0-type(): bir değişkenin veri türünü bulmak için kullanılıyor!

-nesne: pythonda çalışılan her şey, tüm değişken türleri vs.., nesnedir. Şey denmez nesne denir! 😊

-pycharm programı hem python ile hem de bilgisayar işletim sistemi(cmd) ile konuşmak için bir ortam sağlıyor. Pycharmda altta bulunan terminal kısmından yazılan kodlar işletim sistemiyle haberleşmeyi sağlıyor!

\*SANAL ORTAM VE PAKET YÖNETİMİ KODLARI:

#sanal ortam ve paket yönetimi kodları  
## bilgisayardaki sanal ortamlar ile ilgili **TERMİNALE** yazılacak kodlar:  
#1. sanal ortam listeme: conda env list  
#2. sanal ortam oluşturma: conda create -n sanalortamadi  
#3. herhangi bir sanal ortama geçme: conda activate sanalortamadi  
#4. herhangi bir sanal ortamdan çıkma: conda deactivate  
#5. sanal ortam içindeki paketleri görüntüleme: conda list  
#6. paket yükleme: conda install packetadi  
#7. aynı anda birden fazla paket yükleme: conda install paket1 paket2 paket3  
#8. paket silme: conda remove paketadi  
#9. belirli bir versiyona göre paket yüklemek için: conda install numpy=1.20.1  
#10. belirli bir paketin versiyonunu yükseltmek için: conda upgrade packetadi  
#11. tüm paketlerin versiyonlarının yükseltmek için: conda upgrade -all  
#12. pip: pypi (python packege index) paket yönetim aracıdır.  
#13. pip ile paket yükleme komutu: pip install paketadi  
#14. pip ile paket versiyonu yükseltme: pip install numpy==versiyon not: önceki paketi otomatik silip yenisi yükelenecek. Paket yönetim araçlarından biri olarak pip in amacı istenen versiyon yüklenmeden önce, bilgisayardaki mevcut versiyonun silinmesini de sağlamaktır!  
#15. ortamlarda bulunan paket ve bağımlı paketlerin bilgisini topluca bir dosyaya aktarmak için enviroment.yaml(conda) ya da requirements.txt(pip) dosyas oluşturulur.  
Paketleri .yaml ya da .yml uzantılı dosyaya aktarmak için komut: conda env export > dosyaadi.yaml  
#16. sanal ortam silme: conda env remove -n sanalortamadi  
#17. Bir .yaml ya da .yml uzantılı dosyadan yeni bir sanal ortam oluşturma: conda env create -f dosyadi.yaml. -f ifadesi bir dosyadan veri çekileceğini ifade ediyor! Eski environemnt adıyla birlikte yeniden oluşuyor!

\*VERİ YAPILARI

#Sayılar  
a = 5 #integer  
a \*\* 2  
b = 10.5 #float  
##Tipleri Değiştirmek  
int(b) #float olan değerin sadece onralıklı kısmını alıyor!  
float(a)  
int(a \* b / 10)

##Karakter Dizileri(Strings)  
print("Kerem")  
print('Kerem')  
name = "Kerem"  
  
#Çok satırlı Karakter dizileri(string)  
long\_str = """ sema   
kerem   
oğuz""" not: çok satırlı karakter dizilerini yazmak için ilgili dizi üçer çift/tek tırnak içinde yazılır!  
  
#Karekter Dizilerinin Elemanlarını Seçmek  
name[0] #K harfi. index değeri 0 dan başlıyor  
name[4] #m harfi.  
#Karakter dizilerinde Slice İşlemi  
name[0:2] #Ke harfleri. 0.indexten başla, 2. index'e git ama 2. index i dahil etme!  
  
#String içinde eleman sorgulamak  
"kerem" in long\_str #"kerem" kelimesi long\_str dizisinin içinde mi diye soruluyor. cevap:true  
"Kerem" in long\_str # cevap false. python büyük küçük harfe duyarlı!

**NOT: BİR PYTHON FOKSİYONU/METODUNUN ÜSTÜNE CTRL’YE BASILIP TIKLANDIĞINDA ONUNLA İLGİLİ BİLGİLERE ULAŞILIR. FONKSİYONLAR METODLARA GÖRE BAĞIMZISDIR, METODLAR CLASS’LAR İÇİNDE YAZILMIŞTIR! SELF İFADESİ CLASS OLDUĞUNU BELİRTİR!**

##String metodları:  
#metod: sınıflar(class) içinde tanımlanan fonksiyonlardır. ama fonksiyondan farklıdır  
#dir(veritipi) ile ilgili veri tipinin tüm metodlarına ulaşılır  
dir(str) #string veri tipine ait metodlar  
#len fonksiyonu:  
len("kerem") #karakter uzunluğu  
namea = "oğuz"  
len(namea)  
  
#UPPER/LOWER METDOLARI:  
"kerem".upper()  
"miuul".upper()  
"MİULL".lower()  
"oğuz".upper()  
"istanbul".upper() #çıktı:Istanbul  
#replace:istenilen karakteri değiştirir  
hi = "Hello AI ERA"  
hi.replace("l", "p") #çıktı: "Heppo AI ERA"  
#split: istenilen karakteri istenilen yerden böler  
hi.split() #boşluktan itibaren böler. sonuç bir dizidir!  
hi.split()[0] #çıktı:"Hello"  
  
#strip: kırpma metodu : karakterin solundaki ve sağındaki metodu kırpar  
" kerem ".strip() #çıktı: "kerem"  
"kerek".strip("k") #çıktı: "ere" : baştaki ve sondaki k harklerine göre kırpma yaptı  
#capitalize: ilk harfi büyük yapar!  
"kerem".capitalize() çıktı: “Kerem”

#Liste / List  
#Değiştirilebilir, Sıralıdır(index işlemleri yapılabilir), Kapsayıcıdır(Liste içinde birden fazla veriyapısı bulunabilir)  
  
notes = [1,2,3,4]  
type(notes)  
  
notes1 = ["a", "b", True,[1,2,3]]  
notes1[3] #çıktı: [1,2,3]  
notes1[3][1] #çıktı:2  
notes1[2:4] #2'den başla 4'e kadar(4. index hariç) al. cıktı:[True, [1,2,3]]  
notes1[0] = 100 #listenin 0. index değeri değişti. yeni notes1 = [100, "b",True,[1,2,3]]  
####Liste Metodları#####  
notes1.count(100) ‘ “100” ifadesinin kaç tane olduğunu sayıyor  
notes1[3].sort(reverse=False)  
#append: listenin sonuna eleman ekliyor.  
notes1.append(250) #çıktı:[100, 'b', True, [1, 2, 3], 250]  
#pop: index e göre listeden eleman siliyor!  
notes1.pop(0) #0. index'deki eleman siliniyor  
#insert: index e göre istenilen index'e eleman ekler!  
notes1.insert(1,"kalem") #[100, 'kalem', 'b', True, [1, 2, 3]]  
  
#Dictionary / Sözlük  
#key-value olarak çalışırlar.  
##Değiştirilebilirler, Sırasız(3.7'den sonra sıralı), Kapsayıcıdır(birden fazla veri yapısı içerebilir)  
sozluk = {"REG" : "7",  
 "LOG" : 19,  
 "CART" : True}  
sozluk["REG"] #çıktı:"7" --> dictionary'de key sorulur value cevabı alınır!  
sozluk["LOG"] #çıktı:19  
  
#KEY Sorgulama  
"REG" in sozluk #sözlük içinde "REG" key değeri var mı  
#KEY'e göre VALUE değerine ulaşmak  
sozluk.get("REG") #çıktı:'7'  
sozluk.get("LOG") #çıktı: 19  
  
sozluk["REG"] = ["s",14] #yenisözlük listesi: {'REG': ['s', 14], 'LOG': 19, 'CART': True}  
  
#Tüm KEY'lere Erişmek  
sozluk.keys() #çıktı:dict\_keys(['REG', 'LOG', 'CART'])  
#Tüm Value'lere Erişmek  
sozluk.values() #çıktı: dict\_keys(['REG', 'LOG', 'CART'])  
  
#Tüm çiftlere Tupple formatında Erişme  
sozluk.items() #çıktı: dict\_items([('REG', ['s', 14]), ('LOG', 19), ('CART', True)])  
#key-value değerlerini güncelleme/yeni key-value değeri atama  
  
sozluk.update({"REG":38}) #ÇIKTI: {'REG': 38, 'LOG': 19, 'CART': True}  
sozluk.update({"oyun":"counter"}) #çıktı: {'REG': 38, 'LOG': 19, 'CART': True, 'oyun': 'counter'}

Dictionary veri yapısında UPDATE kumutu eğer sözlük yapısında lgili key değeri varsa onu günceller, yoksa yeni ekler!

**NOT: VERİ YAPILARININ SIRALI ÖZELLİKTE OLMASI, İÇLERİNDE TUTTUKLARI VERİLERE ULAŞILABİLECEĞİ ANLAMINDADIR!**

#Tuple(Demet) : Listenin aksi/huysuz kardeşi :)  
#Değiştirilemez, Sıralıdır(index ile tüm verilere ulaşılır) ve Kapsayıcıdır(tüm veri yapılarının içinde tutar)  
t = (1, 2, "oguz")  
t = list(t)  
t[0] = 13  
t = tuple(t)  
print(t) #çıktı: (13, 2, 'oguz')  
### not: tuple veri yapısı değiştirilemez. ama veri eklemek için önce listeye çevirilir, veri eklenir\_  
### tekrar tuple veri yapısına çevrilir  
  
###### Set(Küme): daha çok küme işlemleri için kullanılır; kesişim, birleşim, fark vs..  
#Değiştirilebilir  
# Sırasız + Eşsiz (index işlemleri ile verilere ulaşılamaz ve her veri eşsiz olarak saklanır!  
# Kapsayıcıdır(tüm veri yapılarını içerebilir)  
  
######  
#İki kümenin farkı ( Difference)  
####  
set1 = set([1,2,3,4,42,100]) #çıktı: {1, 2, 3, 4}  
set2 = set([1,2,10,20,3,4,6])  
set3 = {1,2,10,20,3,4}  
#ÖNEMLİ NOT: Set'ler süslü parantez ile ifade edilir. set fonksiyonu ile bir set oluşturulurken  
#normal parantez içinde Liste yapısı(köşeli parantez) kullanılır!  
  
