacher")       # kesme kullanilcak ise dis tirnaklari uygun sekilde kullanilmali
print('I\'m a python teacher')      # I'm a python teacher

print("Bu ders çok uzadı sanki. \nBi an önce bitse mi acaba?")  # “\n”-> alt satira enter ypr
       # Bu ders çok uzadı sanki.
         Bi an önce bitse mi acaba?

print("Betül", "Ayşe", "Aytaç", sep = "\n")
       Betül
       Ayşe
       Aytaç

print("Betül", "Ayşe", "Aytaç", sep = "\t") # \t = bir tab tuşuna basar
       Betül   Ayşe    Aytaç

print("Bu ders çok uzadı sanki.\bBi an önce bitse mi acaba?") # \b = bir backspace tuşuna basmak anlamına gelir.
# Bu ders çok uzadı sankiBi an önce bitse mi acaba?

print("Betül", "Ayşe", "Aytaç", sep = "\b")
       BetüAyşAytaç

print(r"Bizde geri \nvites olmaz") # r→ tirnak icindeki escape’leri iptal eder islem yoksayar
       Bizde geri \nvites olmaz       # ve “ “ icindeki kullanilan \n islemide string ile bir gorur

print(f"deneme: {x}")   #-->() icine "f" string eklenerek {} ile degiskeni string olark yazlblr
                            # cucumber gherkin lang gibi
```

Input () Fonksiyonu

```
input() # input ile alınan TUM data'lar stringtir.

input("Lütfen yaşınızı girin")            # '29'

age = input("Lütfen yaşınızı girin")
age                  # '32'

2024 - age           # -> unsupported operand type(s) for -: 'int' and 'str'
age = int(age)
2024 - age           # -> 1992

age_1 = int(input("Lütfen yaşınızı girin"))       # almak istediğimiz Data Type içinde yazabilirz
type(age_1)          # int

ORN:
       yaş = input("Lütfen yaşınızı girin")
       yaş = int(yaş)
       doğum_yılı = 2024 - yaş
       print("Doğum yılınız:", doğum_yılı) # Doğum yılınız: 1992

ORN2: Kullanıcıdan kilo ve boy bilgisini alan ve girilen bilgilere göre vücut kitle    indeksini(vki) hesaplayan kodu yazın

       # girişlerin type'i acik ve anlasilir, işleme özel hale getirilmeli
       kilo = input("Lütfen kilonuzu kg cinsinden girin")
       boy = input("Lütfen boyunuzu cm cinsinden girin")
       vki = kilo / ((boy / 100) ** 2) # boy bilgisi m cinsinden olmalı

       print(vki)

ORN: #Kullanıcıdan Celcius bilgisini alın ve fahrenheit a çevirin.

       celc = float(input("Lütfen hava sıcaklığını celcius cinsinden giriniz. Örn: 25.5"))
       fahr = celc * 1.8 + 32
       print("Hava sıcaklığı", celc, " C derece", fahr, "fahrenheit eder")

       # Hava sıcaklığı 23.6  derece 74.48 fahrenheit eder
```

Round() Fonksiyonu

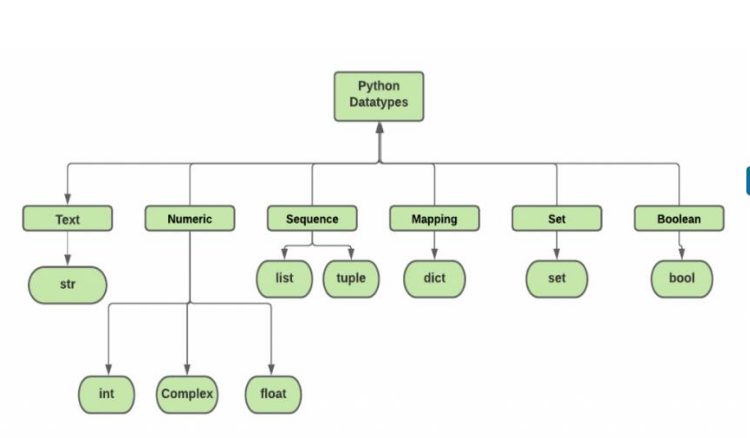
```
round(5.64)          # 6 → round() methodu yakin oldugu sayiya yuvarlar
int(5.65)            # 5 → integer datalarda ondalikli kismi zaten görmez
round(5.5)           # 6
round(4.5)           # 4 → round() tam ortadaki ondalikli degerleri cift olan sayiya yuvarlar
```

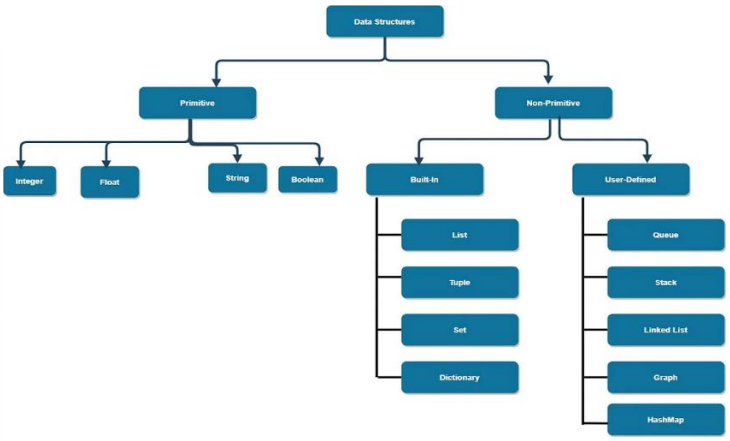
Variables:

```
course = "Techpro education"
print(course)

age_2 = 35
print(age_2)                # 35

name_7, name_8 = "Mustafa", "Ceren" # ayni sirada vermek sartiyla birden fazla variable tanimlanabilir
print(name_7, name_8) #"Mustafa", "Ceren"

age = 32, name = "sedat"     # → invalid syntax. Maybe you meant '==' or ':=' instead of '='?
age, name = 32 , "sedat"     # calisir
```

DATA TYPES

Data Structor:






```
type("hello world")   # str
type("35")              # str
```

```
age_4 = 25
type(age_4)    # int
type(25)       # int

type(2.4)      # float

pi = 3.14
type(pi)       # float → ondalikli sayilarsa “.” Nokta kullanilir

pi_number_2 = 3,14
type(pi_number_2)     # tuple !!!!!!!!--> virgül kullanildiginda dizilerden olan Tuple Type Data oluşur

type(False)
type(True)     # bool
```

Note: String’ler icin hangi tirnak turu ve adetinde kullanirsan ayni sekilde kapatmalisin

```
print('''üç tırnak ifadesi önemli bir yazım şeklidir''')
print("tırnak işaretleri bizim için önemli.")
print("""Hello world""")
print(""Hello world"")              # iki adet tirnakla calismaz→ 1 veya 3 adet ile calisiyor
print("""techpro"""'gül'"?"'!')       # → techprogül?!
```

Booleans:

True (1), False (0)
Var Yok

```
# Boolean karsiligi False olan ifadeler ---> 0, 0.0, "", [], (), {}, None, False

bool("sedat") # True → #dolu -var -true- 1
bool(0)       # False→ 0 = Yok
bool(1)       # True → var
bool("")             # False→ Bos
bool(" ")            # True → # space karakteri var 1 dir
bool("0")            # True → string icinde yazili -> var
```

Type Converison:

All types-→ Str

```
age = 32
str(age)              # '32' gecici olarak str cevirildi
type(age)             # int

age = str(age)        # variable olarak atadiginda bellekte kayit olur ve kalici olarak kaydedilir
type(age)             # str

str(3.14)             # '3.14'
str(True)             # 'True'

pi_number = 3.14
type(pi_number)       # float
str(pi_number)# '3.14'
```

Str to → int

```
age_1 = "33"
type(age_1)           # str

2024 - age_1          # int – str matematiksel islem yapmaz

age_1 = int(age_1) # 33
2024 - age_1          # 1991 → int'e çevirip kayit ettik islem yapti
```

Str to →float

```
float("3.14") # 3.14 → float için uygun bir str data'yi -> float'a cevirebiliriz

float("3.14a")# could not convert string to float: '3.14a'
```

int ← → float

```
int(5.4)      # 5     -→ float olarak verilip int'e cevirildiginde ondalikli sayinin "." noktadan sonrasini almaz

int(5.999)    # 5

int(22/7)     # 3
float(5)      # 5.0  → float data ise int datayi ondalikli sayiya çevirerek alir

int("3.14")   # ERROR  → hem float hem str gorunumuneki bir data int'e çevirilemez.

float("3.14")# 3.14  → float için uygun bir str data'yi -> float'a cevirebiliriz
```

bool → int

bool → float

```
int(True)     # 1
int(False)    # 0
float(True)   # 1.0
float(False)  # 0.0

str(False)    # 'False'
```

ARITHMETIC OPERATORS

```
3 + 5          # 8
5 -3           # 2
8 * 5          # 40
40 / 10# 4.0  → division(bölme) işleminde sonuç her zaman Float Data Typle ile gelir
2.4 * 5# 12.0
25 // 3        # 8     →   floor division, float (.) noktadan sonrasini kaybetmiş olur
25 %3          # 1     →  modulus -> kalani bulmak için kullanilir (25'in 3'e bol kalani verir)

2 ** 3         # 8 → 2 uzeri 3; cift yildiz işareti kullanilir ve Carpma, bölme gore de islem oceligi vardir

25 ** 0.5      # Karekok almak için ise ust ½ yani 0.5 olarak alinir
               # farkli köklere de alinir -> x=kok derecesi ise ust 1/x olarak alinir
25 ** (1 / 2)  # ustu () içine de alabilirsiniz more readable

x = (4 + 2) ** (4 ** (1 * 2 / 2) / 2)
print(x)    # 36.0

ORN: yarıçapı 7 olan bir dairenin alanını bulan python kodunu yazın. pi = 3.14

       r = 9
       pi = 3.14
       alan = pi * (r ** 2)

       print("yarıçapı 9 olan dairenin alanı:", alan)
```

COMPARISON OPERATORS (karsilastirma)

```
print(5 > 4)                  # Treu
type(5 < 6)                   # bool

6 >= 6                        # True
8 <= 7                        # False

8 == 9                        # False

True > 0.6                    # True ---> # True matematiksel degeri 1 dir
True > 1.6                    # False
False > 0.6                   # True  --> # False matematiksel degeri 0 dir

"Sedat" > "istanbul"  # False --> # S: 83 > i: 105
                              -> sadece ilk harflerin ascii degerini karsilastirir (case sensitive)

"sedat" > "istanbul"  # (s)115 > (i)105     # True

"feda" > "fener"              # d ile n arasında karşılaştırma yapar
                              -> ilk karakterler ayni ise ikinci karakterlerin ascii değerlerine gore

"feda" > 90                   # TypeError: '>' not supported between instances of 'str' and 'int'

ord("a")                      # 97  → ord(): fonksiyonu verilen ifadenin ascii degerini dondurur

chr(105)                      # 'i' → crd(): fonk ise tam tersi verilen int degerinin ascii ye gore
                                            string karsiligini getirir


ORN:
sifre = "1234"
kull_girişi = input("Şifrenizi girin")
kull_girişi                   # '1234'
sifre == kull_girişi  # True
```

LOGIC EXPRESSIONS

```
# bütün koşollu durumların True olmasını istiyorsak and kullanırız.
# koşullardan bir tanesinin dahi True olması yetiyorsa da or kullanıyoruz.

#işlem önceliği
#1 - not --> işlem sonucu her zaman True veya False tur.
#2 - and
#3 - or
```

1.not:

```
not False      = True
not True       = False
bool("Techpro")       # True
not "Techpro"  # False
not 0                 # True
not ""                # True
```

2.and:

```
# Hepsi True ise en sondaki elemanı döndürür.
  Bir tane bile False varsa ilk gördüğü False u döndürür

5 and 8                       # True
5 and 0 and 8         # 0       → ilk gordugu false değeri
5 and False and 0 and 8       # False  → ilk gordugu false değeri
"" and 0 and False            # ‘’      → ilk gordugu false değeri
5 and 7 and True              # True    → Hepsi True; en sondaki elemanı döndürür.
```

3.or:

```
# Hepsi False ise en sondakini döndürür.
Bir tane bile True varsa ilk gördüğü True’yu döndürür.

