Lise Python Tekrar Notları

1) Python Veri Tipleri
   a) Tamsayı(Integer): Pozitif ve negatif tüm sayıları kapsar (1, 100, -1000, 400 ...)
   b) Ondalıklı Sayı(Float): Ondalıklı sayıları kapsar. (3.14, 5.22 ...)
   c) Karakter Dizileri(String): Gerçek hayatta kullanığımız yazıların aynısıdır.

   NOT: Pythonda değişkenler kendilerine verilen değişkenin tipinin otamatik olarak almaktadır.

2) Tip Dönüşümleri:
   a) Tam Sayıya Çevirme: Tam sayıya çevirmek için int() fonksiyonunu kullanıyoruz.
      - print(int("55")) #Çıktı:55
      - print(int("06")) #Çıktı:6
      - print(int(5.5)) #Çıktı:5 ondalıklı sayıyı tam sayıya çevirme.
      - print(int("test")) #Çıktı:hata
   b) Tamsayıdan Ondalıklı Sayıya Çevirme:
      - Print(float(5)) #Çıktı:5.0
   c) Sayıları String'e Çevirme:
      - Print(str(1923)) #Çıktı:'1923'

3) Yorum Satırları
   a) Tek Satırlık Yorum Satırı: Yazdığımız yazının önüne "#" karakteri koyarak kullanılır.
      ÖR: #Tek satır yorum satırım
   b) Çok Satdırlı Yorum Satırı:
      """
      Yorum satırı 1
      Yorum Satırı 2
      """

4) Matematik Operatörleri
   a) Dört İşlem Operatörleri (+, -, *, /):
   b) Tamsayı Bölme (//):
      Ör: print(17//5) #Çıktı:3
   c) Üs Bulma (**):
      Print(2**3) #Çıktı:8
   d) Kalan(Mod) Bulma(%):
      print(17%5) #Çıktı:2
   e) Operatörleri Birlikte Kullanırken İşlem Önceliği
      - Parantez içi henzaman önce yapılır.
      - Çarpma ve bölme işlemleri toplama ve çıkarma işleminden önce yapılır.
      - İşlem öncelikleri eşitse işlemler soldan sağa doğru yapılır.
        ÖR: 5 + 5*7/(8-3) #Çıktı: 12.0

5) String Oluşturma:
   a) Tek Tırnak Ile : 'Burası Btk Burası Zirve'
   b) Çift Tırnak İle : "Burası Btk Burası Zirve"
   c) Üç Tırnak İle : Birden fazla satırdan oluşan bir metin yazdırılmak istendiğinde kullanılır.

```
print("""
Burası Btk Burası Zirve
Btk Python Kursu
""")
```

6) String Birleştirme: İki string ifadeyi birleştirmek için “+” operatörünü kullanıyoruz.
   ÖR: print("Btk" + " " + "Akademi") # Btk Akademi
7) Bir string ifadenin karekter sayısını bulma: Bu işlem için len fonksiyonu kullanılıyor.
   ÖR: print(len("Hey Gidi Dünya Hey")) # 18
8) String ifadeler üzerinde çarpma işlemi:
   ÖR: print("Btk"*4) #Çıktı:BtkBtkBtkBtk
9) String Parçalama ve İndexleme
   a) Berlirli Bir String’in Karakterlerine Ulaşma: Stringler bir karakter dizisidir ve her karakterin bir index numarası vardır. Bu index numarası 0 dan başlamaktadır.

      Örneğin kurum adında bir değişken oluşturup içerisine “BTK” atayalım.

```
print(kurum[0]) #Çıktı:B
print(kurum[1]) #Çıktı:T
print(kurum[2]) #Çıktı:K
print(kurum[-1]) #Çıktı:K
print(kurum[-2]) #Çıktı:T
print(kurum[-3]) #Çıktı:B
```

   b) Belirli Bir Stringin Belirli Bir Kısmına Ulaşma

      degişken_adı[başlangıç index değeri : bitiş index değeri : atlama değeri]

      NOT: Bu tanımlamada başlangıç index’inden başlar ve bitiş index’ine kadar alır yanı bitiş indexsindeki karekteri almaz.