#set1'de olup set2 de olmayanlar  
set1.difference(set2)  
set1 - set2 #eksi işaretiyle de iki küme arasındaki fark bulunuyor1  
#set2'de olup set1 de olmayanlar  
set2.difference(set1)  
#set1'de olup set2'de olmayanlar ile set2'de olup set1'de olmayanların birleşimi(simetrik difference)  
#symetric\_difference iki kümede de birbirinde olmayan elemanların birleşiminden oluşur  
set1.symmetric\_difference(set2)  
  
#intersection: iki kümenin Kesişimi  
set1.intersection(set2)  
set2.intersection(set1)  
set1 & set2 # & işaretiyle iki kümedeki kesişim bulunuyor!  
  
#union: iki kümenin birleşimi  
  
set1.union(set2) #çıktı: {1, 2, 3, 4, 6, 10, 20, 42, 100}  
#kümelerde aynı eleman birden fazla bulunamaz!  
  
#elimizdeki iki kümenin kesişimi boş mu ?  
#isdisjoint() ile bu sorunun cevabı bulunuyor!  
  
set2.isdisjoint(set1) #cevap: false / boolean veri yapısı  
  
#bir küme diğernin alt kümesi mi ? issubset()  
set1.issubset(set2) #cevap false (set1 set2'nin alt kümesi mi?)  
set3.issubset(set2) #cevap true (set3 set2'nin alt kümesi mi?)  
  
#Bİr küme diğerini kapsıyor mu? issupperset()  
set1.issuperset(set2) #set1 set2'yi kapsıyor mu?  
set2.issuperset(set3) #set2 set3'ü kapsıyor mu?

-----------------------------------------------------FONKSİYONLAR-------------------------------------------------------

##Fonksiyon OKUR YAZARLIĞI!!  
?print  
#çıktı: print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False) sep,end ve flush fonksiyonların argümanlarıdır  
#sep=seperator olmalı. ikiden fazla değeri yazdırırken kullanılan ayraç. default olarak boşluk tanımlı  
print(5,6, sep = "--")  
##DOCSTRING: KULLANILAN TÜM FONKSİYONLARIN DÖKÜMASYONLARINA DOCSTRING DENİR!  
#fonksiyonların dökümantasyon kısmına ulaşmak için "help(fonksiyon)" ya da "?fonksiyon" kullanılır!  
  
#######################  
## FONKSİYON TANIMLAMA  
#######################  
def my\_function(a): #def define dan geliyor olmalı.  
 print(a\*2)  
  
my\_function(5) #çıktı: 10  
  
def cikarma(arg1, arg2):  
 print(arg1 - arg2)  
  
cikarma(7,8) #çıktı:-1  
cikarma(8,7) #çıktı:1  
#not: fonksiyon tanımlarken argümanların sıralaması önemlidir!

##########################  
####DOCSTRING (FONKSİYONLARA BİLGİ NOTU/KULLANIM KILAVUZU EKLEYEREK DOCSTRING/FONKSYONUN DÖKÜMANTOSYONU OLUŞTURULUR)  
###########################  
  
def toplamak(x,y): #bu satırın altına gelinir ve 3 tane çift tırnak oluşturduktan sonra enter a basılır!  
SONRA DOCTRING AŞAĞIDAKİ GİBİ YAZILIR----------------------------------------------------  
 *"""  
 sum of two numbers  
 Parameters  
 ----------  
 x: int, float  
 y:int, float  
  
 Returns  
 -------  
 int, float  
 """* print(x + y)  
-----------------------------------------------------------------------------------------------------  
help(toplamak)

###FONKSİYONLARIN STATEMENT/BODY/GÖVE BÖLÜMÜ  
#def my\_function(a):  
 #Statement (Function Body)  
  
#girilen değerleri birbiriyle çarpıp bir liste içine atan fonksiyon yazalım  
liste1 = []  
def myfunction(a, b):  
 *"""  
  
 Parameters  
 ----------  
 a: int, float  
 b: int, float  
  
 Returns  
 c: integer, float  
 -------  
  
 """* c = a \* b  
 liste1.append(c) #not: burada liste1 = liste1.append(c) kullanılmadı. çünkü append metodu gbi bazı metodlar kalıcı değişiklik yapabiliyor!  
 print(liste1)  
myfunction(9,18)

###################### ön tanımlı argümanlar  
##### ön tanımlı argümanları hem kullanııc hatasının önüne geçmek hem de default değer tanımlamak için kullanıyoruz!  
  
def bolme(x, y=1): #y değeri ön tnaımlı olarka 1 değerine eşitlendi ki, kullanıcı hatasına vs tedbir alınsın  
 return float(x / y) #return ifadesi fonksiyonun değerinin bir değişkene atanmasını sağlamak amacıyla kullanılıyor!!  
k = bolme(4)  
### Ne zaman fonksiyon yazılır!  
# Sürekli tekrar eden işlemleri kolaylaştırmak amacıyla donksiyon yazılır.  
# yazılımcılığın DRY Prensibi: DRY: DONT REPEAT YOURSELF  
  
def akilli\_lamba(nem, sicaklık, pil\_durumu):  
 return (sicaklık + nem) / pil\_durumu  
akilli\_lamba(98, 12, 78)  
  
  
# Return kullanımı: Fonksiyonların çıktılarını belirli bir değişkene atayarak başka işlemlerde girdi olarak kullanmamızı sağlar!  
def akilli\_lamba(nem, sicaklık, pil\_durumu):  
 nem = nem \* 2  
 sicaklık = sicaklık \* 3  
 pil\_durumu = pil\_durumu \* 1.2  
 output = (sicaklık + nem) / pil\_durumu  
 #return nem, sicaklık, pil\_durumu, (sicaklık + nem) / pil\_durumu  
 return (nem, sicaklık, pil\_durumu,  
 output) #pythonda alt satıra geçmek oldukça kolay! parantezler arasında direk olarka geçilebilir!  
 ##Not: Return ile birden fazla çıktı değerini Tuple olarak alabiliriz!  
  
moisture, warm, charge, output = akilli\_lamba(98, 12, 78) #aynı anda birden fazla değişkene değer atanabilir!  
# (196) (36) (93,6) (2,4786)  
  
def akilli\_lamba1(nem, sicaklık, pil\_durumu):  
 return (sicaklık + nem) / pil\_durumu  
akilli\_lamba1(98, 12, 78)  
  
def deger1(a, p):  
 return int(a\* 10 / 100 \* p\*\*2)  
deger1(3,99)  
  
def all\_calculate(a , b, c, d):  
 m = akilli\_lamba1(a, b, c)  
 n = deger1(m, d)

return n  
 # not: fonksiyon içinde fonksiyon çağırırken ana fonksiyonun tüm değişkenleri alt fonksiyonlardda kullanılmalı  
 # yani a,b,c,d değerleri tüm alt donksiyonlarda kullanılmıitır  
all\_calculate(10,20,30,40)

students = ["John", "Mark", "Venessa", "Mariam"]  
  
for i in students: #burada i yerine herşey yazılabilir  
 print(i)  
  
salaries = [1000, 2000, 3000, 4000, 5000]  
def maas\_artis(maas,artis\_orani):  
 k = maas + maas \* artis\_orani/100  
 return k  
maas\_artis (1500, 30)  
  
for i in salaries:  
 n = int(maas\_artis(i,10))  
 print(n) # çıktı = #1100.0 /n 2200.0 /n 3300 /n 44000 /n 5500 (/n=alt satır)  
  
salaries2 = [1220, 202200, 300330, 40040, 50200]  
for i in salaries2:  
 if i <= 3000:  
 n = maas\_artis(i, 30)  
  
 elif i > 3000 and i < 50000:  
 n = maas\_artis(i, 20)  
  
 else:  
 n = maas\_artis(i, 100)  
 print(n)  
#################################  
##########UYGULAMA / MÜLAKAT SORUSU:  
#################################  
# Verilen: "hi my name is john and i am learning python"  
# istenen: "Hi mY NaMe iS JoHn aNd İ aM LeArNiNg pYtHoN"  
  
# Dikkat: istenen kısmda çift sayıdaki index değerlerindeki harfler büyük istenmiş!!!  
  
range(len("miuul")) #range fonksiyonu default olarak 0'dan istenen değere kadar(istenen değer hariç!) dizi oluşturur  
  
def alternating(dizi): # dizi adında bir değişken tanımlandı  
 *"""  
  
 Parameters  
 ----------  
 dizi: string  
  
 Returns  
 -------  
 string  
 """* yeni\_dizi = ""  
 for i in range(len(dizi)): #range ile dizi değişkeninin karakter saylarının index değerleri elde ediliyor  
 if i % 2 == 0: #çift index değeri bulundu!  
 yeni\_dizi += dizi[i].upper() # anlamı: yeni\_dizi = yeni\_dizi + dizi[i].upper()  
 #Not: Python'da metin birleştirme + işareti ile yapılıyor!  
 else:  
 yeni\_dizi += dizi[i].lower()  
 #Not: Sorunun cevabı için tek index değerlerinde olan harfleri küçükltmeye gerek yok  
 # genel bir foksiyon yazılmaya çalışıldı  
 print(yeni\_dizi)  
  
alternating("miuul") #çıktı: MiUuL  
alternating("hi my name is john and i am learning python")  
#############################  
#### break & continue & while  
############################  
  
salaries3 = [1000, 3000, 5000, 7000, 9000]  
 #break:  
for a in salaries3:  
 if a == 5000:  
 print(a)  
 # not: burda sadece 1000 ve 3000 değerleri mevcut. 5000'deki değeri almıyor. i= 5000 olunca döngü kesiliyor  
 #continue  
 break  
#continue  
 for i in salaries3:  
 if i == 5000:  
 continue  
  
 print(i)  
 #Not: burda eğer i değeri 500 olunca döngü bu değeri atlıyor. print ile 5000 değeri yazdırılmıyor!  
 #while:  
 i = 0  
 while i < 5:  
 if i == 4:  
 continue  
 i += 1  
 print(i)

#############  
######Enumerate (Otomatik Counter / Indexer ile for loop (değişkenlerin hem index hem de kendi değerleriyle işlem yapmayı sağlıyor  
#########  
  
students5 = ["John", "Mark", "Venessa", "Mariam"]  
dizi1 = "kerem"  
for i, j in enumerate(dizi1): #enumerate ile otomatik olarak index ve değişken değerleri kastediliyor  
   
 if i % 2 == 0:  
 j = j.upper()  
 else:  
 j = j.lower()  
 print(i, j)  
 #çıktı: 0 K  
 #1 e  
 #2 R  
 #3 e  
 #4 M

####alternatin fonksiyonunun enumarate ile tekrar yazılması  
def alternating2(dizi2):  
 yeni\_dizi2 = ""  
 for i, j in enumerate(dizi2):  
 if i % 2 == 0:  
 yeni\_dizi2 += j.upper()  
 else:  
 yeni\_dizi2 += j.lower()  
 print(yeni\_dizi2)  
  
  
alternating2("hi my name is john and i am learning python")  
  
####ZIP fonksiyonu: farklı listeleri aynı index değerleri birbirine denk gelecek şekilde eşler.! zip bir veri türüdür!  
##### sonucun istenilen şekilde yansıtılması list ya da tupple formatına dönüşüm yapılması gerekmektedir.  
students6 = ["John", "Mark", "Venessa", "Mariam"]  
departments = ["mathematics", "statistics", "physics", "astronomy"]  
ages = [23, 30, 26, 22]  
x = zip(students6, departments, ages)  
list(x)  
  
  
#####ÇOK ÖNEMLİ(lambda, map), filter ve reduce fonsiyonları!!!!  
  