0 or False                    # False
0 or 0.0 or ""        # ‘’
0 or 15 or 20 or False        # 15
False or 5 and 4 and not False or True      # True
False or 5 and 0 and not True or 25         # 25

ORN:
5 >= 5                # True    → (5 > 5) or (5 == 5)
```

STRINGS

```
# immutable = degistirilemez
# iterable  = herbir karaktere ulasilabir ise

name_10 , name_11 = "yusuf"         # ValueError: too many values to unpack (expected 2)
name_10 = name_11 = "yusuf"         # 'yusuf'
name_10 , name_11 = "yusuf", "ahmet"       # 'yusuf', 'ahmet'
```

Indexing & Slicing

```
name = "Techpro Education"
name[0]# 'T' → 0.index
name[7]# ' ' → 7.index -> space karakteri
name[-1]      # 'n'   → en sondaki index'tir (-1)
name[-17]     # 'T' → (-) negatif indexte tersten baslar [-17]. İndexi getirir

"Betül"[3]    # 'ü'
```

Concatination:

```
"Techpro" + "Education"      # 'TechproEducation'
"techpro" - "pro"            # TypeError: unsupported operand type(s) for -: 'str' and 'str'
"Techpro" * 3                # 'TechproTechproTechpro'
"3" * 3                      # '333'

name = "Techpro Education"
name[4] + name[5] + name[6] # 'pro' → her bir indexteki elemani getirip birlestirdi
```

Slicing:

```
# index 0'dan baslar
# [x:y]       → x.index start point(dahil) : y.index stop point(hariç)
# [x:y:z]     → x.index start point(dahil) : y.index stop point(hariç) : z artis miktari
# [::-z]      → tersten baslart, z kadar atlayarak ilerler

name = "Techpro Education"
name[4:6]            # 'pr' → # stop noktasında son  karakter dahil değildir
"techpro"[4:10]      # 'pro'
"techpro"[8:12]      # '' → # bos küme, indexler yok
"techpro"[7]         # IndexError: string index out of range

"techpro"[1:5:2]     # 'eh' → 1.index dahil basla: 5.index haric : 2'ser atlayarak elemanlari getr
"techpro"[0:6:3]     # 'th'
"techpro"[:6:3]      # 'th'   → Baslangic girilmese de 0.indexten baslar; 3'er atlar
"techpro"[::3]       # 'tho'  → bastan sone 3'er atlayarak getirir
"techpro"[::]        # 'techpro' → hepsini getirir
"techpro"[::-2]      # 'opct'
"Techpro"[5:1:-1]    # 'rphc'
"techpro"[1:5:-1]    # ''    → 1.index start: 5.index stop : -1 tersten baslarsa ilk STOP pointi
                                Gorur ve bos string dondurur
```

String Methods

```
"techpro".upper()    # 'TECHPRO'
"TECHPRO".lower()    # 'techpro'

name = "Techpro"
```

```
name.swapcase()       # 'tECHPRO'    → her bir karakteri tersine cevirir

"emre"[2].upper()     # 'R'
"emre"[0:2] + "emre"[2].upper() + "emre"[-1]      # 'emRe'

capitalize() Tumce duzeni -ilk harfi buyuk digerlerini kucuk yapar
"techpro".capitalize()                            # 'Techpro'
"techpro eduCATion".capitalize()                  # 'Techpro education'

Title() her kelimenin ilk harfini büyük yapar
"techpro eduCATion".title()                       # 'Techpro Education'
```

Replace()

```
"tech?pro".replace("?","!")  # 'tech!pro'
"tech?pro".replace("?","")   # 'techpro'
"?serhan?murat?".replace("?","*")   # '*serhan*murat*'
"?serhan?murat?".replace("?","*",5).replace("a","ü").title().replace("*","!") # '!Serhün!Murüt!'


help(str.replace)
       Help on method_descriptor:

       replace(self, old, new, count=-1, /)
       Return a copy with all occurrences of substring old replaced by new.

      count
        Maximum number of occurrences to replace.
        -1 (the default value) means replace all occurrences.

       If the optional argument count is given, only the first count occurrences are
       replaced.
```

İs..() ile baslayan methodlar

```
help(str.isalpha)
       Help on method_descriptor:

       isalpha(self, /)
       Return True if the string is an alphabetic string, False otherwise.

       A string is alphabetic if all characters in the string are alphabetic and there
       is at least one character in the string.

"techpro99".isalpha()              # False
name = input("isiminizi girin")    # 'Sedat'
name.islower()                     # False
name.isalpha()                     # True


help(str.isalnum)
Help on method_descriptor:

isalnum(self, /)
    Return True if the string is an alpha-numeric string, False otherwise.

    A string is alpha-numeric if all characters in the string are alpha-numeric and
    there is at least one character in the string.
```

```
"Techpro".istitle()                 # True
"1234".isnumeric()                  # True
```

Count()

```
"techpro Education".count("e")      # 1
"techpro Education".count(",")      #False
"techpro Education".count("e","t")  # TypeError: slice indices must be integers or None or have an __index__ method

"techpro".replace("e","a").upper().count("O")     #sonuç int olduğu için str methodlara devam edilemez.
"techpro".replace("e","a").upper().isupper()      # True
```

len()

```
str_1 = "kimseyi arkada bırakmadık"
len(str_1)            # 25
str_1.count("")       # 26 → en sondaki “” karakterini de sayar

len(1453)             # integer datalar iterable degildir
```

split()

```
.split(“x”):  içinde belirtilen karakterden ayirma işlemi yapar

str_2 = 'Techpro is the best course in the world'
str_2.split() # ['Techpro', 'is', 'the', 'best', 'course', 'in', 'the', 'world']

str_2.split("o")      # ['Techpr', ' is the best c', 'urse in the w', 'rld']

type(str_2.split("ğ"))       # list

str_2.split().count("the")

str_2.split("o",maxsplit=2)  # ['Techpr', ' is the best c', 'urse in the world'] -> 2: en fazla 2 key uygular

help(str.split)

Help on method_descriptor:

split(self, /, sep=None, maxsplit=-1)
    Return a list of the substrings in the string, using sep as the separator string.

      sep
        The separator used to split the string.

        When set to None (the default value), will split on any whitespace
        character (including \\n \\r \\t \\f and spaces) and will discard
        empty strings from the result.
      maxsplit
        Maximum number of splits (starting from the left).
        -1 (the default value) means no limit.

    Note, str.split() is mainly useful for data that has been intentionally
    delimited.  With natural text that includes punctuation, consider using
    the regular expression module.
```

Strip()

```
Bastan ve sondan space karakterlerini siler-kirpar

"   Techpro           ".strip()           # 'Techpro'
"???Techpro???".strip("?")               # 'Techpro'
"???Techpro???".lstrip("?")              # left strip() sadece soldan siler
"???Techpro???".rstrip("?")              # right strip() sadece sagdan siler
"Techpro is the bessttsst".strip("t")    # verileni siler
"Techpro is the bessttsst".strip("ts")   # 'Techpro is the b'  →  t, s, ts, st ->
                                                                Tum kombinasyonlarini bulmaya calisir
```

İF - ELİF - ELSE STATEMENTS

İf:

```
if 10 < 3:
    print("10, 3 ten büyüktür")
    print(10 - 3)
    print(10 * 3)
    age_2 = 10 * 3 - 2
    print(age)
```

if-else:

```
if 5 > 4:
    print("if kısmı çalışır")
else:
    print("if calismazsa else kısmı çalışır")
```

if-elif:

```
x = 8
if x < 8:
    print("if çalışır")
elif x == 9:
    print("1 inci elif çalışır")
elif x > 10:
    print("2 nci elif çalışır")
elif x < 10:
    print("3 üncü elif çalışır")    # 3 üncü elif çalışır
```

if-elif-else:

```
x = float(input("lütfen 0 ile 50 arasında bir sayı girin"))     # 9 girildi

if x < 10:
    print("sayı 10 dan küçüktür")   # -> sayı 10 dan küçüktür
elif x < 35:
    print("sayı 35 ten küçüktür")
elif x <= 50:
    print("sayı 50 e eşit veya  küçüktür")
else:
    print("sayı 0 ile 50 arasında değil.")
```

```
ds = int(input("Son kahootta 15 soru vardı. siz kaç doğru yaptınız."))

if ds <= 5:
    print("dahi misin")
elif ds <= 10:
    print("şaka mısın sen")
elif ds <= 15:
    print("süpersin")
else:
    print("Girdiğiniz değer 0 ile 15 arasında değil.")
       #->  Girdiğiniz değer 0 ile 15 arasında değil.
```

LISTs

```
# mutable
# iterable
# iki farklı yöntemle liste oluşturabilirz;
       listName=[]
       list() -> fonksiyon

liste_1 = [1, 2, 3.14, "Mustafa", False]

len(liste_1)  # 5 -> eleman sayisi

len(1453)     # integer datalar iterable değildir -> TypeError: object of type 'int' has no len()

sayi=1453
strSayi= str(sayi)
list(str(sayi))
list(strSayi) #-->  ['1', '4', '5', '3']


liste_2 = list("Techpro")   #list() methodu iterable elemani liste haline getirir
liste_2       # ['T', 'e', 'c', 'h', 'p', 'r', 'o']

liste_4 = [2, 3.5, False, [1, 2, 3], (5,6,"Appa")] # list içinde her turlu data type'lari alir


list("mehmet")# ['m', 'e', 'h', 'm', 'e', 't']

liste_5 = [list("mehmet")]  -> [['m', 'e', 'h', 'm', 'e', 't']] #liste icinde liste oluşturuldu
len(liste_5)         # 1->  1 elemanli liste olarak olusturuldu

boş_liste_1 = []     #bos liste
boş_liste_2 = list() #list() fonksiyonu ile bos liste olusturulabilir

liste_6 = ["Fulya, Mustafa, Zeynep, Bilkay"]   # tek tirnak icinde yazilan her sey 1 adet elemandir
len(liste_6)         # 1 elemanli list

liste_7 = ["Fulya", "Mustafa", "Zeynep", "Bilkay"]
len(liste_7)         # 4 elemanli list

liste_9 = list(1545)
len(liste_9)         # TypeError: 'int' object is not iterable

liste_10 = list("Ali", "Murat") #list() fonksiyonu sadece 1 eleman alir ve listeye cevirir,
liste_10             # TypeError: list expected at most 1 argument, got 2

liste_10 = list("Ali")
liste_10             # ['A', 'l', 'i']

liste_10 = list("Ali" + "Murat")
liste_10             # ['A', 'l', 'i', 'M', 'u', 'r', 'a', 't']

ortaya_karışık = [3, 4.4, "Orkun", False, [1, 2, 3, 4, 5]]

ortaya_karışık[0]                          # 3
ortaya_karışık[1:4]                        # [4.4, 'Orkun', False]
ortaya_karışık[2]                          # 'Orkun'
ortaya_karışık[2].upper().count("R")       # 1
ortaya_karışık[4]                          # [1, 2, 3, 4, 5]
len(ortaya_karışık[4])                     # 5
ortaya_karışık[4][3]                       # 4

ortaya_karışık[1]                          # 4.4         -> 1.index elemani
str(ortaya_karışık[1])                     # '4.4'       -> 1.index elemani str cevirildi
str(ortaya_karışık[1])[1]                  # '.'         -> str iterable dir bunun da 1.indexi ulasildi
```

```
ORN: kullanıcıdan boy ve kilo sunu öğrenip vki durumunu yazdırın.

       kilo = float(input("Lutfen kilonuzu kg cinsinden giriniz"))                    # 48 kg
       boy = float(input("Lutfen boyunuzu cm cinsinden giriniz. orn: 160"))           # 160 cm
       BKI = kilo /(boy/100) ** 2
       print("kilo:", kilo,"\nboy:",boy, "\nBKI: ", round(BKI,2))

       if BKI < 18:
              print ("BKI Kategorisi: Zayif")
       elif BKI >= 18 and BKI < 25:
              print ("BKI Kategorisi: Normal")
       elif BKI >= 25 and BKI < 30:
              print("BKI Kategorisi: Obez")

       kilo: 48.0
       boy: 160.0
       BKI: 18.75
       BKI Kategorisi: Normal

ORN:
1 - Kullanıcıdan bir sayı alın.
2 - Bu sayı 3 e bölünebiliyorsa "Tech",
3 - 5 e bölünebiliyorsa "Pro",
4 - Hem 5 e hem de 3 e bölünebiliyorsa "Techpro" yazdırın.
5 - sayı bu şartların hiç birini sağlamıyorsa sayının kendisini yazdıran python kodunu yazın.

       num= int(input("Please enter a number"))

       if num % 3 ==0 & num % 5==0:        # -> en kapsamli olan en uste yazilir
              print ("TechPro")
       elif num % 3==0:
              print("Tech")
       elif num % 5==0:
              print("Pro")
       else:
              print ("The number you entered is " , num)

ORN:
x = input("Lütfen bir sayı girin")  # x input degeri str
y = int(x)                          # x'in int degerini y'ye atadik
print(type(x))                      # x'i hem str hem de int (y kalibinda) kullanabiliriz
print(type(y))
       #<class 'str'>
       <class 'int'>



potpori = ["Seyda", 3.14, True, 7, [1, 2, 3, 4, 5, 6]]
potpori[0] = "Betül"         # List'lerde elemanalri degistirebilir, silebilir veya ekleyebiliriz
                             #['Betül', 3.14, True, 7, [1, 2, 3, 4, 5, 6]]

NOTE:
#stringler ise immutable dir degisitirlemez yeni bir deger atanmadigi surece
name = "Techpro Education"
name[0] = "B"                # TypeError: 'str' object does not support item assignment
```

Basic Operations with Lists

Append():

List'in en sonuna sadece 1 adet eleman ekler

```
sayılar_1 = [1, 2, 3, 4, 5]
sayılar_1.append(6)          # [1, 2, 3, 4, 5, 6] -> tekrar calistirirsan tekrar ekler
sayılar_1.append(8,9)        # TypeError: list.append() takes exactly one argument (2 given)
```

insert():

verilen index'e elemani ekler (mevcut elemani silmez, onune eklemis olur)

list.insert(x, y) -> x index no, y ise eklenecek olan elemandir

```
sayılar_2 = [11, 12, 13, 14, 15]
sayılar_2.insert(1, 20)             # [11, 20, 12, 13, 14, 15] -> 1.index'e 20 eklendi
sayılar_2.insert(-1, 16)            # [10, 11, 20, 12, 13, 14, 16, 15] -> tersten 1.index'e 16 ekle
sayılar_2.insert(100, 17)           # [10, 11, 20, 12, 13, 14, 16, 15, 17, 17]
                                    # mevcut index sayisindan buyuk bir deger verilir ise insert()
                                      methodu ile de en sona eleman eklenebilir
                                    # () içine min 2 arguman kabul eder
```

remove():

remove(): icine yazilan elemani siler, eger eleman lsitede yok ise hata verir

```
sayılar_2 = [10, 11, 20, 12, 13, 14, 16, 15, 17]
sayılar_2.remove(17) # [10, 11, 20, 12, 13, 14, 16, 15] -> sadece siler gostermez
sayılar_2.remove(2)  # ValueError: list.remove(x): x not in list

isimler_1 = ["Fulya", "Serhat", "Aslan"]
isimler_1.remove("Serhat")   #['Fulya', 'Aslan']

isimler_1[1] = "Mehmet"             #1.index e "Mehmet" ismi geldi mevcut "Aslan" kaldirildi -uzerine yazar
                                    # ['Fulya', 'Mehmet']
```

Pop():

```
#listenin indexindeki elemanı siler. bişey yazmazsanız son elemanı siler
- ve sildigi elemani gosterir-dondurur

sayılar_1 = [1, 2, 3, 4, 5, 6, 7, 7, 7, 8, 8, 8, 9, 10, 12, 11]
sayılar_1.pop()      #11 ->  #icine eindex no yazmayinca Listenin en sonundaki indexini siler
sayılar_1.pop(6)     # 7  ->  #6.indexteki elamani sildi ve gösterdi

isimler_1     = ['Fulya', 'Mehmet']
isimler_1.pop(0)     # 'Fulya' → sildi

help(list.pop)

Help on method_descriptor:

pop(self, index=-1, /)
    Remove and return item at index (default last).