      Örnekler:

```
site ="Seçmek özgürlüktür. güvenlinet.org"
print(site[0:7])   #Çıktı:Seçmek
print(site[20:])   #Çıktı:güvenlinet.org
print(site[::-1])   #Çıktı:gro.tenilnevüg .rütkülrügzö kemçeS
print(site[7:18])  #Çıktı:özgürlüktür
```

10) Print Fonksiyonu Ve Formatlama: Print fonksiyonu ekrana değer basmak için kullandığımız bir fonsiyondur.

a) Genel Kullanımı:

ÖR:

```
print("Burası Btk Burası Zirve")
```

b) Aynı Satırda Birden Fazla Değer Bastırma: Print fonksiyonu ile birden fazla değer aralarına virgül konularak yazdırılırken yazdırılan değerler arasına deafult olarak boşluş karakteri eklenir ve bu karakteri sep paremetresi ile değiştirebilir.

ÖR:

- print("Burası","Btk","Burası","Zirve") #Çıktı:Burası Btk Burası Zirve
- print("Burası","Btk","Burası","Zirve", sep="-")
  #Çıktı:Burası-Btk-Burası-Zirve
- print("Burası","Btk","Burası","Zirve",sep = "\n")
  #Çıktı:
  Burası
  Btk
  Burası
  Zirve
- print("Burası","Btk","Burası","Zirve",sep = "\t") #"\t" yazdırılan değişkenlerin
  #arasına bir tab boşluk bırakır.
  # Çıktı:Burası Btk Burası Zirve

c) Format Fonksiyonu ile Formatlama İşlemi: Format fonksiyonunu bir string'in istediğimiz bir yerine int, float veya string tiplerinden bir değer yerleştirmek istediğimiz zaman kullanırız.

- Ör:
  ```
  mesaj = "Btk'nın {}. yılı kutlu olsun.".format(18)
  #Süslü parantezin olduğu yere format fonksiyonu içerisine
  yerleştirdiğimiz #parametreyi ekler.
  print(mesaj) #Çıktı:Btk'nın 18. yılı kutlu olsun.
  ```
- ÖR:
  ```
  a=3
  b=4
  print("{} + {} 'nin toplamı {} 'dır".format(a,b,a+b))
   # Çıktı:3 + 4 'nin toplamı 7 'dır
  ```
- ÖR:
  ```
  mesaj = "{1} ile Programlama Sanatı - {0}".format("BTK", "Python")
  #Süslü parantezlerin içerisindeki sayılar format fonksiyonu içerisindeki
  #parametrelerin string içerisinde yerini göstermek için kullanılır.
  print(mesaj)
  # Çıktı:Python ile Programlama Sanatı - BTK
  ```

11) Listeler: Listeler içinde farklı türden veriler barındıran taşıyıcılarımızdır. Listelerde her bir eleman bir indis(index) numarasına sahiptir ve her listenin başlangıç indisi 0 (sıfır) dır.

a) Listeler ile değişik tipte veriler saklayabiliyoruz.
ÖR: liste = [3,4,5,6,"Elma",3.14,5.324]

b) Boş liste oluşturma
```
bos_liste = []
bos_liste = list()
```

c) Bir string'in karekterlerini bir diziye aktarmak
```
mesaj = "Merhaba"
liste = list(mesaj)
print(liste) # Çıktı: ['M', 'e', 'r', 'h', 'a', 'b', 'a']
```

d) İndex numarası ile listenin elemanlarına erişme:
```
# 0. eleman
liste = [1,2,3,4,5,6,7,8,9,10]
print(liste[2]) # Çıktı:3

#Sonuncu Eleman
liste = [1,2,3,4,5,6,7,8,9,10]
print(liste[len(liste)-1])# Çıktı:10

# Baştan 4. indeks e kadar (4 dahil değil)
liste = [1,2,3,4,5,6,7,8,9,10]
Print(liste[:4]) veya Print(liste[0:4]) #Çıktı: [1, 2, 3, 4]

#2.indeksten 5.indekse kadar
liste = [1,2,3,4,5,6,7,8,9,10]
print(liste[2:5]) # Çıktı: [3, 4, 5]
```