##lambda (def gibi fonksiyon tanımlama için kullanılır. tek farkı atamam yapılmadan tek seferde kullan at fonksiyon  
# tanımlanabilir  
  
def summerr(a, b):  
 return a + b  
  
  
new\_sum = lambda a,b: a + b # normalde lambda kullanıldığında değişkene atama yapılmaz. bunun yerine def kullanılır. !!  
new\_sum(3, 6)  
  
##map: map'e bir fonsiyon ver, akabinde iteratif bir nesne ver. o fonksiyonu o nesneye uygulasın!! çıktıyı liste olark alabiliriz!  
  
salaries2 = [1000, 2000, 3000, 4000]  
  
  
def zammer(maas):  
 return maas \* 20 / 100 + maas  
  
  
for i in salaries2:  
 i = zammer(i)  
 print(i)  
  
list(map(zammer, salaries2))  
list(map(lambda x: x \* 0.2 + x,  
 salaries2)) # burada lambda fonksiyonu ile her bir salaries2 nesnesine %20 zam yapan fonksiyon yazılıyor!  
# del new\_sum  
list(map(lambda x: x \*\* 2 / 100 + x, salaries2))  
  
# Filter: filter'a bir fonksyion ve iteratif bir nesne ver, fonksyimonu her bir nesneye iteratif uygulasın!  
  
list\_store = [1, 2, 5, 10, 11, 14, 15]  
list(filter(lambda x: x % 2 == 0, list\_store)) # çift sayıları filtreledi!  
  
# REDUCE: bir nesnenin tüm elemanlarına işlemler yapılarak tek bir nesneye dönüştürür  
  
from functools import reduce  
list\_store = [1, 2, 5, 10, 11, 14, 15]  
reduce(lambda a, b: a + b, list\_store) #list\_stor elemanlarının toplamını verir. çıktı: 58

# COMPREHENSIONS

#ÇOK ÖNEMLİ KONUDUR!  
#comprehension: Birden fazla yapılacakk işlemleri istediğimiz veri türüne göre çıktı almamızı sağlayan yapılardır.  
##list comprehendisons  
def zammer(x):  
 return x \* 0.2 + x  
salaries = [1000, 3000, 5000, 7000, 9000]  
new\_salaries = [zammer(a) for a in salaries] #çıktı: [1200.0, 3600.0, 6000.0, 8400.0, 10800.0]  
new\_salaries = [zammer(a) for a in salaries if a > 3000] #çıktı: [6000.0, 8400.0, 10800.0]  
new\_salaries = [zammer(a) if a > 3000 else a \* 1.3 for a in salaries] #çıktı: [1300.0, 3900.0, 6000.0, 8400.0, 10800.0]  
#Dikkat: Eğer iki koşul var ise for döngüsü en sağa geçer. tek koşul var ise if en sağa geçer!  
students = ["John", "Mark", "Venessa", "Mariam"]  
no\_students = ["John", "Venessa"]  
k = [x.upper() if x not in no\_students else x.lower() for x in students ]  
print(k)  
l = []  
for x in students:  
 if x in no\_students:  
 l.append(x.lower())  
 else:  
 l.append(x.upper())  
 print(l)  
  
#dictionary comprehensions:  
sozluk = {'a': 1,  
 'b': 2,  
 'c': 3,   
 'd':4} # kod okunması açısından alt satırlara geçtik

sozluk.keys()  
sozluk.values()  
sozluk.items() #key ve value değerleri tupple ikilisi olarkak liste formatında geliyor  
sozluk.items() # çıktı:dict\_items([('a', 1), ('b', 2), ('c', 3), ('d', 4)])  
yenisozluk = {k: v \*\* 2 for (k, v) in sozluk.items()}  
yenisozluk = {k.upper(): v \*\* 3 for (k, v) in sozluk.items()}  
  
###Uygulama-Mülakat Sorusu - Çift sayıların karesi alınarak bir sözlüğe eklenmek istenmektedir  
## çift sayılardan key kıssımları işlem görmemiş kısımlar, value kısımları ise karesi olan kısımlar olacak  
  
numbers1 = range(10)  
new\_dic={}  
  
for n in numbers1:  
 if n % 2 == 0:  
 new\_dic[n] = n \*\*2  
new\_dic2={}  
ab = {n: n \*\* 2 for n in numbers1 if n % 2 == 0} #burada dictionary comprehension kısmında anlaşılıyor ki dictionary.items  
#ile işlem şart değil!  
  
###########  
### List & Dict Comprehension Uygulamalar  
###########  
  
# Bir veri setinin değişken isimlerini değiştirmek  
  
#before: ['total', 'speeding', 'alcohol', 'not\_distracted', 'no\_previous', 'ins\_premium', 'ins,losses', 'abbrev']  
#istenen: yukarıdaki listedeki elemanların hepsi buyuk olacak!  
kucuk = ['total', 'speeding', 'alcohol', 'not\_distracted', 'no\_previous', 'ins\_premium', 'ins,losses', 'abbrev']  
  
buyuk = [n.upper() for n in kucuk]  
import seaborn as sns #seaborn kütüphanes sns olarak aktarıldı. öncesinde terminalden ilgili kütüphane kuruldu'  
df = sns.load\_dataset("car\_crashes") #burdaki df dataframe (veri çerçevesi kısaltılması)  
#burdaki df içinde excel tablosu gibi veri seti bulunur. sütün ve satırlardan oluşur  
df.columns #sütun isimlerini alıyoruz  
  
df.columns = [col.upper() for col in df.columns]  
#comrehension olmadan df.columns büyütelim  
a = []  
for col in df.columns:  
 a.append(col.upper())  
df.columns = a  
  
#isminde "INS" geçen değişkenlerin başına FLAG diğerlerine NO\_FLAG ekleyelim  
df = sns.load\_dataset("car\_crashes")  
df.columns = ["FLAG\_" + col.upper() if "INS" in col.upper() else "NO\_FLAG\_" + col.upper() for col in df.columns]  
[col.upper() for col in df.columns if "INS" in col.upper()] #burada sadece isminde INS geçenleri ayırdık  
#sadece sayısal değişkenler için key'i string value'su liste olan bir sözlük oluşturmak  
#istenen aşağıdaki gibi bir sözlük  
#{'total': ['mean', 'min', 'max', 'var'],  
# 'speeding': ['mean', 'min', 'max', 'var'],  
# 'alcohol': ['mean', 'min', 'max', 'var'],  
# 'not\_distracted': ['mean', 'min', 'max', 'var'],  
# 'no\_previous': ['mean', 'min', 'max', 'var'],  
# 'ins\_premium': ['mean', 'min', 'max', 'var'],  
# 'ins,losses': ['mean', 'min', 'max', 'var'],  
# 'abbrev': ['mean', 'min', 'max', 'var']}  
  
  
  
agg\_list = ['mean', 'min', 'max', 'var']  
  
import seaborn as sns  
df = sns.load\_dataset("car\_crashes")  
df.columns  
  
num\_cols = [col for col in df.columns if df[col].dtype != "O"] #object data tipi kategorik değişken(sayısal olmayan) olduğunu ifade ediyor. object olmayan  
 #kolonları seçiyoruz!. sql mntığı ile df[col] kullandık.  
  
########################## eski usül#############  
a\_dict = {}  
for col in num\_cols:  
 a\_dict[col] = agg\_list  
############### comprehension ile ###############  
new\_dict = {col: agg\_list for col in num\_cols}  
#################################################  
df.head() # default olarak ilk 5 kaydı getiriyor  
df[num\_cols] #yukarıda seçilen num\_cols'daki sayısal ifade içeren kolonları alıyoruz  
df[num\_cols].head() # yukarıda seçilen num\_cols'daki sayısal ifade içeren kolonların ilk 5 kaydını alıyoruz  
####aşağıdaki çok önemli bir durum  
df[num\_cols].agg(new\_dict) #aggragation fonksiyonu(agg) içine yazılan sözlükteki key değerler ile dataframe'deki kolon  
 #isimlerini eşleeyrek ilgili kolonlara value kısmındaki aggragate fonksiyonları uyguluyor!

# NUMPY KÜTÜPHANESİ

#numpy:çok boyutlu arrayler, matrisler ve diğer bilimsel hesaplamalar için yüksek verimli çalışmak amacıyla kullanılır.  
#1. verimil veri saklamada önceliklidir  
#2. yüksek seviyede işlemler için kullanışlı ve performanslıdır.  
# daha az çabayla daha fazla işlem yapmaya olanak verir  
  
##neden Numpy:  
a = [1, 2, 3, 4]  
b = [2, 3, 4, 5]  
c = []  
for i in range(0,len(a)):  
 c.append(a[i] \* b[i])  
import numpy as np  
a = np.array([1, 2, 3, 4])  
b= np.array([2, 3, 4, 5])  
c = a \* b #çıktı: array([ 2, 6, 12, 20])  
#yularıdaki c dizisi numpy ile çok daha kolay, append ve döngüle gerek olmadan, elde ediliyor!  
# Numpy sabit tipte, verimli yolla, veri tutar. Bu yüzden hızlıdır.!  
# Yüksek seviyede, vektörel işlem yapılmasına olanak sağlar! daha az çabayla daha fazla işlem!  
# array = vektör !  
######################## Numpy Array Oluşturma #################  
#numpy array = ndarray(n dimension array)  
  
np.array([ 2, 6, 12, 20]) # bu şekilde dizi oluşturmak için dizinin köşeli paranterz içinde yazılması gerek.  
x = np.zeros(10,dtype=int) #istenilen sayıda 0 oluşturmayı sağlar.  
np.random.randint(1,9,size=6) #rastgele olarak 1'den 9'a kadar 6 elemanlı dizi oluşturur!  
np.random.normal(10, 5, (3, 4)) # ortalaması 10 ,standart sapması 5 olan 3 satır 4 sütun matris oluşturur  
  
################### Numpy Array Özellikleri #############  
  
  
y = np.random.randint(10, size=7) # 0-10 arası 7 elemanlı array  
y.ndim #dimension: boyut saysısı bilgisi. tek boyutlu bir dizi  
y.shape # boyut bilgisisini detaylı veriyor. çıktı: (7, ) = 7 elemanlı tek boyutlu dizi  
y.size # toplam eleman sayısını veriyor  
  
k = np.random.normal(8,3, (3,4,2)) # 3 ana elamanı olan, bu 3 ana elaman 4'er alt eleman içeren ve bu 4 alt elemanın içinde  
 # de 2'şer tane elaman olan 3 boyutlu dizi  
k.shape #çıktı: (3, 4, 2)  
k.dtype #çktı: float64  
  
######## Reshaping #####  
  
#elimizdeki diziyi tekrar bpyutladırma istiyoruz  
  
np.random.randint(0, 12, 9)  
np.random.randint(0, 12, 9).reshape(3,3)  
ar = np.random.randint(0, 12, 9)  
ar.reshape(3,3) #ar dizisi 3 x 3 diziye(matrise) çeviriliyor!  
  