    Raises IndexError if list is empty or index is out of range.
```

Count():

```
sayılar_5 = [1, 2, 3, 2, 4, 3, 5, 1, 2, 4]
sayılar_5.count(3)          #2     ->  3 elemanindan 2 adet var
sayılar_5.count(10)         #0     -> 10 adinda bir elemani yoktur

karışık = [1, 0.0, False, 2, True, 1, 1.0, 0]
karışık.count(0)            # 3 adet -> False=0, 0.0=0, 0=0
karışık.count(1)            # 4 adet -> True=1, 1, 1.0=1
```

clear():

```
#fonksiyon olarak kullanilir, listeyi bosaltir, liste [] bos liste olarak ram bellete durur

sayılar_2.clear()           # []
```

del():

```
tamamen siler, Dikkatli kullanmalidir --> del name

del sayılar_3
sayılar_3                   # NameError: name 'sayılar_3' is not defined
```

sort ():

```
ayni data type larini siralar, buna dikkat edilmelidir
sayılar_6 = [4, 8, 3, 5, 15, 5.5 ,26, 41, 12]
sayılar_6.sort()                    # [3, 4, 5, 5.5, 8, 12, 15, 26, 41]
sayılar_6.sort(reverse = True)      #default olarak k'ten buyuge siralar, Reverse=true ile ters cevirir
                                    # [41, 26, 15, 12, 8, 5.5, 5, 4, 3]

isimler_2 = ["Murat", "ayşe", "seyda", "mehmet"]
isimler_2.sort()                    # ['Murat', 'ayşe', 'mehmet', 'seyda'] ->ascii degerlerine gore siralar

ord("M")                            # 88 → ascii degirini getirir

liste_1 = ["sedat", "mehmet", 1, 2.5]
liste_1.sort()                      #farki data type larini siralayamaz
                                    # TypeError: '<' not supported between instances of 'int' and 'str'

help(list.sort)
Help on method_descriptor:

sort(self, /, *, key=None, reverse=False)
    Sort the list in ascending order and return None.

    The sort is in-place (i.e. the list itself is modified) and stable (i.e. the
    order of two equal elements is maintained).

    If a key function is given, apply it once to each list item and sort them,
    ascending or descending, according to their function values.

    The reverse flag can be set to sort in descending order.
```

Extend():

```
#Iterable elemanlari eklemek icin kullanilir,
#append() methodu gibi fakat tek farki append sadece 1 eleman ekler, extend() 1 adet ITERABLE olan elemani ekler
```

```
#ONEMLI:
extent() methodu int data type’inda en az 2 elemanli iterable bir datayi ekler,
String’lerde ise karakterleri tek tek ekler iterabledir zaten

liste_2 =[1, 2, 3]
liste_2.extend(4)           # TypeError: 'int' object is not iterable

liste_2.extend([4, 5])      #[1, 2, 3, 4, 5]

liste_2.extend("Techpro")   #[1, 2, 3, 4, 5, 'T', 'e', 'c', 'h', 'p', 'r', 'o']
```

TUPLEs

```
#immutable / degistirilemez
#Iterable  / sayilabilir

#tuple() fonskiyonu icine sadece tek 1 argument alir ve aldigi string i listeye cevirir
#Tuple icine her telden data type'ta eleman alabilir (String, int, Boolean, list, tuple icinde tuple)

# list() lerden tek farki immutable olmasidir; List'ler mutable yani degistirilebilir-fonksiyoneldir
#Tuple lar List'lere gore bu nedenle daha hizlidr ve guvenlidir, saklanan datalarin direk olarak degistirilmesine izin vermez
#collection type lar icinde en guvenli olandir

tuple("Techpro")      # ('T', 'e', 'c', 'h', 'p', 'r', 'o')
                      # fonksion olarak kullanilabilir ve icine yazilan sadece 1 adet ITERABLE
                      elemani Tuple (liste gorunumlu) haline getirir

tuple("Duygu","Appa") #--> sadece 1 adet ITERABLE argument kabul eder
                            TypeError: tuple expected at most 1 argument, got 2

tuple_1= ("Techpro", 3, 3.8, False, [1,2,3])
                      # liste olarak olusturuldugunda elemanlr arasina "," ile ayrilir ve
                      1 den fazla eleman alabilir bu sekilde

tuple(12345)          # int iterable olmadigi icin list() fonsiyonu gibi hata verir
                      # TypeError: 'int' object is not iterable

name="Duygu"          # String bir degisken atayalim

tuple(name)           # ('D', 'u', 'y', 'g', 'u')
                       # degiskenin icindeki eleman Iterable ise Tuple a cevirir---> HEr bir
                         karakteri eleman haline getirir

tuple_2 =("Techpro",3, 3.14, False, [1,2,3]) # Tuple liste gorunumunde (,) ile assgin
                                          ettgimizde icine her turden istedigimiz kadar eleman ekleyebilriz

pi=3.14               # Ondalikli sayilarda "." kullanilir

pi2= 3,14             # virgül,dize halinde code yazken elemanlari ayirmak icin kullanilir
pi2  # (3, 14)       # ve burada iki eleman arasina virgul konulursa TUPLE olark algilar-->
                      ()[] vs paranteze gerek yok
                      # fakat code okunabilirligi icin parnatez kullanilmalidir.

["Techpro",3, 5, False]     # -> bu bir listedir
("Techpro")                 # -> sadece bir String dir- list veya tuple değildir
("Techpro",)                # -> Bu bir TUPLE dir, tek elemanli dahi olsa en az bir  ","  konuldugu
                                 icin Tuple olark algilar ve devami gelecek diye dusunur

(123,)                      # Tuple dir

tek_elemanli_tuple= ("Appa",)
len(tek_elemanli_tuple)     # 1 → 1 adet elemani vardir
type(tek_elemanli_tuple)    # tuple

tuple_olmayan = ("Techpro") # --->  "," kullanilmadi
type(tuple_olmayan)         # str ‘dir

tuple_olmayan2= (123.5)     # --->  "," kullanilmadi
type(tuple_olmayan2)        # float

type("type elemanidir",)    # str → type icine direk yazilan eleman type() fonksiyonuna aittir

type(("tuple eleman",))     # tuple -> tuple parantezi icinde tekrar belirtilir ise type ini tuple olarak verir
```

```
type(1,2)                    # TypeError: type() takes 1 or 3 arguments

list_1= 4,24,7,3,2,"name",6  # list'lerde mutlaka ()/[] kullanilmalidir, yoksa tuple olark algilar
list_1                       # (4, 24, 7, 3, 2, 'name', 6)

[1,2,3,5] + [5,8]            #iki ayri listeyi toplayip concat yapabilirz veya * 1 degerr ile carpabiriz vs
[1,2,3,5] * 3                # [1, 2, 3, 5, 1, 2, 3, 5, 1, 2, 3, 5]
(1,2,3) + (4,5)              #Tuple da da ayni concat+ / carpma* islemleri yapilabilir
(1,2,3) * 3                  # (1, 2, 3, 1, 2, 3, 1, 2, 3)

("Techpro",3,False) + ("Sedat",)    # ('Techpro', 3, False, 'Sedat')
```

Indexing & Slicing Tuple

```
tuple_3 = ("Aslan", 4, True, ["Tulay", 5,6], ("Zeynep", 2, 3.14))

tuple_3[3]           # ['Tulay', 5, 6] → listenin 3.indexindeki elemani bir listtir
tuple_3[0]           # 'Aslan'
tuple_3[4][2] # 3.14  →  4.indexin 2.indexi


tuple_3[1:4]         # (4, True, ['Tulay', 5, 6])
                     # Slicing: 1.indexten baslayip 4.index dahil tum elemanlari yazidirir
                     #Slice islemini hangi data type turunde yapiyorsak o type ile dondurur
                     #burada donen bir Tuple dir (List'lerde yaparsak list dondurur vs)

type(tuple_3[4][2]) # float -> indexle eleman sectigimizde gelen elemanin kendi data type i ile doner

tuple_3[3:0]         # 3.idenxt Start point <---  0.index Stop point ve
                     #---> code'lar soldan saga dogru calistirir, ilk 0.indexte stop pointi gorur ve
                        bos tuple dondurur

tuple_3[0] ="Meltem"         # TypeError: 'tuple' object does not support item assignment
                             #0.index'ine yeni bir eleman assign ettik ve hata verdi
                             -> Tuple'in kendisine direk mudahale edilemez

tuple_3[3].append(7)         #('Aslan', 4, True, ['Tulay', 5, 6, 7], ('Zeynep', 2, 3.14))
                             #tuple icindeki 3.indexteki list icerisine "7" elemani eklendi
                             # tuple kendi eleman sayisi degismedi, altkume olan list icindeki eleman
                               sayisi degisti
                             # alt kumeler dis islerde Tuple'a bagli/ ic islerinde bagimsiz gibi dusunebilirz
tuple_3[0].upper()    #'ASLAN'
                             # anlik olarak upper/lower case olur bellekte bir degisiklik olmaz
tuple_3[3].remove(5) # ('Aslan', 4, True, ['Tulay', 6, 7, 7], ('Zeynep', 2, 3.14))

del tuple_3[3]               # TypeError: 'tuple' object doesn't support item deletion
                             # 3.idenx elemn bir list olmasina ragmen listeyi tamen silmeye izin
                                vermez tuple objesi oldugu icin

del tuple_3          # tuple tamamini silebilirz fakat icerigi ile oynayamayiz
tuple_3              # NameError: name 'tuple_3' is not defined
```

Tuple Methods

Count():

```
tuple_4 = (1,3,4,6,2,5,8,9,4)
tuple_4.count(4)      # 2 -> iki adet 4 elemani vardir
tuple_4.count("Appa")        # 0 -> 'Appa' adinda bir elemani yok--> 0 matematiksel karsilik
tuple_4.index(4)      # 2 -> 4.idenxteki elemani 2 dir
tuple_4.index(11)            # ValueError: tuple.index(x): x not in tuple
```

range():

```
range() fonksiyonu;  icinde baslangic ve bitis sayilari verilir ve aradaki tum sayilari getirir
                              (start dahil, stop haric)
help(range)
        class range(object)
         |  range(stop) -> range object
         |  range(start, stop[, step]) -> range object
         |
         |  Return an object that produces a sequence of integers from start (inclusive)
         |  to stop (exclusive) by step.  range(i, j) produces i, i+1, i+2, ..., j-1.
         |  start defaults to 0, and stop is omitted!  range(4) produces 0, 1, 2, 3.
         |  These are exactly the valid indices for a list of 4 elements.
         |  When step is given, it specifies the increment (or decrement).

range(4,10)           # range(4, 10)
list(range(2,8))      # [2, 3, 4, 5, 6, 7]         → list icine koyunca aradaki sayilari da getirir
tuple(range(5,8))     # (5, 6, 7)           → tupe'a cevirmek istersek tuple icinde yazilir
range(3,9)[4] # 7  →  3 ile 9 haric arasindaki tum sayilari olusturdu ve 4.indeki getirdi
list(range(10))       # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
                      # baslangic default degeri 0 dir, bos birakilir ise otomatik olarak sifirdan
                        baslar, verilen degerden once de durur(10)

list(range()) # TypeError: range expected at least 1 argument, got 0
                      # en az 1 argument ister stop noktasi icin

list(range(2, 5.5)) # TypeError: 'float' object cannot be interpreted as an integer
                       # float/double sayi uretmez sadece integer

list(range(4, 12,  2))       # [4, 6, 8, 10] →  (start, stop_haric, artis_miktari)
list(range(-5,15, 3))        # [-5, -2, 1, 4, 7, 10, 13]-> -5 ile baslar, 3'er artirarak +15'ten once durur
list(range(10,2, -1))        # [10, 9, 8, 7, 6, 5, 4, 3] -> (start 10 / stop_before 2 / geri_sayim -1)
list(range(2,40, 2))         # [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
```

DICTIONARIES