e) İki listeyi toplama(+) oparörörü ile birleştirmek
```
liste1 = [1,2,3,4,5]
liste2 = [6,7,8,9,10]
print(liste1 + liste2)
# Çıktı: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

f) Listenin sonuna "btk" 'yı ekliyoruz.
```
liste = [1,2,3,4]
liste = liste + ["BTK"]
print(liste)
# Çıktı: [1, 2, 3, 4, 'BTK']
```

g) Listenin Birden Fazla Elemanını Birlikte Değiştirme İşlemi:
```
liste = [1,2,3,4,5,6,7,8,9]
liste[0:3] = [10,20,30]
print(liste)
# Çıktı: [10, 20, 30, 4, 5, 6, 7, 8, 9]
```

h) Bir listeyi bir sayıyla çarpma işlemi:
```
liste = [1,2,3]
liste *=3 #liste = liste *3
print(liste)
#Çıktı: [1, 2, 3, 1, 2, 3, 1, 2, 3]
```

i) Append metodu ile listenin sonuna yeni bir eleman ekleme işlemi:

```
meyveListesi = ["Elma", "Armut", "Portakal"]
meyveListesi.append("Muz")
print(meyveListesi)
#Çıktı:['Elma', 'Armut', 'Portakal', 'Muz']
```

j) Pop metodu ile dizinin son karakterini silme:

```
liste = [1,2,3,4,5]
liste.pop()
print(liste)
#Çıktı: [1, 2, 3, 4]
```

k) Pop metodu ile index numarasını kullanarak kayıt silme:

Kullanımı: liste_adi.pop(index_numarasi)

ÖR:

```
liste = [1,2,3,4,5]
liste.pop(2)
print(liste)
#Çıktı: [1, 2, 4, 5]
```

l) Sort metodu ile sıralama işlemi:

- Küçükten büyüğe sıralama:

  ÖR:

  ```
  liste = [34,1,56,334,23,2,3,19]
  liste.sort()
  print(liste)#Çıktı: [1, 2, 3, 19, 23, 34, 56, 334]
  ```

- Büyükten küçüğe sıralama:

  ÖR:

  ```
  liste = [34,1,56,334,23,2,3,19]
  liste.sort(reverse=True)
  print(liste)
  #Çıktı: [334, 56, 34, 23, 19, 3, 2, 1]
  ```

- Alfabetik olarak küçükten büyüğe sıralama işlemi:

  ÖR:

  ```
  liste = ["Elma","Armut","Muz","Kiraz"]
  liste.sort()
  print(liste)
  #Çıktı: ['Armut', 'Elma', 'Kiraz', 'Muz']
  ```

- Alfabetik olarak büyükten küçüğe sıralama işlemi:

  ÖR:

  ```
  liste = ["Elma","Armut","Muz","Kiraz"]
  liste.sort(reverse=True)
  print(liste) #Çıktı: ['Muz', 'Kiraz', 'Elma', 'Armut']
  ```

- İç içe listeler
  ÖR:

```
sayilar = [1,2,3]
meyveler = ["Elma", "Armut", "Muz"]
hayvanlar = ["Köpek","Kedi","Fare"]
yeniListe = [sayilar,meyveler,hayvanlar]
print(yeniListe)
print(yeniListe[0][2])#Çıktı:3
print(yeniListe[1][1])#Çıktı:Armut
print(yeniListe[2][2])#Çıktı:Fare
```

- index metodu ile arama yapma: Aranılan dizi elemanının index numarasını verir.
  ÖR:

```
hayvanlarim = ["Köpek","Kedi","Fare"]
bulunanIndex = hayvanlarim.index("Fare")
```

12) Sözlükler(Dictionary): Gerçek hayattaki sözlükleri benzer. Sözlüğün içindeki her bir eleman indeks ile değil, anahtar (key), değer (value) olarak tutulur.

ÖR1:

```
sozluk = {"give":"vermek","red":"kırmızı","moon":"ay"}
print(sozluk["give"]) #Çıktı: vermek
```

ÖR2:

```
sozluk = {"give":"vermek","red":"kırmızı","moon":"ay"}
anahtar = input("İngilizce kelime giriniz..") #klavyeden give girildiğinde
print(anahtar + " = " + sozluk[anahtar]) #Çıktı: give = vermek
```

a) Boş Sözlük Oluşturma