####### Index Seçimi ########  
  
ar2 = np.random.randint(0,10,size=10)  
ar2[0:5] #0-5 arası index'teki değerleri alıyor. 5 index değeri hariç!!!  
  
z = np.random.randint(10,size=(3, 5))  
z[0, 2] #iki boyutlu dizilerde index seçerken satır ve sütunun indexini belirtmek gerekiyor.  
  
 z[:, 1] #slicing yapılarak : ile bütün satırlar seçildi. 1 ile de 1. sütunlar seçildi.  
  
 z[0:2, 1:5] #satırlardan 0-2 arasını, sütunlardan 1-5 arasını seçtik. 2 ve 5 dahil değil!  
  
 ########  
 ## Fancy Index  
 #########  
 v = np.arange(0, 30 ,3) #arange fonksiyonu belirli aralıklar ile dizi oluşturmada kullanılır. burada 3 er artan dizi oluşuyor  
 #örneği elimizde ana listede seçmek istediğimiz elemanların index numaraları olsun. bu index'ler python listesinde tutularak  
 #numpy array i içine bu index listesi girildiğinde, python listesi içindeki index değerlerine göre elemanları getiriyor!  
 indis\_listesi = [0, 1, 5]  
 v[indis\_listesi] # çıktı: array([ 0, 3, 15])  
  
################ Numpy'da Koşullu İşlemler ######  
#klasik yöntem ile  
m = np.random.randint(19, size=6) #çıktı: array([11, 7, 4, 1, 13, 3])  
n = []  
for i in m:  
 if i < 6:  
 n.append(i)  
print(n) #çıktı: [4, 1, 3]  
  
#Numpy ile  
  
m < 6 #çıktı: array([False, False, True, True, False, True])  
m[m < 6] #çıktı:array([4, 1, 3])  
#Önemli Not: Köşeli parantez eleman seçileceği anlamına geliyor! Yukarıda m < 6 sonucundan true olan değerleri alıyor!  
# m < 6 ifadesi ile ortya çıkan true, false dizisi fancy index yöntemiyle seçim yapmak için kullanılıyor!  
  
### Numpy Array'lerde matematiksel işlemler  
#Not: Numpy dizilerinde sadece dizinin ismi kullanılarka bütün elemanlara işlem yapılabilir! Döngüden çok daha hızlı  
dizi = np.array([1, 2, 3, 4, 5, 6])  
  
dizi / 5 # çıktı: array([0.2, 0.4, 0.6, 0.8, 1. , 1.2])  
dizi \* 5 # çıktı: array([ 5, 10, 15, 20, 25, 30])  
dizi \*\* 2 # çıktı: array([ 1, 4, 9, 16, 25, 36])  
dizi - 1 # çıktı: array([0, 1, 2, 3, 4, 5])  
dizi + 1 # çıktı: array([2, 3, 4, 5, 6, 7])  
np.mean(dizi) # ortalama  
np.var(dizi) #varyans: standart sapmanın karesi. çıktı: 2.9166666666666665  
np.sum(dizi) #tüm dizi elemanlarını topluyor. dizi.sum() ile aynı. miuul'da dizi.sum() anlatılmıyor  
dizi.sum() # tüm elemanları topluyor  
  
## Numpy ile iki bilinmeyenli denklen Çözümü  
  
# 5 \* x0 + -3 \* x1 = 12  
# x0 + 3 \* x1 = 10  
  
t = np.array([[5, -3], [1, 3]])  
p = np.array([12, 10])  
np.linalg.solve(t, p) #çıktı: array([3.66666667, 2.11111111])  
# Numpy kütüphanesine yukarıdaki bilinmeyenlerin katsayıları ve sonuçlar dizi olarak gönderildiğinde linalg metoduyla  
#sonuc elde edilir  
  
#NEden Numpy: kısaca Hızlı ve Kolay / Kullanışlı

# PANDAS KÜTÜPHANESİ

# pandas serisi : TEK BOYUTLU index içerin veri tipi.  
#pandas kütüphanesi: çok boyutlu inddex içeren VERİ ÇERÇEVESİ(EXCEL TABLOSU İKİ BOYUTLU BİR PANDAS DATAFRAE OLUYOR!)  
import pandas as pd  
  
x = pd.Series([10,40,50,100])  
type(x)  
x.values # veri tipi numpy array'idir. DİKKAT: INDEX DEĞERİ HARİÇ TUTULDUĞUNDA PANDAS KÜTÜHPANESİNDE HER SÜTUN BİR NUMPY ARRAY'İDİR!  
type(x.values)  
x.index # çıktı: RangeIndex(start=0, stop=4, step=1)  
x.dtype  
x.size # eleman sayısına erişiyoruz!  
x.ndim  
x.shape  
x.head() # head metodu, default olarak, seri ve dataframe içindeki ilk 5 metodu getirir.  
x.tail()# sondan 5 değeri default olrak getiriyor!  
  
############## VERİ OKUMA (DİKKATTTTT) EXCEL DOSYLARINI OKUMAK İÇİN ÖNEMLİ  
  
df = pd.read\_excel("python for excel.xlsx") # excel dosyasını okutmak için excel dosyası ilgili python klasörünün içine atıldı!  
  
df.head()  
df.index  
  
##############Veriye Hızlı Bakış  
  
import seaborn as sns  
df = sns.load\_dataset("titanic")  
df.head()  
df.tail()  
df.shape # çıktı: (891, 15) 891 satıri 15 sütun  
df.head().info() #(çıktı: sütunları, verinin boş olup olmadığını ve dtype değerini verioyr  
 # Column Non-Null Count Dtype  
#--- ------ -------------- -----  
 #0 survived 891 non-null int64  
 #1 pclass 891 non-null int64  
 #2 sex 891 non-null object  
 #3 age 714 non-null float64  
 #4 sibsp 891 non-null int64  
#dtypes: bool(2), category(2), float64(2), int64(4), object(5) )  
  
#pandas veritipinde category ve object değişkenleri kategorik değişkenlerdir(sayısal olmayan, string!)  
df.columns # sütun isimleri, yani değişken isimlerini alıyoruz!!  
df.describe() # veri setindeki sayısal verinin hızlı/betimsel istatiklerine(ortalama, medayna vs..) hemen ulaşılıyor!  
df.describe().T #buradaki T ile transpose alınıyor!  
  
df.isnull().values.any() # veri setinde herhangi bir boş değer var mı ögrenmek için bunu kullanıyoruz!  
df.isnull().sum() #hangi değişkende(sütunda) kaç tane boş değer var onu buluyoruz !!  
df["sex"].value\_counts() # cinsiyet değişkenine ait alt sınıflarına ve bu sınıflarda kaç tane elaman olduğuna ulaşıyoruz  
 #çıktı:  
 #male 577  
 #female 314  
 #Name: sex, dtype: int64  
  
######################### PANDAS SEÇİM İŞLEMLERİ ########################################  
  
df.index  
df[0:13] # index numaraları 0-13 arası, 13 hariç, 13 adet satır seçildi!  
df.drop(1,axis=0).head() # axis=0 satırdan axis=1 sütundan silmeyi ifade ediyor!. 1 ise satir index numarsaıdır! işlem  
 #burada işlem kalıcı değişdir.  
#drop ile silme işleminin kalıcı olması için 2 yol var!  
#1  
df = df.drop(1, axis=0)  
#2  
df.drop(2, axis=0, inplace=True)  
df.head()  
#not: satir index i yerine liste içinde birden fazla index yazılabilir!  
###### INDEX İ DEĞİŞKEN, DEĞİŞKEN İ INDEX YAPMAK ! ÇOK ÖNEMLİ BİR KONUDUR !  
df = sns.load\_dataset("titanic")  
df.index = df["age"]  
df.drop("age", axis=1, inplace=True)  
df["age"] = df.index # age değişkenini yuarıda sildik. eğer seçilen sütun(değişken) dataframe'de yok ise o değişken dataframe'de oluşturulur!  
 #burada age değişkeni baştan oluşturuluyor!!  
df.head()  
  
df.reset\_index() # index değerinde olan "age" değişkenini yeni sütuna(değiişkene) atıyor ve index i sıfırlıyor!