```
# mutable
# iterable
# fonksiyon veya list seklinde kendi ozellikleri ile olusturulur;
# "key": "value" yapisindadir -->Json gibi yazimi
       {}
       dict()

dict_1 = {"key1":"value1", "key2":"value2"} -> # {'key1': 'value1', 'key2': 'value2'}

len(dict_1)          # 2
type(dict_1)         # dict
dict_1.items()# dict_items([('key1', 'value1'), ('key2', 'value2')])
                     # items(): methodu dictionary elemanlarini getirir gosterir

dict_1.keys() # dict_keys(['key1', 'key2']) → keys(): methodu sadece key leri getirir
dict_1.values()      # dict_values(['value1', 'value2']) -> values():sadece value leri getirir

dict_2= {"name":"omer", "age":30, "job":"Data Scienstist"}

dict_2.keys() # dict_keys(['name', 'age', 'job'])
dict_2.values()      # dict_values(['omer', 30, 'Data Scienstist'])
dict_2[0]            # KeyError: 0
                     #Dictionariler iterabledir fakat indexlenemez, Key veya Value ile bulunurlar

dict_2["name"]       # 'omer' -> key veya value degeri ile iterable dirlar

dic_3={"name": input("isminizi giriniz")}  ----> # {'name': 'duygu'}
```

“Keys” :

```
#key'ler string olmak zorunda değildir, int deger de alabilir
#key'ler unique olmalidir !!!!

dict_4={0:"zero", 1:"one", 2:"two"} -> # {0: 'zero', 1: 'one', 2: 'two'}
dict_4[0]            # 'zero'
                     # DIKKAT!! burada 0 index no degil, key olarak tanimladigimiz sifirdir-->
                       indexlemeler Key'lerle yapilir
dict_4.keys()              # dict_keys([0, 1, 2]) →   #tum key leri getirdi

dict_5= {False:0, True:1}                 # Boolean lar da key olarak atanabilir
dict_6= {2.14:"pi_number"}                # Float type lar da Dict Key olabilir
dict_8={(91,2,3): "Tuple_sayilar_listesi"} # Tuple lar Key olarak atanabilir

dict_7={[1,2,3]: "sayilar listesi"}       # TypeError: unhashable type: 'list'
                                #Sadece List'ler Dict’lerde Key olarak atanamazzzzzzzzz!!!!!!
```

: “Values”

```
# value attribute'lari her tum Data type olarak atanabilir
# birden fazla value da alabilir → Unique olmaz zorunda degildir

dict_9={"ogrenci1":["orkun", 32, "data scientist"], "ogrenci2":["duygu", 29, "python dev"]}
       # {'ogrenci1': ['orkun', 32, 'data_scientist'],
          'ogrenci2': ['duygu', 29, 'python_dev']}

dict_9.keys()         # dict_keys(['ogrenci1', 'ogrenci2'])
dict_9["ogrenci1"]           # ['orkun', 32, 'data_scientist']

dict_9["ogrenci1"][0]        # 'orkun'
                             #Value'lar bir list olarak tanimlandi ve List'lerin ozelliklerini kullanabiliriz,

#Value attribute'lari Tuple olarak da atanabilir; bu durumda tuple'da bir degisiklik yapilamaz
dict_10={"ogrenci1":("orkun", 32, "data scientist"), "ogrenci2":("duygu", 29, "python dev")}
       {'ogrenci1': ('orkun', 32, 'data_scientist'),
        'ogrenci2': ('duygu', 29, 'python_dev')}

#bir dict icinde farkli data typlarinda key=value kullanilabilir
DIKKAT -> sadece key'ler icin List kullanilamaz
dict_11= {"student1":('Sam',22,'data_sci'), 3.14:"pi_number", True:1}
       {'student1': ('Sam', 22, 'data_sci'), 3.14: 'pi_number', True: 1}
```

Dict() Fonksiyonu:

```
genelde ikili key=value yapisindaki HAZIR data'lari cevirmek icin dict() fonks kullanilir#

dict(one=1, two=2, three=3)  # {'one': 1, 'two': 2, 'three': 3}
                                    #ikili yapidaki Datalar Dictionary'e cevirir

dict(say1="one", sayi2="two") # {'say1': 'zero', 'sayi2': 'one'}

dict(0="zero", 1="one")      # SyntaxError: expression cannot contain assignment, perhaps you meant "=="?
                             # hazir Datadaki Key isimleri rakam veya sembol ile baslayamaz

list_1=[("name","duygu"), ("name2","Appa"), ("name3","Rich")]    → Map yapisinda bir List
dict(list_1)                 # {'name': 'duygu', 'name2': 'Appa', 'name3': 'Rich'}
```

List() Fonksiyonu:

```
dict_9
       {'ogrenci1': ['orkun', 32, 'data_scientist'],
        'ogrenci2': ['duygu', 29, 'python_dev']}

#bir dictioanry de list() fonks ile dict'yi listeye cevirebilirz--> sadece key'leri alir
list(dict_9)  # ['ogrenci1', 'ogrenci2']

#eger dict'yi items() olarak ele alip list()'e çevirirsek;
list olark tum ikili key,value larini List içine alir
list(dict_9.items())    # [('ogrenci1', ['orkun', 32, 'data_scientist']),
                           ('ogrenci2', ['duygu', 29, 'python_dev'])]
```

```
list_2= list(dict_9.items())  #seklinde yeni bir degiskene list() olark atarsak listlerin tum
                                ozelliklerini kullanabilirz

dict_friends ={"friend1":"Luna",   "friend2":"Mocha",  "friend3":"Arwen"}
             # {'friend1': 'Luna', 'friend2': 'Mocha', 'friend3': 'Arwen'}

len(dict_friends)                  # 3
dict_friends["friend1"]            # 'Luna'

dict_friends["friend4"] = "Maya"    # dict'lar mutable dir, eleman eklenebilir, silinebilir
             # {'friend1': 'Luna', 'friend2': 'Mocha', 'friend3': 'Arwen', 'friend4': 'Maya'}

dict_friends["friend2"] = "Pasa"
             # {'friend1': 'Luna', 'friend2': 'Pasa', 'friend3': 'Arwen', 'friend4': 'Maya'}
                           # mevcut key'in value degeri degistirilebilr, Key'ler unique dir, ayni
                             isimde key'e bir atama yaparsak mevcut olanin uzerine yazar

del dict_friends["friend2"]        # del ile () içinde belirtilen key silinebilir
dict_friends                       # {'friend3': 'Arwen', 'friend4': 'Maya'}


ORN:
# verilen sayilar isimli dict ye eleman ekleyiniz
sayilar ={"tek_sayilar": [], "cift_sayilar": []} -> # {'tek_sayilar': [], 'cift_sayilar': []}

1.yol:  # list[] olarak direk ekleme yapabilirz
sayilar["tek_sayilar"]= [1,3,5,7,9]
sayilar["cift_sayilar"]= [2,4,6,8,10]

--->   # {'tek_sayilar': [1, 3, 5, 7, 9], 'cift_sayilar': (2, 4, 6, 8, 10) }

2.yol:  # append() methodu ile tek tek ekleme yapabilirz (sadece tek eleman ekleneblir)

sayilar["tek_sayilar"].append(11)
sayilar["tek_sayilar"].append(13)
sayilar["tek_sayilar"].append(15)

---> # {'tek_sayilar': [1, 3, 5, 7, 9, 11, 11, 13, 15],
        'cift_sayilar': [2, 4, 6, 8, 10]}

3.yol: # extent() methodu ve range() ile aralik verip eklenebilir

sayilar["cift_sayilar"].extend(range(12,20,2))

---> # {'tek_sayilar': [1, 3, 5, 7, 9, 11, 11, 13, 15],
         'cift_sayilar': [2, 4, 6, 8, 10, 12, 14, 16, 18]}
```

Basic Operations with Dictionaries

Update():

```
dict_friends = {'friend3': 'Arwen', 'friend4': 'Maya'}

dict_friends.update({"friend5": "Melow"})
```

```
                    # {'friend3': 'Arwen', 'friend4': 'Maya', 'friend5': 'Melow'}

# append() methodu ile tek tek ekleme yapabilirz (sadece tek eleman ekleneblir)
# olusturlan bir dict'i diger bir dict icine update() ile ekleyebilirz
bf={"friend 7":"Susi", "friend 8":"Simos"}
dict_friends.update(bf)      -->
     { 'friend3': 'Arwen',
       'friend4': 'Maya',
       'friend5': 'Melow',
       'friend_7': 'Susi',
       'friend_8': 'Simos'}

#Concat veya carpma gibi islemler yapilamaz X
{"friend_9":"Maco"} + {"friend_10":"Korsan"} # TypeError: unsupported operand type(s) for +: 'dict' and 'dict'
```

Clear():

```
# dict items larini siler fakat Dict kendisini bellekten silmez, bos olark dondurur
dict_10
       {'ogrenci1': ('orkun', 32, 'data_scientist'),
        'ogrenci2': ('duygu', 29, 'python_dev')}

dict_10.clear()    # {}

NOTE:
emty_dict= {}         # bos dict olusturulabilir
type(emty_dict)       # dict
```

pop():

```
!!! Hatirlatma:
Pop() methods: normalde verilen INDEX'e gore silme yapar,
                 eleman yoksa hata vermez en sonrakini siler, sildigini dondurur

Dict'lerde ise KEY verilerek silme islemi yapilir (indexleme yapilamadigi için)



dict_11 = {'student1': ('Sam', 22, 'data_sci'), 3.14: 'pi_number', True: 1}

 dict_11.pop()        # TypeError: pop expected at least 1 argument, got 0
                            #icinde key verilmesi zorunludur

dict_11.pop(3.14)     # 'pi_number'
                            # verilen key ve value'yi siler ve silinen dataya ait value dondurur

dict_11               # {'student1': ('Sam', 22, 'data_sci'), True: 1}
```

NESTED DICTIONARIES

```
myfamily = {"child_1" : {"name" : "Emily", "year": 2004},
            "child_2" : {"name" : "Tobias", "year": 2007},
            "child_3" : {"name" : "Linus", "year": 2011}}

myfamily.items()      # dict_items ([('child_1', {'name': 'Emily', 'year': 2004}),
                                    ('child_2', {'name': 'Tobias', 'year': 2007}),
                                    ('child_3', {'name': 'Linus', 'year': 2011})])

myfamily.keys()       # dict_keys(['child_1', 'child_2', 'child_3'])

myfamily["child_1"]          # {'name': 'Emily', 'year': 2004}

myfamily["child_1"].items()  # dict_items([('name', 'Emily'), ('year', 2004)])

myfamily["child_1"]["name"]  # 'Emily'

#name1 değerine tuple formatta 2 adet value verildi;
myfamily = {"child_1" : {"name" : ("Emily","Rose"), "year": 2004},
            "child_2" : {"name" : "Tobias", "year": 2007},
            "child_3" : {"name" : "Linus", "year": 2011}}

myfamily["child_1"]["name"]  # ('Emily', 'Rose')
myfamily["child_1"]["name"][0]      # 'Emily'
```

SETS

```
# mutable- degistirilebilir
# iterable
# elemanlari unique tir !!! tekrarli eleman kabul etmez
# elemanlari rastgele siralar ---> Index lemeye izin vermez bu nedenle
# Set icinde; List, Set ve Dict'leri almaz ama tuple'lari alir
# set() fonksiyon ve {} liste gorunumlu parantez ile olsuturulabilir

set()
{}

help(set)
class set(object)
 |  set() -> new empty set object
 |  set(iterable) -> new set object
 |
 |  Build an unordered collection of unique elements.

# boş küme oluşturmanın tek yolu set() fonk dur
empty_set = set()            # set()
type(empty_set)              # set

empty_set2={}                # {} --> bu bir dict dir
type(empty_set2)             # dict

# 1 adet Iterable argument alir ve set'e cevirir
set("sedat", "techpro")      # TypeError: set expected at most 1 argument, got 2

set(12222345)                # TypeError: 'int' object is not iterable

set(str(12222345))           # {'1', '2', '3', '4', '5'}

# aldigi datadaki tekrarli elemanlari siler -ve- print() içinde rastgele siralar--> unordered
set("hello")                 # {'e', 'h', 'l', 'o'}

set("Techpro")               # {'T', 'c', 'e', 'h', 'o', 'p', 'r'}
print(set("Techpro"))        # {'c', 'r', 'T', 'h', 'o', 'e', 'p'}


#set olarak olusturuldu ve tekrarli ifadeleri sildi
{"blue", "green", "yellow", "blue", "black", "green"} --> # {'black', 'blue', 'green', 'yellow'}

# SET'ler içinde liste barındırmaz X
{"mikail", 25, True, 3.14, [1,2,3], (4,5,6)}            # TypeError: unhashable type: 'list'

# SET'ler içinde dict barındırmaz X
{"mikail", 25, True, 3.14, (4,5,6), {"name": "sedat"}}   # TypeError: unhashable type: 'dict'

# SET'ler içinde SET barındırmaz X
{"mikail", 25, True, 3.14, (4,5,6), {"name", "sedat"}}   # TypeError: unhashable type: 'set'
```

```
# SET'ler içlerinde tuple barındırabilir +
{"mikail", 25, True, 3.14, (4,5,6)}                    # {(4, 5, 6), 25, 3.14, True, 'mikail'}

liste_1 = ["kalem", "silgi", "kağıt", "kalem_ucu"]
liste_2 = ["silgi", "defter", "kağıt"]
liste_1+liste_2             # ['kalem', 'silgi', 'kağıt', 'kalem_ucu', 'silgi', 'defter', 'kağıt']

#unique degerlere ihtiyac oldugunda Set'leri kullanabilirz
set(liste_1+liste_2)        # {'defter', 'kalem', 'kalem_ucu', 'kağıt', 'silgi'}