```
sozluk = {}
print(type(sozluk))

sozluk1 = dict()
print(type(sozluk1))
```

b) Yeni eleman ekleme

```
urunler = {"Elma": 2, "Kiraz": 3, "Muz":6}
urunler["Çilek"] = 5
print(urunler)
```

c) İç içe sözlükler:

```
urunler={"meyveler":{"Muz":11,"Elma":3,"Salatalık":2},"icecekler":{"Kola":4,
"Süt":3,"Meyve_suyu":2}}
print(str(urunler["meyveler"]["Elma"]) + " TL" ) #Çıktı: 3 TL
```

13) Mantıksal Değerler ve karşılaştırma Operatörleri:

a) Mantıksal Değerler:

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

b = False
print(type(b))

bool(1.2) #True
bool(-5) #True
bool(0) #False
bool("a") #True
bool("") #False
```

b) Karşılaştırma Operatörletri

== , != , > , <, >= ,<=

c) Mantıksal Bağlaçlar

and, or, not

```
not (( 2 != 2 and "Btk" == "Btk") or (3.14 > 3.15))# False
```

14) Koşul Durumları:

a)

Python programları çalışmaya başladığı zaman kodlarımız yukardan başlayarak teker teker çalıştırılır ve çalıştıracak kod kalmayınca programımız sona erer. Şöyle bir örneği düşünelim;

```
a=21
b = 6
print(a+b)
```

Yukarıdaki basit kodda program teker teker her bir satırı ve ifadeyi çalıştırır ve çalıştıracak kod kalmayınca program sona erer.Ancak Python'da her program bu kadar basit olmayabilir. Çoğu zaman Python programlarımız belirli bloklardan oluşur ve bu bloklar her zaman çalışmak zorunda olmaz.Peki bu bloklar nasıl tanımlanır ? Pythonda bir blok tanımlama işlemi Girintiler sayesinde olmaktadır. Örnek olması açısından, Pythonda bloklar şu şekilde oluşabilir.

ÖR:

```
say = int(input("Bir sayı giriniz: ")) # Blok 1 'e ait kod
if (say % 2 == 1):
   print("{} tek sayıdır.".format(say)) # Blok 2'ye ait kod
print("Btk Python Eğitimi ") # Blok 1 ' ait kod
```

b) İf – else

ÖR:

```
yas = int(input("Lütfen yaşınızı giriniz : "))
if(yas >= 18):
   print("Ehliyet alabilirsin")
else:
   print("18 yaşı beklemek zorundasın!")
```

c) if - elif – else

```
vize1 = int(input("Vize1:"))
vize2 = int(input("Vize2:"))
final = int(input("Final:"))

genel_not =  vize1 * 3/10 + vize2 * 3/10 + final * 4/10

if (genel_not >= 90):
   print("AA")
elif (genel_not >= 85):
   print("BA")
elif (genel_not >= 80):
   print("BB")
elif (genel_not >= 75):
   print("CB")
elif (genel_not >= 70):
   print("CC")
elif (genel_not >= 65):
   print("DC")
elif (genel_not >= 60):
   print("DD")
elif (genel_not >= 55):
   print("FD")
else:
   print("FF")
```

15) Döngü Yapıları:

Şimdiye kadar yazdığımız programlarda yazdığımız programlar bir defa çalışıyor ve sona eriyordu. Ancak biz çoğu zaman programlarımızın belli koşullarda çalışmasını sürekli devam ettirmesini ve işlemlerini tekrar etmesini isteriz. İşte bunları yapmamızı sağlayan yapılara döngü diyoruz.

a) For:

1) in operatörü:

```
"b" in "btk"
3 in [1,2,3,4]
```

2)
```
liste = [1,2,3,4,5,6,7]
for eleman in liste:
    print("Eleman",eleman)
```

3)
```
liste = [1,2,3,4,5,6,7]
toplam = 0
for eleman in liste:
    toplam += eleman
print("Toplam",toplam)
```

4) Karakterler üzerinde gezinme
```
s =  "Python"
for karakter in s:
    print(karakter)
```