###Değişkenler Üzerinde İşlemler #####  
import seaborn as sns  
import pandas as pd  
df = sns.load\_dataset("titanic")  
pd.set\_option("display.max\_columns", None) # python consele'daki çıktıda tüm sütunları görmek için bir ayar set ediyoruz!  
df.head()  
  
"age" in df #"age" değişkeni dataframe'de var mı ?  
######çok çok çok önemli!!!!!!  
df["age"] # type(df["age"].head()) = pandas series  
df[["age"]] #type(df[["age"]].head()) = pandas dataframe !!!!! çift köşeli parantez ile tek sütn seçilirse dataframe oluyor!!!  
####################################################  
df[["age", "alive"]].head() # birden fazla değişken(sütun ismi) seçilebilir!  
col\_names = ["age", "alive", "embarked"]  
df[col\_names].head() # listeyi temsil eden bir değişken ile de sütun isimleri seçilebilir.  
###NOT: FANCY INDEX KAVRAMI DATAFRAM'DE GEÇERLİDİR. ÇÜNKÜ PANDAS DATAFRAME'DE HER SÜTUN INDEX OLMAYAN HALİYLE BİR NUMPY ARRAY'İDİR.  
df["age2"] = df["age"] \*\* 2 #Datafram'e bu şekilde yeni bir değişken(sütun) eklenebilir!  
df["age3"] = df["age"] / df["age2"] # tekrar yeni bir değişken eklendi  
df.drop("age3", axis=1).head() # axis=1 sütunları ifade eder! "age3" değişkeni(sütunu) tamamen silinmiştir. inplace=True denmediği için değişiklik kalıcı değildir  
df.drop(col\_names,axis=1).head() #birden fazla değişken de aynı şekilde silinebilir.  
df.head()  
#ÇOK ÖNELİ!! İÇİNDE BELİRLİ BİR ARGÜMAN İÇEREN DATAFRAME DEĞİŞKENİNİ SİLMEK İÇİN:  
#1. yol en klasik yol  
for x in df.columns:  
 if "age" in x:  
 df.drop(x, axis=1, inplace=True)#inpalce=True ifdadesi olmadan çalışmıyor!  
 print(df.head())  
  
df = sns.load\_dataset("titanic")  
#2. yol = sadece "age" siliyor. "age2" ve "age3" duruyor.  
A = [y for y in df.columns if y.find("age") != -1] #içerisinde age içermeyenleri alıyor!!!  
df.drop(A, axis=1).head()  
##EN ÖNEMLİ YOL!(loc: pandas dataframde label based seçim payılmasını sağlıyor!!)  
df.loc[:, df.columns.str.contains("age")].head() #sadece içerisinde "age" karakterlerini içeren "age", "age2","age3" alıyor!  
 # df.loc[:,] ile bütün satırlar seçilmiş oldu  
#yurıdaki kodun türkçesi: dataframe kolonlarına bir string operasyonu uyguluyoruz ve içerisinde "age" karakterlerini içerenleri alıyoruz!  
df.loc[:, ~df.columns.str.contains("age")].head() # ~ (tilda işareti) ile "age"karakterlerini içermeyenleri kastediyoruz!  
  
##################################  
#### loc(label based seçim) & iloc(integer based seçim)  
##################################  
#iloc:  
df[0:3] #0-3 indexleri arası, 3 hariç, satırları alıyor  
df[0:3, 1] # iki boyutlu seçim olmuyor ! bu ifade hata veriyor!  
#NOT: DEMEK Kİ DATAFRAME DEĞİŞKEN(SÜTUN) SEÇİMLERİ SADECE loc ve iloc İLE OLUYOR!!!!!!!!!  
df.iloc[0:3, 0:3] # not: integer based olduğundan mutlaka değişken(sütun) indexleri belirtilmelidir.! 3'e kadar seçimler yapılıyor, 3 hariç !!!  
df.iloc[0,0] # 0. satır, 0.sütun  
  
#loc:  
df.loc[0:3, "age"] #age değişkenindeki(sütunundaki) 0-3 index arası, 3 DAHİL!! seçildi. label based olduğundan ixdex in adına göre seçim yapılıyor!!!  
df.loc[0:3, col\_names] #birden fazla değişken(sütun) seçilebilir !  
#NOT: loc ile sütun isimleri, iloc ile sütun indexleri kullanılır. Aksi halde hata alınıyor!  
df["age"].head()  
#NOT: KÖŞELİ PARANTEZLER İLE YA SATIR YA DA SÜTUN SEÇİLİYOR. HEM SATIR HEM SÜTUN ÇİN loc ve iloc İFADELERİ GEREKİYOR  
  
type(df[df["age"] > 50][["age"]].head())

############## KOŞULLU SEÇİM İŞLEMLERİ ###################  
import seaborn as sns  
import pandas as pd  
pd.set\_option("display.max\_columns", None)  
df = sns.load\_dataset("titanic")  
  
df[df["age"] > 50].head() #yaşı 50'den büyük olanlara göre dataframe'i tüm değişkenleriye(sütunşarıyla) getiriyor!  
df[df["age"] > 50]["age"].head() # yaşı 50'den büyük olanlara göre sadece "age" değişkenini(sütununu) getiriyor!!!  
df[df["age"] > 50].count() #yaşı 50'den büyük olanlara göre dataframe'de tüm değişkenleri kendi içinnde ayrı ayrı saymış oldu!  
df[df["age"] > 50]["age"].count# yaşı 50'den büyük olanlara göre sadece "age" değişkenini saymış oldu!!!  
  
#yaşı 50'den büyük olanların class,embark\_town ve yaş bilgileri  
df.loc[df["age"] > 50, ["class", "age","embark\_town"]].head()#  
  
#yaşı 50'den büyük erkeklerin yaş ve sınıf bilfileri ulaşalım:  
df.loc[(df["age"] > 50) & (df["sex"] == "male"),["age", "class"]].head()  
#NOT: BİRDEN FAZLA KOŞUL GİRİLDİĞİNDE, İSTENİLEN KOŞULLAR PARANTEZ İÇİNDE BELİRTİLMELİDİR!!!  
  
#yaşı 50'den büyük olan erkeklerin gemiye bindiği liman Cherbourg olanların yaş, sınıf ve embark\_town bilgileri  
df.loc[(df["age"] > 50) & (df["sex"] == "male") & (df["embark\_town"] == "Cherbourg"),  
 ["age", "class", "embark\_town"]].head()  
#yaşı 50'den büyük, erkek, gemiye bindiği liman Cherbourg ya da Southampton olanların yaş, sınıf ve embark\_town bilgileri  
df.loc[(df["age"] > 50) & (df["sex"] == "male") &  
 ((df["embark\_town"] == "Cherbourg") | (df["embark\_town"] == "Southampton")),  
 ["age","class", "embark\_town"]].head()  
## python'da | sembolü (ya da) koşulunu belirtiyor!!!!  
  
df["embark\_town"].value\_counts() #çıktı:  
#Southampton 644  
#Cherbourg 168  
#Queenstown 77  
#Name: embark\_town, dtype: int64  
  
#######TOPLULAŞTIRMA VE GRUPLAŞTIRMA (AGGRAGATION & GROUPING)  
# count(), first(), last(), mean(), median(), min(), max(), std(), var() - varyans, sum()  
df.head()  
df.shape  
#cinsiyete göre yaş ortalamasını bulalım.  
df.groupby("sex")["age"].mean() #dataframe yaşa göre gruplandırıldı, "age" değişkeni+ seçilerek ortalaması alındı  
df.groupby("embark\_town").count()  
#agg fonksiyonu ile kullanım önemlidir. agg ile aynı jdeğişkene(sütuna) birden fazla aggregetaion fonksiyonu uygulanır!  
  
df.groupby("sex").agg({"age":"mean"}) # df değişkeni cinsiyete göre kırıldı/sınıflandırıldı, yaş değişkeninin ortalaması alındı.  
df.groupby("sex").agg({"age": ["mean", "median", "sum"]})  
df.groupby("sex").agg({"age": ["mean", "median", "sum","count"],  
 "embark\_town" : "count"  
 })  
#not: yukarıda groupby yapılınca embark\_town değişkeninin sayısı cinsiyet kırılımına göre verilmiş oluyor. yani hangi limandan  
#kimin kaç kadın kaç erkek bindiği belli değil. İşte tam bu noktada pivot table özelliği devreye girmektedir!!!  
  
df.groupby("sex").agg({"age": ["mean", "sum"],  
 "survived": "mean"  
 })  
df.groupby(["sex", "embark\_town"]).agg({"age":["mean", "sum"],  
 "survived":"mean"  
 }) #bu kod ile veri önce cinsiyete göre sonra d limanlara göre kırılmış.  
#yani kadınların bindikleri limanlara göre yaş ortalamsı, toplamı ve kurtulma durumlarının ortalaması alınmıştır.  
#NOT: VERİYİ KIRAR İKEN KIRILMAK İSTENEN DEĞİŞKENLERİN SIRALAMASI ÖENMLİDİR.!!!  
df.groupby(["sex", "embark\_town", "class"]).agg({"age": "mean",  
 "survived" : "mean",  
 "sex" : "count"  
 })

##### PIVOT TABLE #####  
# veriyi istenilen değişkene(sütuna) göre kırar !  
#df.pivot\_table("values, yani kesişimlerde görmek istediğin değerler/sonular", "satırlarda görmek istediğin değerler", "sütunlarda görmek istediğim değerler")  
#df.pivot\_table("values, yani kesişimlerde görmek istediğin değerler/sonular", "satırlarda görmek istediğin değerler", "sütunlarda görmek istediğim değerler", aggfunc="istenlen aggregate fonksiyon")  
#NOT: PİVOT TABLE AGGREGAATE FONKSİYON OLARAK DEFAULT BİÇİMDE ORTALAMA(MEAN) İLE TANIMLIDIR  
df.pivot\_table("age","sex", "embark\_town") # age değişkeninin ortalaması alındı. defaul oarak aggfunc="mean"  
#çıktı: embark\_town Cherbourg Queenstown Southampton  
# sex  
# female 28.344262 24.291667 27.771505  
# male 32.998841 30.937500 30.291440  
df.pivot\_table(["age"],"sex",["embarked", "class"])  
df.head()  
df.groupby(["sex", "embarked","class"]).agg({"age" : "mean"  
 })  
#ÇOK ÖNEMLİ NOT1: YAŞ DEĞİŞKENİNE GÖRE SINIFLANDIRMA YAPMAK İÇİN YAŞ DEĞİŞKENİNİ KATEGORİK BİR DEĞİŞKENE ÇEVİRMEK GEREKİYOR!!(cut ve qcut fonksiyonları ile yapılıyor)  
#ÇOK ÖENMLİ NOT2: EĞER KATEGORİK DEĞİŞKENE ÇEVİRELECEK SAYISAL DEĞİŞKEN TANINIYORSA(ÖRNEĞİN YAŞ DEĞİŞKENİNDE AHNGİ YAŞ ARALIĞININ GENÇ-YAŞLI OLDUĞU BİLİNİYOR!) cut FONKSİYONU KULLANILIR!  
#EĞER KATEGORİK DEĞİŞKENE ÇEVİRELECEK SAYISAL DEĞİŞKEN TANINMIYORSA,BU SAYISAL DEĞİŞKENİ KÜÇÜKTEN BÜYÜĞE SIRALAYIP YÜZDELİK ÇEYREK DEĞERLERİNE BÖLECEK OLAN qcut FONKSİYONU KULLANILIR  
##### cut ve qcut FONKSİYONLARI ÇOK YAYGIN KULLANILIR!!!!!  
df["new\_age"] = pd.cut("ne bölünecek", "bölünecek olan nereden bölünecek")  
df["new\_age"] = pd.cut(df["age"], [0, 10, 18, 25, 40, 90]) #not: burada aynı zamanda yeni bir değişken olarak new\_age sütunu da eklenmiş oldu  
df.head()  
df.pivot\_table("survived",["sex", "new\_age"],["class", "embarked"])  
df.pivot\_table("survived","sex",["class", "new\_age"])  
  