#True=1, False=0 oldugu icin ilk gordugu elemani getirir, ikinci tekrari gormez
set_1 = {True, False, 1, 5, 0.0, 3.5}             # {False, True, 3.5, 5}

# set'ler elemnalarini rastgele siraladigi icin Indexleme yoktur
set_1[0]                                          # TypeError: 'set' object is not subscriptable
```

Unions (|), Intersections (&), Differences(-)

Birlesim "|", Kesisim "&", Fark "-"

```
a= set("suzan")
b= set("yavuz")
type(a)       # set
```

1. Union " | "

```
a                   # {'a', 'n', 's', 'u', 'z'}
b                   # {'a', 'u', 'v', 'y', 'z'}

a.union(b)          # {'a', 'n', 's', 'u', 'v', 'y', 'z'} --> a birlesim b
b.union(a)          # {'a', 'n', 's', 'u', 'v', 'y', 'z'}

a | b  == b | a     # {'a', 'n', 's', 'u', 'v', 'y', 'z'}  #--> a birlesim b
```

2. Intersection "&"

```
a                   # {'a', 'n', 's', 'u', 'z'}
b                   # {'a', 'u', 'v', 'y', 'z'}

a.intersection(b)   # {'a', 'u', 'z'} ---> a kesisim b
a & b  == b & a     # {'a', 'u', 'z'}    ---> a kesisim b
```

3. Difference "-"

```
a                   # {'a', 'n', 's', 'u', 'z'}
b                   # {'a', 'u', 'v', 'y', 'z'}

a.difference(b)     # {'n', 's'} --->  a'nin b'den farki --> yazim sirasi onemli
a-b                 # {'n', 's'}

b.difference(a)     # {'v', 'y'}
```

```
b-a                 # {'v', 'y'}

a-b | b-a           # {'n', 's', 'v', 'y'}
a ^ b               # {'n', 's', 'v', 'y'} --->  "^" symmmetric_difference isaretidir
```

Basic Operations with SETs

Clear():

```
set_3.clear()       # set()
```

remove():

```
a                   # {'a', 'n', 's', 'u', 'z'}
b                   # {'a', 'u', 'v', 'y', 'z'}

a.remove("s") # SET'ler mutable olmasi nedeniyle tek tek elemanlarda degisiklik yapilabilir
a.remove("e") # KeyError: 's'---->  olmayan elemani silerken hata verir
```

pop():

```
#pop() fonksiyonu ici bos olarak yazilirsa rastgele eleman siler set'ler de pek tercih edilmez
help(set.pop)
       Help on method_descriptor:

       pop(...)
              Remove and return an arbitrary set element.
              Raises KeyError if the set is empty.



a                   # {'a', 'n', 's', 'u', 'z'}
b                   # {'a', 'u', 'v', 'y', 'z'}

b.pop()       # 'y' ---->  rastgele "y" elemanini sildiğini gosterdi
```

add() – update():

```
set_3 = set("fıstık") # {'f', 'k', 's', 't', 'ı'}
set_4 = set("fındık") # {'d', 'f', 'k', 'n', 'ı'}

set_3.add("x")        # {'f', 'k', 's', 't', 'x', 'ı'}
set_3.update("techpro")      # {'c', 'e', 'f', 'h', 'k', 'o', 'p', 'r', 's', 't', 'x', 'ı'}
```

isSubSet(): --> altkumesi mi?

-> Boolean sonuc dondurur,
->siralama onemlidir
-> a altkumesi mi b'nin seklinde

```
set_5 = set("selin") # {'e', 'i', 'l', 'n', 's'}
```

```python
set_6 = set("enes")  # {'e', 'n', 's'}

# set_5, set_6 nın alt kümesi midir?  --> boolean sonuc dondurur
set_5.issubset(set_6)        # False

#6, 5'in altkumesidir
set_6.issubset(set_5) # True


# 7 ve 8 benzer ayni kumelerdir--> birbirinin altkumesi dir ler
set_7 = set("sine")  # {'e', 'i', 'n', 's'}
set_8 = set("enis")  # {'e', 'i', 'n', 's'}

set_7.issubset(set_8) == set_8.issubset(set_7)    == True
```

WHILE LOOP

```
while 10 > 4:
    print("bu döngü sonsuza kadar çalışır")

----------------------------------------
x = 1
while x < 10:
    print(x)
    x = x + 1

----> # 1,2,3,4,5,6,7,8,9

-----------------------------------------
x = 1
while x < 10:
    print(x)
x = x + 1       # artis yon işlemi döngü içinde olmadığı için sonsuza gider.

----------------------------------------
while True: # çalıştırma sonsuz döngü
    print("Bu kod her türlü çıktı verir. ")

----------------------------------------
x=1
while True:
    print("sonsuza kadar gider")
    x +=1

----------------------------------------
while False:
    print("sanki çıktı vermez")
----------------------------------------
while "":
    print("bu da çıktı vermez")
----------------------------------------

#calistirma
y=True
while y:
    print("While condition False olana kadar calisir")

y=False         #bu dongu sonsuza gider, dongu disina cikamadigi icin false göremez

-------------------------------------------------
x=1
while x<= 3:
    print(f"deneme: {x}")
    x += 1

print("Dongu bitti, (x !<= 3)")

       # deneme: 1
         deneme: 2
         deneme: 3
```

```
        Dongu bitti, (x !<= 3)

#NOTE: --> "f" string eklenerek {} ile degiskeni string olark yazabilirz
-------------------------------------------

x=1

while x <= 5:
    x**2
    print(f"x= {x}, x**2= {x**2}")
    x+=1

x= 1, x**2= 1
x= 2, x**2= 4
x= 3, x**2= 9
x= 4, x**2= 16
x= 5, x**2= 25

-----------------------------------------------------------
ORN:
y = True
aklımdaki_sayı = 55
while y:
    tahmin = int(input("1 ile 100 arasında bir sayı tahmin edin."))

    if aklımdaki_sayı < tahmin:
        print(f"Girdiğiniz sayı : {tahmin}. Tahminini azalt")
    elif aklımdaki_sayı > tahmin:
        print(f"Girdiğiniz sayı : {tahmin}. Tahminini arttır")
    else:
        print(f"Girdiğiniz sayı : {tahmin}. Doğru bildiniz. Tebrikler. ")
        y = False

#Girdiğiniz sayı : 55. Doğru bildiniz. Tebrikler.

-------------------------------------------------------------------
ORN: #listenin elemankarinin karesini bir liste icinde donduren while dongusunu yazin

num_list = [1,3,4,6,5,7]
square_list =[]     # bos list (sepet)
i=0                 # index 0 dan baslat

while i < len(num_list):                    # i < numList eleman sayisi(len) oldugu surece calistir
    square_list.append(num_list[i]**2)      # bos listeye numList'in tum indexlerini ekler
    i +=1                                   # i'yi artirarak donguyu devam ettir

print(square_list)                          # dongu disinda elde ettigimiz square listi yazdirdik
-> [1, 9, 16, 36, 25, 49]
-------------------------------------------------------------------------------

ORN:
num_list = [1,3,4,6,5,7]
square_list =[]
i=0
```

```python
while i < len(num_list):

    if num_list[i] % 2 == 0:
        square_list.append(num_list[i]**2)

    i+=1       # numList indexini artirarak donguyu devam ettirir, While icinde, if disindadir

print(square_list)

# [16, 36]

-------------------------------------------------------------------------
ODEV:  listesinin tek elemanlarının karesini,
       çift elemanlarının 2 katını liste halinde döndüren while döngüsünü yazın

list1 = [1,5,6,7,12,14,15,18,19,32,25]
resultList =[]
i=0

while i< len(list1):

    if list1[i] % 2 ==0:
        resultList.append(list1[i] * 2)
    else:
        resultList.append(list1[i] ** 2)
    i+=1

print(resultList)

# [1, 25, 12, 49, 24, 28, 225, 36, 361, 64, 625]
```

RANDOM MODULE

```
import random   # import ifadesi modulu/classi ice aktarmak icin kullanilir

random.random() #0 dahil, 1 haric araliginda sayilar uretir
# 0.556968573018895

random.seed(35)
random.random()  # 0.5486946056438222 → #belli bir sabite gore ayni degerleri rassal olarak uretilir

random.sample(range(100),5)  # [89, 35, 36, 25, 9]
                                  # 100'e kadar olan sayilardan rastgele 5 tanesini getirdi

random.randint(1,6)                # 3 → 1 ile 6 arasindan rastgele bir int sescti

random.sample(range(1,10),4) # [5, 6, 3, 7]
# range; 1 ile 10 arasinda sayilar uretir, Sample; bu sayilardan istenilen adet kadar(4) sayi secer

help(random.randint)
       Help on method randint in module random:
       randint(a, b) method of random.Random instance
       Return random integer in range [a, b], including both end points.

------------------------------------------------------------------------
y = True
aklımdaki_sayı = random.randint(1,100)
while y:
    tahmin = int(input("1 ile 100 arasında bir sayı tahmin edin."))→ input 52 girildi
    if aklımdaki_sayı < tahmin:
        print(f"Girdiğiniz sayı : {tahmin}. Tahminini azalt")
    elif aklımdaki_sayı > tahmin:
        print(f"Girdiğiniz sayı : {tahmin}. Tahminini arttır")
    else:
        print(f"Girdiğiniz sayı : {tahmin}. Doğru bildiniz. Tebrikler. ")
        y = False                          # tahmin dogru ise burada dongu sonlanir. Amaca ulasilmistir
# Girdiğiniz sayı : 52. Doğru bildiniz. Tebrikler.
------------------------------------------------------------------------
```

BREAK – CONTINUE

-> break; dongu icerisinde kullanildiginda donguyu sonlandirir, code calismasi sonlanir ve dongu disina cikilir.

-> continue ; dongu icinde yazildigi blogu atlar, code calismaya devam eder, sonraki satira gecer

```
ORN:
# sırayla int değerleri toplayın. str ifadeye denk gelince kod sonlansın

liste = [1, 2, 3, 4, "Techpro", 5, 6, 7]
x = 0
toplam = 0    # toplamada etkisiz eleman- bos sepet gibi dusunebiliriz, sonucu burada biriktirecek

while x < len(liste):

    if type(liste[x]) == int:
        toplam += liste[x]

    elif type(liste[x]) == str:
        break

    x += 1

print(toplam)

-----------------------------------------------------------------------

# sırayla int değerleri toplayın. str ifadeye denk gelince o ifedayi atlayın ve devam edin
liste = [1, 2, 3, 4, "Techpro", 5, 6, 7]
x = 0

toplam = 0

while x < len(liste):

    if type(liste[x]) == int:
        toplam += liste[x]
    elif type(liste[x]) == str:
        x += 1              # bulundugu index'e 1 ekle ve EN BASA DON!!!
        continue
    x += 1
print(toplam)               # en son sonucu yazdirir

-----------------------------------------------------------------------
```

MAX () – MIN()

```
list1= [1,3,54,56,7,8,456,86,21,-8]
min(list1)    # -8
max(list1)    # 456

tuple1= (1,2,3,4,5,6)
min(tuple1)   # 1
max(tuple1)   # 6

min(["techpro", "aylin","irem"])          # 'aylin'
min(["techpro", "aylin","irem"], 15)      # TypeError: '<' not supported between instances of 'int' and 'list'
```

```
-----------------------------------------------------------------------
#ODEV:
#listenin en küçük ve en büyük elemanını min ve max fpnksiyonu kullanmadan bulun. sort fonk da kullanmayın.

liste_1 = [4, 5, 6, 9, 85, 23, 65, 47, 32, 91]
# define the first element of the list as the min_num and max_num
min_num = liste_1[0]
max_num = liste_1[0]

i = 1
while i < len(liste_1):
    if liste_1[i] < min_num:   # index 1 ile baslayip 0.index ile karsilasitirir
        min_num = liste_1[i]   # min_num degiskenine yeniden deger ataması yapilir

    if liste_1[i] > max_num:
        max_num = liste_1[i]
    i += 1

print("min number:", min_num)
print("max number:", max_num)
# min number: 4
  max number: 91
```

FOR LOOP

```
# bir kosul belirtilmiyor ise ve Iterable Data'nin her bir elemani ile islem yapilacak ise
For-i-in Loop kullanilir

# for loop kalibi --->  for i(index) in IterableData:

for i in "iterableData": # i: verilen Iterable Data'nin 0.indexinden baslayarak elemanlarina ulasir
    print(i)

# i,t,e,r,a,b,l,e,D,a,t,a

-------------------------------------------------------------------------------------//
ORN:
# verilen listedeki tum elemanlarin karesini alan loop code'larini yaziniz
# for loop-----------------------------------

list1 = [1,2,3,4,5]

for i in list1:
    print(i**2)

# 1,4,9,16,25

# while loop---------------------------------

list1 = [1,2,3,4,5]          # 5 elemanli bir liste
x=0                          # while'da baslangic indexini(x) belirtmemiz gerekiyor

while x < len(list1):        # index no, liste eleman sayisindan kucuk oldugu surece calisir
       print(list1[x] ** 2)  # list1[x] : her dongude uzerinde oldugu index elemanidir
                               ** 2 ===> her elemanin karesini alir ve cikti verir

       x += 1                # While Loop'ta dongunun yonunu bildirmemiz gerekir--> index no ilerleyerek devam

# 1,4,9,16,25
-----------------------------------------------
"e" in "hello"    # True -> "hello" stringi icinde "e" var mi --> bool dondurur True
"a" in "hello"# False