5) Demetler Üzerinde Gezinme
```
# Listelerle aynı mantık
demet =  (1,2,3,4,5,6,7)
for eleman in demet:
    print(eleman)
```

6)
```
liste = [[1,"İsmail","1990"],[2,"Sami","2019"],[3,"irem","1999"]]

for a,b,c in liste:
    print(a,b,c)
```

7) Sözlükler üzerinde gezinme

```
sözlük = {"bir":1,"iki":2,"üç":3,"dört":4}
sözlük.values()
sözlük.keys()

for eleman in sözlük:
   print(eleman) # sadece keyleri

for eleman in sözlük.values():
   print(eleman)#degerleri
```

b) While Döngüleri

```
i = 0
while (i < 10):
   print("i'nin değeri",i)
   i += 1

i = 0
while (i < 20):
   print("Python Öğreniyorum")
   i +=1 #eğer bu komut olmasaydı sonsuz döngü
```

c) Range() Fonksiyonu:

Pythondaki bu hazır fonksiyon bizim verdiğimiz değerlere göre range isimli bir yapı oluşturur ve bu yapı listelere oldukça benzer. Bu yapı başlangıç, bitiş ve opsiyonel olarak artırma değeri alarak listelere benzeyen bir sayı dizisi oluşturur.

1) range(0,20) # 0'dan 20' a kadar (dahil değil)
   print(*range(5,10))
2) Listeye çevirme
   liste = list(range(0,20))
3) print(*range(0,20,2)) #2 atlayarak oluştur
4) print(*range(20,0,-1)) # 20'den geri gelen sayıları oluşturur.
   #20 19 18 ...

```
for sayı in range(0,20):
      print("Python Öğreniyorum")
```

5)
```
for sayı in range(0,10):
      print("* " * sayı)
```

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

6)

a)

```
liste1 = [1,2,3,4,5,6,7,8,9,10]
liste2 = list()
for i in liste:
   liste2.append(i)
print(liste2)
```

b)

```
liste1 = [1,2,3,4,5,6,7,8,9,10]
liste2 = [i for i in liste1] #List Comprehension
print(liste2)
```

7) 6 üzerinde 2 ile çarpılmış hali

```
liste = [1,2,3,4,5,6,7,8,9,10]
liste = [i ** 2 for i in liste]
print(liste) #Çıktı: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

8)

```
liste = [1,2,3,4,5,6,7,8,9,10]
liste = [i ** 2 for i in liste if(i % 2 == 0)]
print(liste) #Çıktı: [4, 16, 36, 64, 100]
```

9) break

```
liste = [1,2,3,4,5,6,7,8,9,10]
for i in liste:
   if(i == 5):
      break
   else:
      print(i)
```

Çıktı:

1
2
3
4

10) continue

```
i = 0

while (i < 6):

   if (i == 3):
      i += 1  # 1 artırma işlemi yapmasaydık sonsuz döngü olurdu
      continue

   print(i)
   i += 1
```

Çıktı:

0
1
2
4
5

11) break ve continue kullanım örneği

```
import random
while  True:
   tahmin = int(input("0 - 5 arası bir sayı giriniz!"))
   if(tahmin < 0 or tahmin > 5):
      print("Lütfen girdiğiniz sayı 0-5 aralığında olsun!")
      continue
   say = random.randint(0,5)
   if(say == tahmin):
      print("Tebrikler tahmininiz doğru.")
      break;
```

12) ALIŞTIRMALAR

a) Kullanıcıdan input olarak bir cümle alınız.
Ve çıktı olarak cümledeki 'a' harfi sayısını ekrana basınız.
Çözüm:

```
counter = 0
for c in cumle:
   if c == 'a':
      counter += 1
print(counter)
```

b) Çarpım Taplosu

```
for i in range(1,11):
   for j in range(1,11):
      print("{0} * {1} = {2}".format(i,j,i*j))
   print("********************({})********************".format(i))
```

16) Fonksiyonlar:

1) Fonksiyon tanımlama

```
def selamla():
 print("Merhaba")
```

2) Parametre Kullanımı

```
def selamla(isim):
   print("Merhaba",isim)

selamla("İsmail")#Çıktı: Merhaba İsmail
```

3) Girilen 3 sayının toplamını

```
def topla(a,b,c):
   print("{} + {} + {} = {}".format(a,b,c,a+b+c))
topla(1,2,3)
```

4) Return kavramı