  
###################  
#### çıktı genişletme ayarı : pd.set\_option("display.width", 500)  
###############  
pd.set\_option("display.width", 500)  
  
df.groupby(["sex", "new\_age", "class"]).agg({"survived" : "mean"})  
###################  
#### apply & lambda  
###################  
df = sns.load\_dataset("titanic")  
pd.set\_option("display.max\_columns", None)  
pd.set\_option("display.width", 500)  
df["age2"] = df["age"] \* 2  
df["age3"] = df["age"] \* 5  
df.head()  
df["age"]  
####### APPLY FONKSİYONU PANDAS KÜTÜPHANESİNE AİT BİR FONKSİYONDUR. DÖNGÜYE GEREK OLMADAN, BİR DEĞİŞKENE(SÜTUNA) YAPILACAK OLAN İŞLEMİ +O DEĞİŞKENİN ELEMANLARINA  
####### BİRER BİRER UYGULANMASINI SAĞLIYOR!!!!  
#İÇİNDE "age" ifadesi geçen değişkenleri 10'a bölelim!!!  
  
  
df.loc[:,df.columns.str.contains("age")].head()  
df.loc[:,df.columns.str.contains("age")].apply(lambda x: x/10).head()  
  
# NOT: apply fonksiyonu SATIR VE DE SÜTUNLARA UYGULANABİLİR! DEFAULT OLARAK axis=0'dır  
def myfonksiyon(x):  
 y = (x-1)/x.std()  
 return y  
  
df.loc[:,df.columns.str.contains("age")].apply(lambda x: (x-x.mean())/x.std()).head()  
df.loc[:,df.columns.str.contains("age")].apply(lambda x: (x-x.mean())/x.std()).head()  
### ÇOK ÖNEMLİ NOT!!!!: lambda x: (x-x.mean())/x.std()) İFADESİNDEKİ X BU SEFER DEĞİŞKENİ(SÜTUNU) İFADE EDİYOR. YANİ HER DEĞİŞKENDEN O SÜTUNUN ORTALAMASI ÇIKARILIP  
## STANDART SAPMASINA BÖLÜNÜYIR!!!!!  
  
#APPLY İLE HERHANGİ TANIMLI BİR DONKSİYON KULLANILABİLİR!!!!  
def standart\_scaler(col\_name):  
 return (col\_name-col\_name.mean()) / col\_name.std()  
df.loc[:,df.columns.str.contains("age")].apply(standart\_scaler).head()

#PANDAS BİRLEŞTİRME(JOIN) İŞLEMLERİ !!!!  
import numpy as np  
import pandas as pd  
m = np.random.randint(1,30, size=(5, 3))  
df1 = pd.DataFrame(m, columns=["var1", "var2", "var3"])  
df2 = df1 + 99  
  
###### CONCAT İLE VERİ BİRLEŞTİRME######  
pd.concat([df1, df2]) #burada her iki dataframe'in satırları alt alta gelecek şekilde birleşiyor! default olarak axis=0. fakat bu noktada indexler de birleşmiş oluyor ki, aynı değere ship birden fazla index olabilioyr!!!  
pd.concat([df1, df2], ignore\_index=True) #burada dataframe'in index değerleri ignore edilerek aynı index'li değerlerin sonu. dataframe'inde olmaması sağlanıyor!  
## Merge ile Birleştirme Yöntemi ####  
df1 = pd.DataFrame({"employees" : ["john", "dennis", "mark", "mariam"],  
 "group" : ["accounting", "engineering", "engineering", "hr"]})  
df2 = pd.DataFrame({"employees" : ["john", "dennis", "mark", "mariam"],  
 "start\_date": [2010, 2009, 2014, 2019]})  
  
df3 = pd.merge(df1, df2, on="employees") # SQL INNER JOIN gibi. default olarak KESİŞİME GÖRE BİRLEŞTİRİYOR. left ve rıght joın seçenekleri de mevcut. left on / right on  
  
df4 = pd.DataFrame({"group" : ["accounting", "engineering", "hr"],  
 "namager" : ["Caner", "Mustafa", "Berkcan"]})  
pd.merge(df3, df4, on="group")

NOT: BİR DEĞİŞKENİN TİPİNİ DĞEİŞTİRMEK İÇİN astype kullanılır.

df["embarked"] = df["embarked"].astype("category") #burada embarked değişkenine yeniden atama yapılarak değişken tipi kategori olarak düzenlenmiştir!

# VERİ GÖRSELLEŞTİRME (MATPLOTLIB & SEABORN KÜTÜPHANELERİ)

###Matplotlib: DÜşük sevieli veri görselleştirme aracıdır. python'da veri görselleştirme araçlarınınn atasıdır!!!  
  
  
###ÇOK ÖNEMLİ NOT:  
# Kategorik Değişkenler: Sütun Grafik ile gösterilmeli. Seaborn:countplot, matplot:barplot  
# Sayısal Değişkenler: histogram(hist:sadece dağılım gösterir), boxplot(aykırı değerleri de göstermektedir)  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
pd.set\_option("display.max\_columns", None)  
pd.set\_option("display.width", 500)  
  
df = sns.load\_dataset("titanic")  
df.head()  
df['sex'].value\_counts().plot(kind='bar')  
plt.show(block=True)  
  
##### SAyısal Değişken Görrselleştirme  
plt.hist(df["age"])  
plt.show(block=True)  
plt.boxplot(df["fare"])  
plt.show(block=True)  
  
##NOT: BİR VERİYİ TANIRKEN ESAS AMAÇ DEĞİŞKENLERİN YAPISINI GÖZLEMLEYEBİLMEKTİDİR. DOLAYISYLA BOXPLOT VE HİSTOGRAM GRAFİKLER İSTATİSTİK GRAFİKLERİ OLUP  
## SAYISAL DEĞİŞKENLERİN GÖRSELLEŞTİRİLMESİNDE ÖNEMLİDİR!!!

### MATPLOTLIB ÖZELLİKLERİ  
import numpy as np  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
pd.set\_option("display.max\_columns", None)  
pd.set\_option("display.width",500)  
  
#### plot özellliği  
x = np.array([1, 8])  
y = np.array(([0, 150]))  
plt.plot(x, y, "o") #burada "o" ile grafikte gösterilecek kesişim noklatlarının marker'ını belirtiyoruz!  
plt.show(block=True)  
  
###marker  
y = np.array([13, 28, 11, 100])  
plt.plot(y,marker="o") #y dizisindeki noktalar içi dolu o işaretiyle gösteriliyor  
plt.show(block=True)  
  
plt.plot(y,marker="\*") #y dizisindeki noktalar \* işaretiyel gösteriliyor  
plt.show(block=True)  
  
# kullanılacak marker türlerine https://www.w3schools.com/python/matplotlib\_markers.asp adresinden ulaşabilriz  
  
## line  
y = np.array([13, 28, 11, 100])  
  
plt.plot(y, linestyle="dashed") #kesik çizgili grafik  
plt.plot(y, linestyle="dotted") #nokta çizgili grafik  
plt.plot(y, linestyle="dashdot") #kesik çigi ve noktalı grafik  
plt.plot(y, linestyle="dashdot", color="r") #kesik çigi ile noktalı ve kırmızı grafik  
  
plt.show(block=True)  
  
# Multiple lines  
x = np.array([23, 2, 28, 50])  
y = np.array([13, 28, 11, 100])  
plt.plot(x)  
plt.plot(y)  
plt.show(block=True)  
#yukrıdaki x ve y numpay dizileri aynı plot üzerine farklı renklerde grafiğe yazdırılıyor!  
  
  
#Labels (Grafik ögelerine başlıklar, etiketler verme)  
x = np.array([23, 2, 28, 50])  
plt.plot(x, color="b")  
plt.title("Bu Ana Başlık")  
plt.xlabel("X ekeseni isimlendirmesi")  
plt.ylabel("y ekseni isimlendirmesi")  
plt.grid() # grafiğe okubalirliği artırmak amacıyla ızgara eklemek için!!!  
plt.show(block=True)  
  
#Subplots  
x= np.array([80, 85, 90, 95, 100, 105, 110, 115])  
y= np.array([400, 425, 450, 475, 500, 525, 550, 575])  
#plot1  
plt.subplot(1, 2, 1) #subplot ile 1 satırda, 2 adet grafik oluşturuldu. ve 1. grafik belirtildi.  
plt.title("1")  
plt.plot(x, y)  
#plot2  
z = y \* 5  
k = x \* 3  
  
plt.subplot(1, 2, 2) #subplot ile 1 satırda, 2 adet grafik oluşturuldu. ve 2. grafik belirtildi  
plt.title("2")  
plt.plot(z, k)  
plt.show(block=True)  
  
  
#########  
### SEABORN İLE VERİ GÖRSELLEŞTİRME  
######  
import pandas as pd  
import seaborn as sns  
from matplotlib import pyplot as plt  
df = sns.load\_dataset("tips")  
df.head()  
df["sex"].value\_counts() #sex değişkeininde kategorilerin frekanslarına ulaştık !  
sns.countplot(data=df, x=df["sex"]) #barplot ile aynı. kullanırken x ekseni ile veri seti mutlaka tanımlanmalı!!  
plt.show(block=True)  
  
sns.boxplot(data=df, x=df["total\_bill"])  
plt.show(block=True)  
df["total\_bill"].hist() # pandas içinde bulunan histogram grafiği !!!  
plt.show(block=True)  
sns.histplot(df,x=df["total\_bill"])  
plt.show(block=True)