-----------------------------------------------
for i in (1,3,5,7,9):
    print(f"{i} sayisinin karesi: {i ** 2}")

1 sayisinin karesi: 1
3 sayisinin karesi: 9
5 sayisinin karesi: 25
7 sayisinin karesi: 49
9 sayisinin karesi: 81
-----------------------------------------------.
for i in 13446:         # int iterable data degildir, elemanlarina ulasilamaz
    print(i)
# TypeError: 'int' object is not iterable
-------------------------------------------------------------------------------------------
isimler = ["emre", "Cihan", "mikail", "DUYGU"]
isimler.sort()

for i in isimler:
    print(i.title())
#
Cihan
Duygu
Emre
Mikail
-------------------------------------------------------------------------------------------
```

```
ORN: "Techpro" ile "T-e-c-h-p-r-o" ifadesini nasil elde edebiliriz

# 1.yol - eksik ------------------------
for i in "Techpro":
    print(i , end="-")
# T-e-c-h-p-r-o-      # en sonda (-) var

# 2.yol: --------------------------------
# bir degisken icerisinde toplayip r-trip() methodu ile en sondaki (-) symbolunden kurtulabilirz

empty_str = ""              # bos string sepeti

for i in "Techpro":
    empty_str += i + "-"    # String sepetine ekledigi elemanlar arasina "-" symbol ekler

print(empty_str.rstrip("-")) # rstrip(): rigth side "-" siler
# T-e-c-h-p-r-o
-----------------------------------------------.

dict1 = {"name": "dilek", "age":26, "job":"DataScientist"}

for i in dict1:             # for loop da dict ait sadece Key'leri gorebilir. List() ler gibi
    print(i)
#
name
age
job
----------------------------.
for i in dict1.values():    # dict'lerde kullanilan value() fonks ile value'lara ulasabilir
    print(i)
#
dilek
26
DataScientist
-------------------------------.
for i in dict1.items():             # items() ile cikti tuple formatinda gelir
    print(i)
#
('name', 'dilek')
('age', 26)
('job', 'DataScientist')
-------------------------------.
for i,j in dict1.items():           # dict(key:value) yapisinda; "i=key : j=value " temsil eder
    print(i, j, sep= " *** ")
#
name *** dilek
age *** 26
job *** DataScientist
-------------------------------.
for i,j in dict1.items():
    print(f"key: {i}, value: {j}")
#
key: name, value: dilek
key: age, value: 26
key: job, value: DataScientist
--------------------------------.
dict2 = {"name": "Fatma", "age":30, "job":"DataAnalysist"}

for i,j in dict2.items():
    print(f"key: {i}, value: {j}")
#
key: name, value: Fatma
key: age, value: 30
key: job, value: DataAnalysist
```

```
---------------------------------.
for i,j in dict2.items():
    print(f"key: {i:<5} = value: {j}")     # { :< } == metni SOLA yaslar, 5 karakter alan kaplar
#
key: name  = value: Fatma
key: age   = value: 30
key: job   = value: DataAnalysist
---------------------------------.
for i,j in dict2.items():
    print(f"key: {i:<5} = value: {j:>15}")  # { :> } sembolu ise SAGA yaslar verilen karakter alani icinde
#
key: name  = value:           Fatma
key: age   = value:              30
key: job   = value:   DataAnalysist
---------------------------------.
for i,j in dict2.items():
    print(f"key: {i:^15} = value: {j:^15}") # { :^ } sembolu verilen karakter alani icinde ORTALAR
#
key:      name       = value:      Fatma
key:       age       = value:       30
key:       job       = value:  DataAnalysist

-----------------------------------.
# string lerde metni yaslamak icin;
"techpro".center(20)           # 20 karakter icinde ortalar

"techpro".rjust(20)            # 20 karakter icinde SAGA yaslar

"techpro".ljust(20)            # 20 karakter icinde SOLA yaslar

-------------------------------------.
# For Loop ile sonsuz dongu nasil olur? Dikkat! yapmamak icin ogren)

list2 = ["sonsuz dongu"]     # 1 elemanli list

for i in list2:
    print(i)                # her bir elemani yazdirir
    list2.append(i)         # var olan listeye 5 elemani SUREKLI ekler --> infite

-----------------------
# eleman eklemek icin bos bir liste acip onun icine eleman eklemeli
list2 = ["sonsuz dongu degil"]  # 1 elemanli list

bosList=[]

for i in list2:
    print(i)                    # tek bir elemani var 1 kere yazdirir
    bosList.append(i)           # bagimsiz bos listeye i indexteki elemani ekler
print(bosList)
#
sonsuz dongu degil
['sonsuz dongu degil']
-----------------------------------------.

for i in range(1,51):       # 1 dahil- 51 haric ==> 50 adet dongu
    print(f"{i:<2} -ceza olarak 50 adet bu ciktiyi yazdir..")
                            # 1: satir no olarak tanimadik,
                            # 2 karakter icinde sola yasladik, Cikti sayisi gorunur oldu
#
1  -ceza olarak 50 adet bu ciktiyi yazdir..
2  -ceza olarak 50 adet bu ciktiyi yazdir..
3  -ceza olarak 50 adet bu ciktiyi yazdir..
       ...
```

```
-----------------------------------------------.
#ODEV 1
#Kullanicidan 1(dahil) 9(dahil) arasinda 1 sayi isteyin
#Bu sayinin carpim tablosunu olusturun

number = int(input("1 ile 9 dahil aralikta bir sayi giriniz"))

for i in range(1,10):
    print(f"{number} x {i} = {number*i}")
#
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
-----------------------------------------------.
#ODEV 2
#Kullanicidan 1(dahil) 9(dahil) arasinda 1 sayi isteyin
#Bu sayinin carpim tablosunu olusturun
# EGER kullanici gecersiz bir deger girdiyse tekrar dogru degeri girmesini isteyin

y = True
while y:
    number = int(input("1 ve 9 dahil bu aralikta bir sayi giriniz"))
    if 1 <= number <=9:
        for i in range(1,10):
            print(f"{number} x {i} = {number*i}")

        y=False

    else:
        print("gecersiz sayi girdiniz, lutfen tekrar deneyiniz")

#
gecersiz sayi girdiniz, lutfen tekrar deneyiniz
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
-------------------------------------------------------------------------------//


# 3.yol: sadece 3 deneme hakki vermek icin

count = 0

while True:
    number = int(input("1 ve 9 dahil bu aralikta bir sayi giriniz"))
    count += 1

    if count == 3:
        print("en fazla 3 kez hatali giris yapabilirsiniz, kartiniz bloke olmustur")
        break
```

```
    if 1 <= number <=9:
        for i in range(1,11):
            print(f"{number} x {i} = {number*i}")

        break
    else:
        print("gecersiz sayi girdiniz, lutfen tekrar deneyiniz")
#
gecersiz sayi girdiniz, lutfen tekrar deneyiniz
gecersiz sayi girdiniz, lutfen tekrar deneyiniz
en fazla 3 kez hatali giris yapabilirsiniz, kartiniz bloke olmustur
----------------------------------------------------------------------.

# orn: 1'den 50 dahil sayilar toplamini veren code yaziniz
# (n * (n+1) / 2) --> 1'den n'e kadar olan sayilarin top veriri

toplam =0
for i in range(1,51):
    toplam += i

print(toplam)

#1275
----------------------------------------------------------------------.
```

Sum() Fonksiyonu (toplama)

```
#-> sum() fonks verilen degerleri toplar # sum(0) fonks default degeri 0 sifirdir (toplamaya gore etkisiz eleman)
sum(range(1,51))                    # 1275

# Sum() default degerini 1000 yaptigimiz icin 1000 sayisinin uzerine toplama islemini yapar
sum(range(1,51), 1000)              # 2275
sum([True, 2,3,4,5],100)            # 115
```

Break – Continue with for loop

```
# 3 ve 3'un kati olan sayiya denk gelince code calismayi durdursun, diger durumlarda sayilarin karesini yazdirin
list3 = [1,2,4,3,5,7]

for i in list3:

    if i % 3 ==0:
        break
    else:
        print(i**2)
# 1,4,16
----------------------------------.

# 3 ve 3'un kati olan sayiya denk gelince code devam etsin fakat islem yapmadan atlasin, diger durumlarda
sayilarin karesini yazdirin
list3 = [1,2,4,3,5,7]

for i in list3:
```

```
    if i % 3 ==0:
        continue
    else:
        print(i**2)
# 1,4,16, 25, 49
----------------------------------.
```

Zip() Fonksiyonu

```
# zip() fonksnu Eleman sayisi en kucuk olana gore zipler
# Iterable datalarin icindeki index No ayni olan elemanlari Concat yapar ve ikili yapida bir list donderir

a = ["apple", "orange", "lemon"]    # 3 adet elemani var
b = [1,2,3]                         # 3 adet elemani var
zip(a,b)                                  # <zip at 0x259c21be600>

# list(fonks icinde yazildiginda elemanlari eslestirerek zipler)
list(zip(a,b))                      # [('apple', 1), ('orange', 2), ('lemon', 3)]

# verilen siralama ya gore ilk o listin elemanlarini yazdirir → ilk eleman=Key, ikinci=value olarak yazidirir
# 2'li dict items lari olusturur
# zip list eleman sayisi = eleman sayisi kucuk olan list'in eleman sayisina

# eleman sayisi kucuk olana listeye gore islem yapar
c = ["apple", "orange", "lemon","pear"]    # 4 adet elemani var
d = [1,2,3,5,6]                            # 5 adet elemani var --> 5.indexteki eleman bosta kalir
list(zip(d,c))#[(1, 'apple'), (2, 'orange'), (3, 'lemon'), (5, pear')] # # 4 adet elemani var

# dict'e cevirmek icin uygun datalar olusturur zip() fonk'u
dict(zip(c,d))       # {'apple': 1, 'orange': 2, 'lemon': 3, 'pear': 5}

# tekrarli elemanlarda en son gordugu elemana gore dict olusturur
c = ["apple", "orange", "apple"]    # 3 elemanli --> 2 elemani ayni
d = [1,2,3]                         # 3 elemanli

dict(zip(c,d))       # {'apple': 3, 'orange': 2}


# 3 adet degisken icin 3 adet index no degeri olusturulmalidir (i,j,k)
# sadece 2 adet index i ve j degiskenleri verildi

for i, j, in zip(name, age, job):
    print(f"name: {i:<6}, age: {j:<6}")
# ValueError: too many values to unpack (expected 2)

list(zip(x,y,z,w))      # w degiskeni int --> iterable degildir
# TypeError: 'int' object is not iterable
```

Enumerate() fonk

```
# iterable elemanlari olan bir datayi index 0 dan baslayarak(default) numaralandirir
# iterable cikti verir--> for dongusu icinde tum iterable datalar kullanilabilir
# sadece 1'er arttirir artis miktari degistirilemez
# baslangic no degistirilebilir--> default degeri yerine girdigimiz sayi ile baslatabiliriz

Name = ['betul', 'Mert', 'Can']

enumerate(name)      # <enumerate at 0x259c2740ef0>

list(enumerate(name)) # [(0, 'betul'), (1, 'Mert'), (2, 'Can')]
```

```
 # deafult deger yerine verdigimiz sayidan baslatabiliriz numaralandirmaya
list(enumerate(name, 100))   # [(100, 'betul'), (101, 'Mert'), (102, 'Can')]
```

```
# SORU: listenin min ve max degerleri arasindaki kayip sayilari bulun
number_list = [48, 10, 11, 21, 36, 5, 6, 52, 28, 29,
       53, 54, 45, 19, 20, 47, 55, 39, 41, 7,
       9, 17, 26, 27, 42, 22, 37, 51, 46, 18,
       44, 30, 34, 13, 15, 35, 33, 16, 50, 24]
-------------------------------------------------.
# çözüm - 1
kayıp_sayılar = []

for i in range(min(number_list),max(number_list)):

    if i not in number_list:
        kayıp_sayılar.append(i)

print(kayıp_sayılar)         # [8, 12, 14, 23, 25, 31, 32, 38, 40, 43, 49]
-------------------------------------------------.
# çözüm - 2
kayıp_sayılar = []

for i in range(min(number_list),max(number_list)):
    if i in number_list:
        continue
    kayıp_sayılar.append(i)
print(kayıp_sayılar)         # [8, 12, 14, 23, 25, 31, 32, 38, 40, 43, 49]
-------------------------------------------------.
#çözüm - 3
# range sayesinde min ve max değerler arasinda tum elemanlari unique olarak set olusuturulur

full_set = set(range(min(number_list),max(number_list)))
number_set = set(number_list)
full_set - number_set        # [8, 12, 14, 23, 25, 31, 32, 38, 40, 43, 49]
-------------------------------------------------.
```

LIST COMPREHENSION

```
# List comp sadece for loop yapilarinda kullanilir
# daha kisa code yazilmasini saglar
# code suresini de kisaltir
# buyuk datalarda calisirken kolaylik saglar
```

```
# ORN: listenin tum elemanarinin karesini veren code yaziniz

#1.yol: for loop
list1= [1,2,3,4,5]
emptyList =[]
for i in list1:
    emptyList.append(i ** 2)
print(emptyList)

# [1, 4, 9, 16, 25]
---------------------------------.
# 2.yol: List Comprehension
list1= [1,2,3,4,5]
[i ** 2 for i in list1]

# [1, 4, 9, 16, 25]
-----------------------------------.
ORN:
list2 = ["renault!", "polo*", "ford?"]
for i in list2:
    print(i.replace("!","").replace("*","").replace("?",""))

