**LISTS**

**[]**

**List()**

**#list() #bos liste olusturuyor**

**#list(str1) #iterable bir veri koydugunuzda listin icine liste ceviriyor**

**#liste\_1[2:2] #2.elemandan basla 2 ser git demek bu**

**#liste\_1+liste\_2==liste\_2+liste\_1 ##toplamlari ayni degildir. siralama onemli**

**#listeler birbiri ile toplanabilir**

**#list ile integer toplayamam**

**#list ile int toplanmaz hatasini verdi**

**liste\_1+3**

**Insert and Append**

**## insert indekse kendi yerlesir. digerlerini saga kaydirir**

**#liste\_2.insert(4, "Hanife") 4. indekse kendi yerlesti. digerlerini saga kaydirdi.**

**sort**

**thislist.sort(key=str.lower) #burda ne demek istedim**

**#key atadim, yani once hepsini lower'a cevirsin ondan sonra siralasin**

**#yani buyuk kucuk onemi olmadan harfleri ona gore siralasin.**

**#bana bir anahtar ver, ona gore yapayim demek**

**#burda lower fonksiyonuna tabi tuttum. lower**

**#Python'daki bir listenin sort() yönteminin listeyi yerinde sıralamasıdır, yani yeni bir sıralanmış liste döndürmez. Bunun yerine, orijinal listenin kendisini değiştirir. list5.sort(key=str.lower) sonucunu liste6'ya atadığınızda, aslında list6'ya Yok(None) atarsınız, çünkü bu sort() yönteminin dönüş değeridir.**

liste5 = ["banana", "Orange", "Kiwi", "cherry"]

liste5.sort(key=str.lower) #bu islem liste5'i degistirme islemidir sadece.

liste6 = liste5 #bu calisir cunku liste5'in sortlanmis halini liste6ya atadim

**Pop()**

**thislist.pop()**

**## Remove and return item at index (default last).**

**#yani son elemani siler.**

**#thislist.pop() #neyi sildigini dondurur.**

**Clear()**

**liste\_3 # liste her seyi temizledi.**

**#ama liste hala hayatta, bos liste olarak geliyor**

**Del()**

**# clear'dan farki olarak tamamen siler. ortada liste kalmaz. empty bile**

**count()**

**#listenin icinde elemanin kac kere oldugunu sayar**

**Extend()**

**## extend'in append'den farki ayri birer eleman olarak atamasi listeye**

**##append'de ornegin iki eleman yazdin diyelim onlari butun olarak ekliyor, iki elemanli bir liste ekliyor, eleman sayisi 1 artiyor.**

**##extendle iki ayri eleman olarak ekliyor, eleman sayisi 2 artiyor.**

**## yani biz extendle listenin icine iterable bir sey gonderebiliriz.yani ya liste ya da string gibi. bunlar iterable'dir cunku. sayi gonderemeyiz**