# GELİŞMİŞ FONKSİYONEL KEŞİFÇİ VERİ ANALİZİ

######## GENEL RESİM  
import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
pd.set\_option("display.max\_columns", None)  
pd.set\_option("display.width", 700)  
df = sns.load\_dataset("titanic")  
df.head()  
df.tail()  
df.shape  
df.describe().T # sayısal değişkenlerin betimsel istatistikleri. ortalama medyan  
  
df.columns  
df.index  
df.isnull().values.any()  
df.isnull().sum()  
df.info()  
df["deck"].dtype  
  
def check\_df(dataframe, head=5):  
 print("########################### Shape #################")  
 print(dataframe.shape)  
 print("########################### Types #################")  
 print(dataframe.dtypes) # pandas dataframe de dtype yerine dtypes kullanılmalı!!!  
 print("########################### Head #################")  
 print(dataframe.head(head))  
 print("########################### Tail #################")  
 print(dataframe.tail(head))  
 print("########################### NA #################")  
 print(dataframe.isnull().sum())  
 print("########################### Quantiles #################")  
 print(dataframe.describe([0, 0.05, 0.25, 0.95, 0.99, 1]).T) # burdaki describe ile verinin belirli yüzdeliklerine göre dağılımını elde ediyoruz  
check\_df(df)  
  
df = sns.load\_dataset("tips")  
check\_df(df)  
  
### KATEGORİK DEĞİŞKEN ANALİZİ ### ÇOK ÇOK ÖNEMLİ BÖLÜM !!!!  
df = sns.load\_dataset("titanic")  
df["embarked"].value\_counts()  
df["embarked"].unique() # benzersiz/eşsiz değerlerin isimleri -- numpy array  
df["embarked"].nunique() #benzersiz / eşsiz değerlerin sayısı  
  
cat\_cols = [col for col in df.columns if str(df[col].dtype) in ["object", "category", "bool"]] # içeriğinde kategori barındıran, sayısal olmayan değişkenler bulunuyor  
num\_but\_cat = [col for col in df.columns if df[col].nunique() < 10 and df[col].dtypes in ["int64", "float64", "int32", "float32"]] # sayısal olup da kategorik olan değişkenler bulunuyor. survived değişkeni kategorktir!  
#kardinelitesi yüksek değişken: ölçümlenemeyecek kadar fazla sınıfı olan değişkenlerdir!!  
cat\_but\_cardinal = [col for col in df.columns if df[col].nunique() >=20 and str(df[col].dtypes) in ["object", "category"]] # değişkenin tipi kategorik ya da object olup, 20'den fazla sınıfı olan değişkenleri buluyoruz !!! bunlar ölçümlenemeyecek sınıf sayısına ait olduğundan verimizin kategorik değişken tanımından çıkarılacak!  
cat\_cols = cat\_cols + num\_but\_cat # tüm kategorik değişkenleri tek bir satırda topluyoruz  
  
cat\_cols = [col for col in cat\_cols if col not in cat\_but\_cardinal] #eğer kardinal değişken olsaydı bu şekilde cat\_cols'a ayar vermek gerekiyordu!!  
  
def cat\_summary(dataframe, col\_name):   
 print(pd.DataFrame({col\_name: dataframe[col\_name].value\_counts(),  
 "RATIO": 100 \* dataframe[col\_name].value\_counts() / len(dataframe)}))  
 print("#####################################################################")  
for col in cat\_cols:  
 cat\_summary(df, col)  
#cat\_summary fonksiyonu elimizdeki dataframe'e ait değişkenlerin kategorik tiptekilerine ait frekans bilgilerini yüzdelik ve sayısal olarak yazdırmayı sağlıyor!

import numpy as np, seaborn as sns, pandas as pd, matplotlib.pyplot as plt  
pd.set\_option("display.max\_columns", None)  
pd.set\_option("display.width", 500)  
df = sns.load\_dataset("titanic")  
df["sex"].unique()  
df["sex"].nunique() # benzersiz değerler sayısı  
df.info()  
cat\_cols = [col for col in df.columns if str(df[col].dtypes) in ["object", "bool", "category"]]  
num\_but\_cat = [col for col in df.columns if df[col].dtypes in ["float64", "int64", "float32", "int32"] and df[col].nunique() < 10]  
cat\_but\_cardinal = [col for col in df.columns if str(df[col].dtypes) in ["object", "category"] and df[col].nunique() > 20]  
cat\_cols = cat\_cols + num\_but\_cat  
  
cat\_cols = [col for col in cat\_cols if col not in cat\_but\_cardinal]  
df[cat\_cols].nunique()  
df["survived"].value\_counts()  
100 \* df["survived"].value\_counts() / len(df) #not: len(df) dataframe in satır sayısını veriyor!  
  
  
def cat\_summary(dataframe, col\_name):  
 print(pd.DataFrame({col\_name: dataframe[col\_name].value\_counts(),  
 "Ratio": 100 \* dataframe[col\_name].value\_counts() / len(dataframe)}))  
 print("########################################################################")  
cat\_summary(df, "survived")  
  
for col in cat\_cols:  
 cat\_summary(df, col)  
  
def cat\_summary(dataframe, col\_name, plot=False):  
 print(pd.DataFrame({col\_name: dataframe[col\_name].value\_counts(),  
 "Ratio": 100 \* dataframe[col\_name].value\_counts() / len(dataframe)}))  
 print("########################################################################")  
 if plot == True:  
 sns.countplot(data=df, x=col\_name)  
 plt.show(block=True)  
  
 #plt.ylabel  
  
cat\_summary(df, "survived", plot=True)  
  
for col in cat\_cols:  
 if df[col].dtypes == "bool":  
 df[col] = df[col].astype(int)  
 cat\_summary(df, col, plot=True)  
 else:  
 cat\_summary(df,col, plot=True)  
  
num\_cols = [col for col in df.columns if df[col].dtypes in ["float64", "int64", "float32", "int32"]]  
num\_cols = [col for col in num\_cols if col not in cat\_cols] #cat\_cols = categorik değişkenler. içeriği sayısal olup kategorik olan değişkenlerden kurtulmak için yaptık.  
  
def num\_summary(dataframe, numerical\_col):  
 quantiles = [0.05, 0.1, 0.2 , 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]  
 print(dataframe[numerical\_col].describe(quantiles).T)  
 print("############################################################")  
for col in num\_cols:  
 num\_summary(df, col)  
  
def num\_summary(dataframe, numerical\_col, plot=False):  
 quantiles = [0.05, 0.1, 0.2 , 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]  
 print(dataframe[numerical\_col].describe(quantiles).T)  
 print("############################################################")  
 if plot:  
 dataframe[numerical\_col].hist()  
 plt.xlabel(numerical\_col)  
 plt.title(numerical\_col)  
 plt.show(block=True)  
  
for col in num\_cols:  
 num\_summary(df, col, plot=True)  
  
  
df.head()  
df["fare"].value\_counts()  
# DEĞİŞKENLERİN YAKALANMASI KISMI. GENEL FONKSİYON YAZIMI  
def grab\_col\_names(dataframe, cat\_th=10, car\_th=20):  
#cat\_th: categorik threshold değeri: değişkenlerin(sütunların) kategorik sayılması için en fazla kaç farklı değeri(nunique) olması gerektiği.  
#car\_th: cardinaş threshold değeri: kardinalitesi(sınıf sayısı ölçülemeycek kadar farklı veriye sahip) yüksek kategorik değişkenlerin en az kaç kaç farklı sınıfı sayısı(nunique) olması gerektiği  
  
  
def grab\_col\_names(dataframe, cat\_th=10, car\_th=20):  
 *"""  
 Veri setindeki kategorik, numerik ve kategorik fakat kardinal değişekenlerin isimlerini verir  
 Parameters  
 ----------  
 dataframe: dataframe  
 değişken isimleri alınmak istenen dataframe'dir.  
 cat\_th: int, float  
 \*numerik fakat kategorik lan değişkenler için sınıf eşik değeri. Yani değişkenlerin(sütunların)  
 kategorik sayılması için en fazla kaç farklı değeri(nunique) olması gerektiği.  
 car\_th: int, float  
 kardinalitesi(sınıf sayısı ölçülemeycek kadar farklı veriye sahip) yüksek kategorik değişkenlerin  
 en az kaç kaç farklı sınıfı sayısı(nunique) olması gerektiği  
 \*kategorik fakat kardinal değişkenler için sınıf eşik değeri  
 Returns  
 -------  
 cal\_cols: list  
 Kategorik değişken listesi  
 num\_cols: list  
 Numerik değişken listesi  
 cat\_but\_car: list  
 Kategorik görümlü kardinal değişken listesi # kategorik değişken: sayısal olmayan değişkenler  
  
 Notes:  
 -----  
 cat\_cols + num\_cols + cat\_but\_car = toplam değişken sayısı  
 num\_but\_cat(numerik tipte fakat kategorik değişkenler) cat\_cols'un içinde  
 """* cat\_cols = [col for col in dataframe.columns if str(dataframe[col].dtypes) in ["object", "bool", "category"]]  
 num\_but\_cat = [col for col in dataframe.columns if dataframe[col].nunique() < cat\_th and dataframe[col].dtypes in ["float64", "int64", "float32", "int32"]]  
 cat\_but\_car = [col for col in dataframe.columns if str(dataframe[col].dtypes) in ["object", "category"] and dataframe[col].nunique() > car\_th ]  
 cat\_cols = cat\_cols + num\_but\_cat  
 cat\_cols = [col for col in cat\_cols if col not in cat\_but\_car]  
  
 num\_cols = [col for col in dataframe.columns if dataframe[col].dtypes in ["float64", "int64", "float32", "int32"]]  
 num\_cols = [col for col in num\_cols if col not in cat\_cols]  
  
 print(f"Observations : {dataframe.shape[0]}") #satır sayısı gözlem syısı  
 print(f"Variables : {dataframe.shape[1]}") #sütun sayısı: değişekn sayısı  
 print(f"cat\_cols : {len(cat\_cols)}")  
 print(f"num\_cols : {len(num\_cols)}")  
 print(f"cat\_but\_car : {len(cat\_but\_car)}")  
 print(f"num\_but\_cat : {len(num\_but\_cat)}")  
 return cat\_cols, num\_cols, cat\_but\_car  
cat\_cols, num\_cols, cat\_but\_cardinal = grab\_col\_names(df)

############ CAT SUMMURY FONKSİYONU  
def cat\_summary(dataframe, col\_name, plot=False):  
 print(pd.DataFrame({col\_name: dataframe[col\_name].value\_counts(),  
 "Ratio": 100 \* dataframe[col\_name].value\_counts() / len(dataframe)}))  
 print("########################################################################")  
 if plot == True:  
 sns.countplot(data=df, x=col\_name)  
 plt.show(block=True)  
  
cat\_summary(df, "age")  
  
### NUM\_SUMMARY FONKSİYONU  
def num\_summary(dataframe, numerical\_col, plot=False):  
 quantiles = [0.05, 0.1, 0.2 , 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]  
 print(dataframe[numerical\_col].describe(quantiles).T)  
 print("############################################################")  
 if plot:  
 dataframe[numerical\_col].hist()  
 plt.xlabel(numerical\_col)  
 plt.title(numerical\_col)  
 plt.show(block=True)  
  
for col in cat\_cols:  
 cat\_summary(df, col)  
  
for col in num\_cols:  
 num\_summary(df, col)  
  