#
renault
polo
ford
--------------------
duzenlenmisListe = []
for i in list2:
    duzenlenmisListe.append(i.strip("!*?"))
print(duzenlenmisListe)

# ['renault', 'polo', 'ford']
--------------------
# for döngüsünü tek başına kullanıyorsak işlem kısmını for'un sol tarafına-basina yapmamız lazım
[i.strip("!*?") for i in list2]
```

For + if

```
sayılar_listesi = [1,15,2,36,5,89,45,62,52,71,30,56,45,95,61,48]
tek_sayılar = []
for i in sayılar_listesi:
    if i % 2:   # 2 ye tam bolunemeyen tek(odd) sayi demek -> i % 2 => True means odd / False means even number
        tek_sayılar.append(i)
print(tek_sayılar)
```

```
#[1, 15, 5, 89, 45, 71, 45, 95, 61]
-------------------------------------------------.
# for - if kullanilirken if yapisi for'un sagina / yapilacak islem ise soluna yazilir
[i for i in sayılar_listesi if i % 2 == 1]

#[1, 15, 5, 89, 45, 71, 45, 95, 61]

[ i for i in sayılar_listesi if i % 2 == 0 ]
# [2, 36, 62, 52, 30, 56, 48]
-------------------------------------------------.
```

Sort() fonk

```
# sort bir listede islem yapar ama cikti vermez, degiskene de atasak cikti vermez!!!!!!!!!!

a = [  i for i in sayılar_listesi if i % 2 == 0 ].sort()
a   # ---> cikti none yok

b = [  i for i in sayılar_listesi if i % 2 == 0 ]
b
# [2, 36, 62, 52, 30, 56, 48]

b.sort  # daha once degiskene atanmis bir variable ile sort yapilirsa cikti alinir
b       # ciktisi sort edilmis sekilde gelir
# [2, 36, 62, 52, 30, 56, 48]
```

For + if + else

```
# çift olanlar yerinde olduğu gibi kalsın, tek olanların karesini yazdırın
#1.yol: for loop;

sayılar_listesi = [1,15,2,36,5,89,45,62,52,71,30,56,45,95,61,48]
sonuc = []
for i in sayılar_listesi:
    if i % 2 == 0:
        sonuc.append(i)
    else:
        sonuc.append(i ** 2)
print(sonuc)
# [1, 225, 2, 36, 25, 7921, 2025, 62, 52, 5041, 30, 56, 2025, 9025, 3721, 48]

#2.yol: list comp;
# for-i en solda, diger islemler en basa if-else sirasiyla yazilir
# "i if i%2==0": eger i, 2'ye tam bolunuyor ise i'yi oldugu gibi kullan demek

[ i if i%2==0 else i**2 for i in sayılar_listesi]

# [1, 225, 2, 36, 25, 7921, 2025, 62, 52, 5041, 30, 56, 2025, 9025, 3721, 48]

-------------------------------------------------------------------------.

for i in (1,2,3):
    for j in [4,5,6]:
```

```
        print(i * j)    # i=1 icin; 1x4, 1x5, 1x6 seklinde tek tek ixj

# [4, 5, 6, 8, 10, 12, 12, 15, 18]
-------------------------------------------------------------------------
[ i * j for i in (1,2,3) for j in [4,5,6]]  # nested yapida da kullanilir
```

FUNCTIONS (Method olusturma)

```
KALIP;

def fonk_ismi (parameters):
    islem_cpde_satirir

def kareler_toplami(x,y):   # x,y degiskenleri ile olsuturulan method
    print(x ** 2 + y ** 2)  # bellekte method olusturulmustur, cikti vermek icin method call
                                    edilip degerler atanmalidir
kareler_toplami(3,4)        # method cagirma: parametrelere degerler atanir

kareler_toplami()           # parametreli method bos birakilamaz
              # TypeError: kareler_toplami() missing 2 required positional arguments: 'x' and 'y'



# method aciklamasi eklenecek ise (zorunlu degil) hemen method isminden sonra ilk satira yazilmalidir
def kareler_toplamı_2(x, y):
    """Bu fonksiyon iki tane sayının karesini alır ve toplar"""
    print(x ** 2 + y ** 2)

help(kareler_toplamı_2)
       Help on function kareler_toplamı_2 in module __main__:
       kareler_toplamı_2(x, y)
              Bu fonksiyon iki tane sayının karesini alır ve toplar

kareler_toplamı_2([1], (3))
# TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int'

# parametresiz method-------------------------------------------------------------------

def neset_baba():
    print("kadınlar insandır biz insanoğlu")

neset_baba()
# kadınlar insandır biz insanoğlu

-------------------------------------------------------------.
ORN:
def calculator(num1,opr,num2):

    if opr == "+":
        print(num1 + num2)
    elif opr == "-":
        print(num1 - num2)
    elif opr == "x":
        print(num1 * num2)
    elif opr == "/":
```

```
        print(num1 / num2)
    else:
        print("Hatalı operatör girişi.")

calculator(2,"x",4)                 # 8
calculator("tech", "+", "pro")      # techpro  # string lerde concat islemi yapar
calculator("tech", "-", "pro")      # FAKAT string lerde cikarma islemi yapilamaz
                                    # TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

```
# ODEV: Kullanıcıdan aldığınız bir sayının Armstrong number olup olmadığını kontrol eden algoritmayı oluşturun
# Armstrong Number: her bir rakaminin kuplerinin toplami kendisine esit sayilar
# 153 >>> (1 ** 3) + (5 ** 3) + (3 ** 3) == 153

x = input("lütfen bir sayı girin")
basamak_sayısı = len(x)
toplam = 0

for i in range(x):                  # x, input(string) bir datadir--> iterable
    toplam += int(i) ** basamak_sayısı
if toplam == int(x):
    print(f"Girdiginiz sayi: {x}. Bu sayı armstrong bir sayıdır.")
else:
    print(f"Girdiginiz sayi: {x}. Bu sayı armstrong değildir.")
```

```
ORN:
# Perfect number belirleyen bir algoritma oluşturun.
# Perfect number: pozitif tam bölenlerinin toplamı, kendisine eşit olan bir doğal sayıdır
# 6 >>> 1, 2, 3

def find_perfect_number(x):
    tam_bölenler = []

    for i in range(1, x):
        if x % i == 0:
            tam_bölenler.append(i)
    if sum(tam_bölenler) == x:
        print(f"girilen sayı:"{x} "perfect sayıdır")
    else:
        print(f"girilen sayı:" {x} "perfect sayı değildir")
```

print() & return() fonksiyonlari

```
# fonk sadece GOSTERSIN istersek print() fonk kullanilmali
# Fakat fonk farkli bir yerde tekrar kullanmak istiyorsak RETURN() fonk kullanilmalidir

# bir fonk gelecek sonucu tekrar kullanmak istiyorsak RETURN default fonk kullanilmali
# RETURN bir fonk icinde kullanilir, print() gibi bagimsiz kullanilamaz
# bir fonk icinde tek 1 tane RETURN kullanilir, dongulerde veya kosullarda tekrarlanabilir tabiki

print("hello world")         # hello world
a = print("hello world")             # hello world
print(a)                             # None
```

```
type(print())               # NoneType

def calculator_1(num1,opr,num2):

    if opr == "+":
        print(num1 + num2)
    elif opr == "-":
        print(num1 - num2)
    elif opr == "x":
        print(num1 * num2)
    elif opr == "/":
        print(num1 / num2)
    else:
        print("Hatalı operatör girişi.")

calculator_1(8,"x",5)       # 40
b= calculator_1(8,"x",5)            # 40
print(b)                            # None
# bir degeri tekrar kullanacksak print kullanlimamali, print onceki code'da birer kez
  calisirip icini bosaltir geriye "None" kaldi

def calculator_2(num1,opr,num2):
    if opr == "+":
        return num1 + num2
    elif opr == "-":
        return num1 - num2
    elif opr == "x":
        return num1 * num2
    elif opr == "/":
        return num1 / num2
    else:
        return "Hatalı operatör girişi."

calculator_2(5,"+",6) # 11
c= calculator_2(5,"+",6)
c                   # 11 → # degerin kendisini calgirmak default return sayesinde geri dondurur

ORN:

def my_len_1(x):
"""len fonksiyonunu beğenmediğim için kendi fonksiyonumu yazıyorum. Bu fonksiyon bir iterable ın uzunluğunu
verir"""
    sayaç = 0
    for i in x:
        sayaç += 1
    print(sayaç)

liste1 = [1,2,3,4,5]
my_len_1(liste1)     # 5
d = my_len_1(liste1)  # 5
print(d)                    # None
# print(), Variable olarak atanan fonk'dan gelen sonucu SADECE gosterir, baska bir islem yapmaz, none kalir
# bir fonk tekrar kullanilacak isse print ile olusturulmaz, bir islem yapmaz cunku

ORN:
```

```python
# iki kenar bilgisi bilinen dik üçgenin hipotenüsünü bulan fonk yazın. Daha sonra üçgenin çevresini hesaplayın
def hypo1(x,y):
    print((x**2 + y**2) **0.5 )

hypo1(3,4)            # 5.0
cevre = 3 + 4 + hypo1(3,4)  # Error
# hypo1(3,4) den gelen sonuc "NoneType" dir bu nedenle int type'lar ile isleme alinmadi, bir sonuc dondurmedigi icin
# icinde bulundugu fonk call edildiginde sadece anlik sonucu gosterir, isleme girmez


def hypo2(x, y):
    return (x ** 2 + y **2) ** 0.5

hypo2(3,4)                          # 5.0
cevre2 = 3 + 4 + hypo2(3,4)         #12.0
# hypo2(3,4) int olarak return ile geldigi icin kendisi bir ayni data typelar ile isleme girebilir, sonuc verir

ORN:
# ORN: girilen isim icerisindeki unlu harf sayisini bulunuz ve her bir unlu harf icin 1000 puan vererek puan kazandiran fonk kodunu yaziniz

#1. print ile sonucu yazdir
def adin_kadar_kazan(isminiz):
    """Bu fonksiyon ismin içindeki sesli harfleri bulur"""
    counter = 0
    for i in isminiz.lower():
        if i in "aeuio":
            counter +=1
    print(counter)

adin_kadar_kazan("duygu") * 1000    # print(counter) nedeniyle sonucu isleme almadi

#2. return ile sonucu dondur
def adin_kadar_kazan2(isminiz):
    """Bu fonksiyon ismin içindeki sesli harfleri bulur"""
    counter = 0
    for i in isminiz.lower():
        if i in "aeuio":
            counter +=1
    return counter

adin_kadar_kazan2("duygu") * 1000   # 2000
# bu method icinde ise return ile sonuc donduruldu, amac sonucu gormek degil, islem icinde sonuc kullanilacak ise sonucu getirmektir
```

```python
%%time          # calisma suresini tam zamanini verir

result = []
for i in range(1,10001):
    if find_armstrong_2(i) != None:
        result.append(i)
result
```

```
#
CPU times: total: 31.2 ms
Wall time: 23.4 ms
[1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407, 1634, 8208, 9474]

-------------------------------------.

%%time                # List comp daha hizlidir
[i for i in range(1,10001) if find_armstrong_2(i) != None]
#
CPU times: total: 0 ns
Wall time: 16.7 ms
[1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407, 1634, 8208, 9474]
```

Positional and keyword Arguments

```
# parametre siralamasina arguman verilirken dikkat edilmelidir

def my_name(first_name, last_name):
    print(f"My name is {first_name} {last_name}")

my_name("Sedat", "ASLAN")    # My name is Sedat ASLAN
my_name("ASLAN", "Sedat")    # My name is ASLAN Sedat

# parametreleri assign ederek verirsek hata olusmaz
my_name(last_name="ASLAN", first_name="Sedat")    #My name is Sedat ASLAN

# belirli bir parametreye defaul olarak dondurecegi bir deger assign edilebilir

def your_name(first_name, last_name = "Erdal"):
    print(f"Your name is {first_name} {last_name}")

your_name("Betül")    # Your name is Betül Erdal
your_name("Fulya", "Çelebi")   # Your name is Fulya Çelebi
 # default degere yeniden deger ataması da yapilabilir

# önce positional olanlar; yani herhangi bir deger atanmamis parametrelerin konumuna gore yazilmali, # en son
degeri atanabilir parametreler yazilabilir

your_name(last_name= "yıldız", "murat")    # Error

def your_name_2(first_name, last_name, second_name = "Ali"):
    print(f"Your name is {first_name} {second_name} {last_name}")

your_name_2("muhammet", "akil")     # Your name is muhammet Ali akil


def tek_sayılar(a, b, c, d, e):     # bes adet eleman alabililir
    tek_sayılar = []
    for i in a,b,c,d,e:
        if i % 2:
            tek_sayılar.append(i)
    return tek_sayılar
```

```
tek_sayılar(4,75,45,26,15)   # [75, 45, 15]
tek_sayılar(4,75,45)    # method parametre sayisi kadar arguman verilmelidir, ne eksik ne fazla
```

Arbitrary Number of Arguments(* args, **kwargs)