```
def topla(a,b,c):
   return a+b+c
print("Toplam",topla(1,2,3))
```

5) Faktoriyel hesaplama

```
def faktoriyel(sayi):
    faktoriyel = 1
    if(sayi < 0):
        return -1
    elif (sayi == 0 or sayi == 1):
        return faktoriyel
    else:
        for i in range(1,sayi + 1):
            faktoriyel *=i
        return faktoriyel

say = int(input("Bir sayı giriniz "))
sonuc = faktoriyel(say)
if(sonuc < 0):
    print("Negatif sayıların faktoriyeli hesaplanamaz!")
else:
    print("{}!={}".format(say,sonuc))
```

ÇIKTI:

```
Bir sayı giriniz 5
5!=120
```

6) Parametrelerin varsayılan değerleri

```
def selamla(isim = "İsimsiz"):
    print("Merhaba",isim)

selamla()
```

7) Bir fonksiyonu esnek sayıda parametre ile çağırma:

a)

```
def test(*sayilar):
    print(sayilar)
test(1,2,3,4,5)
```

b)

```
def topla(*sayilar):
    top = 0
    for i in sayilar:
        top+=i
    return top;
print(topla(1,2,3,4,5))
#Çıktı: 15
```

8) Global ve Yerel Değişkenler

1) <u>Yerel değişken</u>

```
def yerel():
    say = 5
    print(say)

yerel()
print(say) #say değişkeni yerelde tanımlı olduğu için hata!!!
```

2) Global Değişken

```
say = 5
def globalDeg():
  print(say) #say değişkeni global olduğu için fonk içerisinde kullanabildik

globalDeg()
```

3) Sas

```
def globalDeg():
  print(say)

globalDeg()
say = 5
```

Açıklama: say değişkeni global olmasına rağmen fonsiyonu çağırdıktan sonra tanımladığımız için hata aldık

4) Fonksiyon içerisinde global değişkenleri kullana

1)

```
say = 10
def globalDeg():
  say = 5 #Local değişken tanımlandı
  print(say)

globalDeg()
print(say)
```

Çıktı:
5
10

2)

```
say = 10
def globalDeg():
  global say #say adında local bir değişken tanımlamayıp, global say
değişkenini kullanmak istiyorum dedik.
  say = 5
  print(say)

globalDeg()
print(say)
```

Çıktı:
5
5

9) Lambda İfadeleri:
   a) Normal Fonk.
      ```
      def ciftMi(say):
        return (say % 2) == 0

      print(ciftMi(4)) #True
      ```
   b) Lamda Fonk.
      ```
      ciftmi = lambda say : (say % 2) == 0

      print(ciftMi(3)) #False
      ```
   c) 3 Sayının toplamını veren lambda fonk.
      ```
      topla = lambda a,b,c : a+b+c
      print(topla(1,2,3))
      ```
   d) Dinamik sayıda argüman alan lambda fonksiyonlar
      ```
      karesiniAl = lambda *sayilar : [i **2 for i in sayilar]
      print(karesiniAl(1,2,3,4,5,6,7,8,9))
      #Çıktı: [1, 4, 9, 16, 25, 36, 49, 64, 81]
      ```

10) Alıştırmalar
   a) Girilen karaktere göre bir metni parçalayarak bir diziye aktaran bir fonksiyon yazınız.(Python split fonsiyonu gibi)
      ```
      #split fonksiyonu
      msj = "burası btk burası zirve"
      kelimeler = msj.split(" ")
      print(type(kelimeler))
      print(kelimeler)
      ```

      Çözüm:
      ```
      def ayir(metin, karakter):
        dizi = []
        kelime = ""
        for c in metin:
          if c != karakter:
            kelime += c
          else:
            dizi.append(kelime)
            kelime = ""
        dizi.append(kelime) #bu satırı paragraf sonunda boşluk olmadığı için yazdık
        return dizi

      msj = input("Lütfen split etmek istediğiniz metni giriniz")
      liste = ayir(msj," ")
      print(liste)
      ```