##BONUS: BOOL TİPTEKİ DEĞİŞKENLERİ İNT OLARAK DEĞİŞTİRME İŞLEMİ  
  
for col in df.columns:  
 if df[col].dtypes == "bool":  
 df[col] = df[col].astype("int")  
df["adult\_male"].dtypes  
  
## HEDEF DEĞİŞKENİN KATEGORİK DEĞİŞKENLER İLE ANALİZİ  
import pandas as pd, seaborn as sns, numpy as np, matplotlib.pyplot as plt  
pd.set\_option("display.max\_columns", None)  
  
pd.set\_option("display.width", 1000)  
df = sns.load\_dataset("titanic")  
df.groupby(["sex"]).agg({"survived": "mean"})  
df.head()  
  
def target\_summary\_with\_cat(dataframe, target, categorical\_col):  
 print(pd.DataFrame({"TARGET\_MEAN": dataframe.groupby(categorical\_col)[target].mean()}))  
##################################################################################  
###NOT: HEMEN AŞAĞIDAKİ FONKSİYONDA PD.DATAFRAME, GROUPBY VE AGG AYNI ANDA ÇALIŞMIYOR. ÇÜNKÜ GROUPBY VE AGG ZATEN BİR PANDAS DATAFRAME!  
def target\_summary\_with\_cat(dataframe, target, categorical\_col):  
 #print(pd.DataFrame({"TARGET\_MEAN": [dataframe.groupby(categorical\_col).agg({target:"mean"})]}))  
 print(dataframe.groupby(categorical\_col).agg({target:"mean"}))  
 print("####################################################")  
def target\_summary\_with\_cat(dataframe, target, categorical\_col):  
 print(dataframe.groupby(categorical\_col).agg({target:"mean"}))  
#######################################################################  
type(df.groupby("sex")["survived"].mean()) #pandas serisi  
type(df.groupby("sex").agg({"survived": "mean"})) #pandas dataframe !!!!!  
#################################################################################  
target\_summary\_with\_cat(df, "survived", "sex")  
for col in cat\_cols:  
 target\_summary\_with\_cat(df, "survived", col)  
  
df.groupby("survived").agg({"age":["max", "min","mean"]})  
### HEDEF DEĞİŞKENLERİN SAYISAL DEĞİŞKENLERLE İNCELENMESİ  
def target\_summary\_with\_num(dataframe, target, numerical\_col):  
 *"""  
  
 Parameters  
 ----------  
 dataframe: veriseti  
 target: hedef değişken  
 numerical\_col: sayısal kolon  
  
 Returns  
 -------  
 hedef değişkene göre gruplama yaparak yazılan sayısal değişkenin ortalamasını döndürür.  
 """* #print(pd.DataFrame({"TARGET\_MEAN": dataframe.groupby(target)[numerical\_col].mean()}))  
 print(dataframe.groupby(target).agg({numerical\_col:"mean"}))  
 print("####################################################")  
for col in num\_cols:  
 target\_summary\_with\_num(df,"survived", col)  
 help(target\_summary\_with\_num)  
  
  
#### KORELASYON ANALİZİ  
df = pd.read\_csv("breast\_cancer.csv")  
df.head()  
df.iloc[:,1:-1].head() # id sütununu almadık  
  
## YÜKSEK KORELASYONLU DĞEĞİŞKENLERİ YAKALAMK VE FONKSİYON YAZMAK. SADECE İTİYAÇ HALİNDE KULLANILACAK!!!  
## NOT: BİRBİRLERİYLE YÜKSEK KORELASYONLU DEĞİŞKENLER HER ZAMAN SİLİNECEK DİYE BİR DURUM YOK!  
  
num\_cols = [col for col in df.columns if df[col].dtypes in ["int64", "int32", "float32", "float64"]]  
corr = df[num\_cols].corr() # corr() -- bütün değişkenlerin birbirleriyle olan korelasyonunu hesaplar !!  
  
#KORELASYON: DEĞİŞKENLERİN BİRBİRLERİYLE OLAN İLİŞKİSİNİ İFADE EDEN, -1 İLE +1 ARASI DEĞER ALAN BİR İSTATİKSEL ÖÇÜM TÜRÜDÜR!  
# KORELASYON -1 YA DA +1 E YAKLAŞTIKÇA İKİ DEĞİŞKEN ARASINDAKİ İLİŞKİ KUVVETLENİR. İLİKŞİ POZİTİF İSE, BİR DEĞİŞKENİN DEĞERİ ARTARSA  
# DİĞER DEĞİKENİN DEĞERİ DE ARTAR. İLİŞKİ NEGATİF İSE BR DEĞİŞKENİN DEĞERİ ARTAR İSE DİĞERİNİNKİ AZALIR  
  
sns.set(rc={"figure.figsize":(12,12)})  
sns.heatmap(corr, cmap="RdBu") # değişkenlerin birbirleriye olan analizini görüntülemek için ısı haritası yapıyoruz!  
plt.show(block=True)  
  
# YÜKSEK KORELASYONLU DEĞİŞKENLERİN SİLİNMESİ  
# NOT: HER PROJEDE BU DEĞİŞKENLERİN SİLİNMEİSNE GEREK YOKTUR!  
corr\_matrix = df.corr().abs() # negatif korelasyonları, yazılacak fonksiyonlarda daha kolay işlem olması için, pozitif e çevirdik!  
## corr\_matrix MATRİSİNİN KÖŞEGENİ ÜZERİNDE KALAN DEĞERLER İLE KÖSEGEN ALTINDA KALAN DEĞERLER AYNI OLDUĞUNDAN, KÖŞEGENALTINDAKİ DEĞERLER NaN (YOK)  
## DEĞERİNE ÇEVİRİYORUZ!!  
upper\_triangle\_matrix = corr\_matrix.where(np.triu(np.ones(corr\_matrix.shape),k=1).astype(np.bool\_))  
drop\_list = [ col for col in upper\_triangle\_matrix.columns if any(upper\_triangle\_matrix[col]>0.90)]  
#upper\_triangle\_matrix dataframe'inin sütunlarında(değişkenlerinde) %90 dan büyük bir tane dahi gözlem/sayır var mı diye any kullanıyoruz!!  
corr\_matrix[drop\_list]  
df.drop(drop\_list,axis=1)  
df.shape  
def high\_correlated\_cols(dataframe, plot=False, corr\_th=0.90):  
 corr = dataframe.corr()  
 corr\_matrix = corr.abs()  
 upper\_triangle\_matrix = corr\_matrix.where(np.triu(np.ones(corr\_matrix.shape),k=1).astype(np.bool\_))  
 drop\_list = [ col for col in upper\_triangle\_matrix.columns if any(upper\_triangle\_matrix[col] > corr\_th)]  
 if plot:  
 import seaborn as sns  
 import matplotlib.pyplot as plt  
 sns.set(rc={"figure.figsize":(15,15)})  
 sns.heatmap(corr, cmap="RdBu") # değişkenlerin birbirleriye olan analizini görüntülemek için ısı haritası yapıyoruz!  
 plt.show(block=True)  
 return drop\_list  
  
drop\_list = high\_correlated\_cols(df)  
high\_correlated\_cols(df.drop(drop\_list,axis=1), plot=True)  
high\_correlated\_cols(df, plot=True)

COUNTPLOT - ÇOK ÖENMLİ !

def cat\_sumary(dataframe, col\_name, plot=False):  
 print(pd.DataFrame({col\_name : dataframe[col\_name].value\_counts(),  
 "Ratio" : dataframe[col\_name].value\_counts() \* 100 / len(dataframe[col\_name]) }))  
 print("######################################################")  
 if plot:  
 #dataframe[col\_name].value\_counts().plot(kind="bar") #--> pandas grafiği  
 sns.countplot(dataframe,x=dataframe[col\_name])  
 plt.title(col\_name)  
 plt.show(block=True)  
  
  
def cat\_sumary(dataframe, col\_name, plot=False):  
 print(pd.DataFrame({col\_name : dataframe[col\_name].value\_counts(),  
 "Ratio" : dataframe[col\_name].value\_counts() \* 100 / len(dataframe[col\_name]) }))  
 print("######################################################")  
 if plot:  
 #dataframe[col\_name].value\_counts().plot(kind="bar") #--> pandas grafiği  
 sns.countplot(dataframe,x=col\_name)  
 plt.title(col\_name)  
 plt.show(block=True)  
  
##### ÇOK ÇOK ÖENMLİ NOT(COUNTPLOT):   
#sns.countplot(dataframe,x=col\_name) ifadesi ile sns.countplot(dataframe,x=dataframe[col\_name]) ifadesi aynı değildir!!   
#x=dataframe[col\_name] ifadesi ile bool tipindeki değişkenler grafiğe çizdirilememektedir !

NOTLAR: LİSTELERDE BELLEK ÜZERİNDE ADRESLER REFERANS ALINIR. DOLAYISIYLA BİRBİRİNE EŞİTLENEN İKİ LİSTENİN BİRİNDE YAPILAN BİR DEĞİŞİKLİK DİĞER LİSTEYİ DE ETKİLER

numbers = [1, 3, 5, 7, 10]  
x, \*y, z = numbers  
print(x,y,z) # çıktı: 1 [3, 5, 7] 10  
# NOT: X İLK DEĞERİ SAYI OLARAK, Z SON DEĞERİ SAYI OLARAK VE Y İSE ARADA KALAN DEĞERLERİ LİSTE OLARAK İÇİNE ALIR

## UFUK AÇICI KODLAR:

res =['fizzbuzz' if i%3 == 0 and i%5 == 0 else 'fizz' if i%3 == 0 else 'buzz' if i%5 == 0 else i for i in range(1, 101)]  
for nums in res:  
 print(nums)

# ## TEKRAR KISIMLARI BURDAN BAŞLIYOR

NUMPY: NUMPY kütüphanesi fixed type veri tutar. Yani sabit tipte veri sakladığından hızlı çalışır. Matris içindeki değerlerin hepsinin türü aynıdır!

ndim: Array boyut bilgisi

shape: satır, sütun sayısı

size: eleman sayısı

np.random.randint(1,10,9) #1’den 10’a kadar 9 elemanlı bir dizi oluştur!

np.arange(0, 30, 3) # 0’dan 30’a kadar 3’er artan dizi oluştur!