*args:

```
def tek_sayılar_2(* sayılar):       # (* parametre); parametre sayisi limitini ortadan kaldirir
    odds = []
    for i in sayılar:
        if i % 2:
            odds.append(i)
    return odds

tek_sayılar_2(3,64,35,99,43,11,22,3,2,12,2)       #[3, 35, 99, 43, 11, 3]
# istedigimiz kadar arguman verelibiliriz method parametresi olarak

# verilen sayilardan tek ise karesini; cift ise 2 katini alan algoritmayi yaziniz

def tek_çift(* sayılar):
    result = []
    for i in sayılar:
        if i % 2 == 0:
            result.append(i * 2)
        else:
            result.append(i ** 2)
    return result

tek_çift(3,56,4,2,6,4,22,98,5,44,32)       # [9, 112, 8, 4, 12, 8, 44, 196, 25, 88, 64]

def tek_cift_2(* sayilar):
    return [i*2 if i%2==0 else i**2 for i in sayilar]
tek_cift_2(3,4,53,2,134,2,5) # [9, 8, 2809, 4, 268, 4, 25]

tek_cift_2(range(1,10))             # Error
tek_cift_2(* range(1,10))    # [1, 4, 9, 8, 25, 12, 49, 16, 81]
# yildiz vermek zorundayiz ayni data type olustrumasi icin--> range ile sayilari uretir ve *args olarak bu yapiyi
algilar

print(* "duygu")                    # d u y g u  # *args yapisinda dondurur
```

**kwargs:

```
# **kwargs larin, *args lardan farki dict yapida olmasidir
dict_1 = {"name1": "seyda", "name2": "medine", "name3": "orkun"}

def kwargs_function(** kwargs):

    for i,j in kwargs.items():
        print(f"key: {i}, value: {j}")

kwargs_function(name4 = "Oguz", name5 = "Murat")
```

```
#
key: name4, value: Oguz
key: name5, value: Murat

kwargs_function(dict_1)             # error
 # dict oldugunu algilamasi icin ** kwargs yapisini belirtmemiz gerekiyor
kwargs_function(** dict_1)
#
key: name1, value: seyda
key: name2, value: medine
key: name3, value: orkun
```

FILTER()

```
#icerisinde verilen Iterable data'lardaki True ifadeleri dondurur, filtreler

#Filter() icerisine (None, Iterable-Data, veya Fonks) yazilir

help(filter)
Help on class filter in module builtins:

class filter(object)
 |  filter(function or None, iterable) --> filter object
 |  Return an iterator yielding those items of iterable for which function(item)
 |  is true. If function is None, return the items that are true.|
 |  Methods defined here:
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).


filter(None, "hello")
<filter at 0x2c9850c9ed0>

list(filter(None, "hello"))
#
['h', 'e', 'l', 'l', 'o']

# sadece True degerleri getirir, False degerleri eler
list(filter(None, [1,2,0,3.14, True, False, 0.0, "Techpro"]))
#[1, 2, 3.14, True, 'Techpro']

def find_odds(x):
    if x % 2:
        return True     # If x is odd (remainder; when divided by 2 is non-zero), return True
    else:
        return False    # If x is even (remainder when divided by 2 is zero), return False

find_odds(6)
#False

list(filter(find_odds, [1,2,3,4,5,6,7,8,9]))
```

```
#[1, 3, 5, 7, 9]


def find_evens(x):
    if x % 2:
        return False    # x % 2 != 0 ise False return eder
    else:
        return True
# filter fonk. True degerler ile calisir, bulmak istenilen ifadelerin True return etmesini saglamaliyiz


tuple(filter(find_evens, [0,1,2,3,4,5,6,7,8,9]))
#(0, 2, 4, 6, 8)


def len_5(x):
    if len(x) == 5: return True


isimler_listesi = ["Serhat", "sedat", "Metehan", "Bahar", "cennet", "Bilge" ]


# isimler lsitesindeki 5 elemanli(harfli) iterable degerleri getirir => fonksiyona gore true donduren ifadeler
list(filter(len_5, isimler_listesi))
#['sedat', 'Bahar', 'Bilge']



def find_armstrong_2(x):
    x = str(x)
    uzunluk = len(x)
    if int(x) == sum([int(i) ** uzunluk for i in x]):
        return int(x)

find_armstrong_2(40)


# 10 ile 100001 arasindaki sayilari ele alir ve find_armst_2 fonk gore true dondurenleri getirir
list(filter(find_armstrong_2, range(10,100001)))
#[153, 370, 371, 407, 1634, 8208, 9474, 54748, 92727, 93084]
```

LAMBDA()

```
def kareler_toplamı(x, y):
    return x **2 + y ** 2
kareler_toplamı(3,4)    #25


# kareler_toplami() gibi olusturulan fonks lar istenildigi zaman call edilir ve kullanilabilir
# Ayrica kullanilmasa bile bir kez calistirildiktan sonra del yapilmadigi surece bellekte yer kaplamaya devam eder

# Lambda() fonks ise kullan at mantiginda calisir, tek satirda en son deger ataması da yapilarak her calistirmaya
karsi bir kereligine calisir
```

```
# Lambda fonks syntax:
lambda x,y: x ** 2 + y **2
#<function __main__.<lambda>(x, y)>

(lambda x,y: x**2 + y**2)(3,4)      # 3 ve 4 degerleri x, y icin;

(lambda x: len(x) == 5)("sedat")           # x =>"sedat" stringinin length inin 5'e esit ise True dondurur
#True

(lambda x: len(x) == 5)("seda")
#False

list(filter((lambda x: len(x) == 5), isimler_listesi))
# ['sedat', 'Bahar', 'Bilge']


tuple(filter((lambda x: len(x) == 5), isimler_listesi))
#('sedat', 'Bahar', 'Bilge')

set(filter((lambda x: len(x) == 5), isimler_listesi))
#{'Bahar', 'Bilge', 'sedat'}


dict(filter((lambda x: len(x) == 5), isimler_listesi))  # dict ile calismadi
#ValueError: dictionary update sequence element #0 has length 5; 2 is required

# lambda ile icerisine aldigi string ifadeyi tersten yazdirsin
(lambda x: x[::-1])("safa")     # x= "safa" icin; x[start:stop:yon/Artis]
#'afas'

# polindrom: tersten ve duzden okunusu ayni olan ifadeler

(lambda x: x[::-1])("ey edip adanada pide ye")
#ey edip adanada pide ye'


# ifadenin tersten yazilisi kendisine isit ise Polindrom dur
(lambda x: x[::-1] == x)("ey edip adanada pide ye")
#True

(lambda x: x[::-1] == x)("appa")
 #True
```

MAP ()

```
# filter() gibi calisir fakat filter() fonks sonucu True donuyor ise Iterable datanin kendisini getirir

# map() fonks ise Iterable Datanin tum elemanlarini parantez icinde verilen Function'a tabi tutar ve bu fonks
sonucunu dondurur

# standart fonks
```

```python
def sayinin_karesi(x):
    return x**2
sayinin_karesi(5)

sayilar_listesi = [2,3,4,5,4,6,7,8]

map(sayinin_karesi, sayilar_listesi)    # tek basina funct binary bilgisini cikti verir
#<map at 0x26790e865c0>

# sayilar listesini fonks a tabi tuttu ve sonucu cikti verdi
# list veya farkli bir dizi data type'inda okunur sekilde cikti verir
list(map(sayinin_karesi, sayilar_listesi))
#[4, 9, 16, 25, 16, 36, 49, 64]


# iterable in tum datalari var=True=1 seklinde gordugu icin datanin kendisini dondurdu isleme sokmadi
list(filter(sayinin_karesi, sayilar_listesi))
# [2, 3, 4, 5, 4, 6, 7, 8]


#map() fonksiyonu, bir fonksiyonu verilen bir koleksiyonun her elemanına uygular ve sonuçları yeni bir iterable nesne olarak döndürür.
#filter() fonksiyonu, belirli bir koşulu sağlayan elemanları seçmek için kullanılır. Yani, bir koleksiyonu filtrelemek için kullanılır.
#map() fonksiyonu her elemana aynı işlemi uygularken, filter() fonksiyonu belirli bir koşulu sağlayan elemanları seçer.



list(map(len, ["techpro", "yasemin", "murat","ali"]))   # fonks dan gelen sonucu dondurdu
#[7, 7, 5, 3]


# standart function
def tek_cift(* sayi):
    result= []
    for i in sayi:
        if i %2 ==1:
            result.append(i**2)
        else:
            result.append(i)
    return result

tek_cift()
#[]

tek_cift(1,2,3,4,5,5,6,6)
#[1, 2, 9, 4, 25, 25, 6, 6]


# lambda ile tek degerli ayni fonks yazilisi

(lambda x: x**2 if x%2==1 else x)(8)
#8
```

```
# lambda ile *args degerli ayni fonks yazilisi X
# lambda'da *args fonk kullanilamaz, sinirsiz deger verildiginde tuple olarak algilar islem yapmaz
(lambda *x: x**2 if x%2==1 else x)(8,9,2,4,5)
#TypeError: unsupported operand type(s) for %: 'tuple' and 'int'


map((lambda x: x**2 if x%2==1 else x), [8,9,2,4,5])
#<map at 0x26790e8d480>


# lambda fonks'unu ile iterable datanin 1'den fazla elemanini isleme almak icin Map() icerisinde kullanabiliriz
# map() fonk icinde lambda() fonks ve iterable data olarak tum elemanlari fonk islemine sokar

list(map((lambda x: x**2 if x%2==1 else x), [8,9,2,4,5]))
#[8, 81, 2, 4, 25]


list(map(tek_cift, [1,2,3,4,5]))    # fonk map icerisinde iterable data ile calsitirildiginda
#[[1], [2], [9], [4], [25]]


tek_cift(5) # fonk tek basina calistirildiginda
#[25]
```

MODULES (Classes)

```
# pandas librarly import etmek icin ==> import pandas as pd yazilir ve calistirilir
# hata veriyor ise install edilmelidir ilk once; indstall etme icin farkli syntax'ler; birbiri yerine
kullanilabilir
#   pip install pandas
#   !pip install pandas
#   conda install pandas


import pandas as pd
#import pandas as pd

pip install pandas  # zaten var oldugunu dondurdu
#[notice] A new release of pip is available: 23.1.2 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip


# lambda real Data analizinde nasil kullanilir

#1. data analiz icin pandas library import edilir ---> import pandas as pd
#2. analiz edilecek data dosyasi okunmasi icin gerekli code syntax yazilir ve dosya tanimlanir ---> df =
pd.read_csv("adult_eda.csv") ==> csv: virgullerle ayrilmis datalardan olusan dosya uzantisidir, excel gibi gorunur
fakat sutunlar inactive dir virgullerle datalari siralar
#3. dosyadaki data sutunlari incelenmesi icin dosya degiskeni ile birlikte call edilir ---> df.sex
```

```
#4. duzenlenecek veya degisitirilecek datalar tanimlanir - lambda fonk ile bellekte yer kaplamadan bir kerede datalar temizlenebilir
    # --->

# pandas bir Data Analysis Library dir --> pd seklinde kullanilabilir farkli bir isimde verilebilir
import pandas as pd


# csv(comma seperated values) uzantili dosyayi okumasi icin bu dosyayi calisma alanindaki klasor icine almaliyiz gormesi icin
df = pd.read_csv("adult_eda.csv")
```

age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain

32561 rows × 15 columns---→ cvs dosya uzantili bir veri seti yuklendi

```
# Female ve Male degiskenleri birden fazla code ile tanimlanmis sadece 0 ve 1 seklinde tanimlamak icin;
df.sex
#
0          Male
1          Male
2          Male
3          Male
4        Female
          ...
32556    Female
32557      Male
32558    Female
32559      Male
32560    Female
Name: sex, Length: 32561, dtype: object


# if x=="Male" ise x:0 degerini ver----> diger durumlarda x:1 degerini ver
df.sex.apply(lambda x:0 if x == "Male" else 1)
#
0        0
1        0
2        0
3        0
4        1
        ..
32556    1
32557    0
32558    1
32559    0
32560    1
Name: sex, Length: 32561, dtype: int64

(lambda x: x**2 if x%2 else x*2)(20)    # lambda code dizilimi
```

Random Modules (class)

```
import random as new_name
random.random   # random class i artik yeni ismle call edilmeli as new_name seklinde isimlendirebiliriz

new_name.random
#<function Random.random()>

import random
random.random() # random() fonk olan 0 ile 1 arasinda rastgele degerler dondurur
#0.16474852013111907
```

Math Modules ()

```
import math

# factorial() -> 1 den kendisine kadar olan sayilarin carpimi 5!
math.factorial(5)
#120

()   # pi fonks listesine baktigimizda sembolu de farklidir bu bir degiskendir fonk degildir; () ile fonk gibi
yazilirsa hata alinir
math.pi
#TypeError: 'float' object is not callable

math.pi     # bellekteki tanimli variable degerini getirir
#3.141592653589793

sayinin_karesi(3)
#9


# 3'un 2.kuvveti, karesi demek # iki farkli degisken var birinci deger ikinci degeri ust kuvvet olarak gorur
math.pow(3,2)
#9.0


64 ** 0.5
#8.0


math.sqrt(64) # square=karesi
#8.0
```

pyautogui module;

```
# python da herhangi bir kutuphaneye ulasamadigimizda bunu kullanabiliriz

pip install pyautogui
#
[notice] A new release of pip is available: 23.1.2 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```python
import pyautogui as pg  # mause ve klavye hareketlerini automatize eder

pg.position()   # mouse un anlik bulundugu koordinatlari verdi
# Point(x=124, y=159)


pg.moveTo(x=124, y=159) # verilen koordinatlara mouse goturur
pg.leftClick()
pg.write("buraya biseyler yaz bu code calistimak icin sift+enter")


# import etmek istedigimiz dosya disarda ise dosya yolunu verip import edebilirz
from module_01 import my_math


import py_modules
#User/DuyguJones/AppData/Local/Programs/Python/Python311/BURAYA CALISMA DOSYANIZI EKLEYIN
```