**Join()**

**#birlestirme islemi yapar. Iterable olmalidir.**

**#"\*".join(liste\_6) #sep operatorune benziyor yani.**

**# arasina ne koyarsan onla birlestirir.**

**'Berat\*Filiz\*Rmzn' haline getirir.**

**Abs()**

**#mutlak deger**

**abs (9-12)**

**TUPLES**

**# tuple olusturma yontemleri**

**tuple() #normal parantezle de tuple olusturulur.**

**()**

**# Tuples degismez. listeler gibi degildir.**

**# listeler daha yavastir bu yuzden. Tuples degistirilemedigi icin hizlidir.**

**# immutable-degistirilemez.**

**# tuples de cok method yok**

**# listelerin icinden eleman alinir cikartilir**

**# tuples daha guvenlidir. verinin degistirilememesi nedeniyle.**

**# iterable. icine itirable bir eleman verirsen calisir.**

**# icine bir sey vermezsen bos dondurur.**

**# anlami demet tuple in**

**# listeler daha esnektir, degisiklik yapilir**

**#type("Alparslan") bu tuple ama #type'ini str verdi.sonunda virgul olmadigi icin.**

**## tek elemanli bir tuple olusturmak istiyorsak**

**#sonuna virgul koyuyoruz.**

**#iki elemanli olacaksa zaten araya virgul koyacagiz**

**("techproed",)**

**type(("techproed",)) # parantezi iki tane koymayi unutma!**

**# cunku tupple tek parantezle gosterilir.**

**#tuple(3,5,7) \ BU NEDEN HATA VERIYOR --> tuple expected at most 1 argument, got 3 diyor**

**#tuple func. kullanarak yapacaksak icine iterable tek eleman yazmamiz gerek.**

**# () ile tuple yapiyorsan sorun yok**

**# tuple sadece tuple ile toplanabilir**

**#potpori[3].append(4) # 3.indeksine append'le 4 ekledik, tuple icindeki listeyi degistirdik sadece.**

**#cunku 3. indeksi bu tuple'in bir liste. liste de degistirilebilir.**

**#boylece degistirilemeyen tuple'in icindeki listeyi degistirmis olduk.**

**#tuple'in icinde liste olmasaydi, tuple'in icine bir sey eklemek mumkun degil. Immutable tuple.**

**Range**

**## cok kullandigimiz bir func range()**

**#sayi uretir. aralik veririz**

**# starti stepi stopu olan**

**#range(stop) -> range object range(start, stop[, step]) -> range object**

**#son eleman dahil degil**

**# iterable.dir**

**#tuple(range(5, 15)) (5, 6, 7, 8, 9, 10, 11, 12, 13, 14)**

**## start stop step**

**list(range(2,25,2)) #list urettik. aralarina virgul konur range'de.**

**# 2'den basla 25'e kadar 2ser 2ser git demek bu,**

**#start stop step**

**DICTIONARY**

**{}**

**dict()**

{}.fromkeys()

**Seklinde olusturulabilir.**

**# coklu verilerde cokca kullanilir. biraz daha yavaslardir.**

**# elemanlari virgulle ayrilir ama bir key olacak ve ona denk gelen**

**# dictionaryler listler gibi mutable.dir/ degistirilebilir.**

**# yani assign etmeden degistirebilirsin**

**# bir key'e, bir value atayarak yapabiliriz. birden fazla value da atanabilir**

**# itemler key ve value dan olusur, ve virgullerle ayrilir**

**# {} kullandigimiz zaman ayirmak icin ‘:’**

**# dict() fonksiyonu kullandigimiz zaman da ayirmak icin ‘=’ kullaniriz**

**# tuple gibi, liste gibi indekslenemez. key'leri cagirilarak indekslenebilir.**dict\_1["key1"]

**# value olarak list de tuple da dict de kullanabilirsin.**

**# key ler icin ise sadece tuple kullanabilirsin bunlardan.**

**#key ve value birlikte item olarak adlandirilir.eleman sayisi item olarak sayilir.**

**# key'ler dict'den olusmaz**

**#duplicate key olmaz dict’lerde. Ayni key den iki tane olmaz!**

**#listede pop'un icine bir sey yazmazsam sonuncusunu siliyordu**

**# ama burda icine bir sey yazmam gerek**

**(# append le tek eleman eklerken# update ile istedigin kadar ekliyorsun)**

**Dict()**

**dict() # bos dict olusturur. #sanki atama yapar gibi = kullanirim.**

**# dict func icine atayinca one mesela str gibi alir.**

**dict(one=1, two=2, three= 3)** {'one': 1, 'two': 2, 'three': 3}

**# yani int olanlari dict() icinde kullanamazsin.**

**# int icin {} kullanabiliriz.**

**# dict() icinde yazinca str olarak aliyor.**

**SETS**

**# degistirilebilir. mutable**

**# setler indekslenemez. siralamasi yok.**

**# iterable.dir. icine iterable alir**

**# unordered unique elements**

**# ayni elemandan iki tane barindirmaz.**

**# siralama yoktur. undordered**

**# asciiye gore siralar jupyter kullanirsan. printte boyle bir siralama yok**

**# yani setlerde normalde siralama yok**

set([])

aa= {}

**# bu sekillerle tekrar etmeyen listeleri elde etmis oluruz.**

**# empty\_set olusturmak icin set() func kullanmam gerek.**

**# {} bunun type'i dict olur.**

**# setler 2 ile 2.0'i ayni olarak algilar**

**# # {"key" : "value"} # dict formati**

**# {"elma", "armut"} # set formati**

**# True'yu 1 olarak alir setler.**

**# ilk True'yu gorurse True'yu alip 1'i eler.**

**UNION-INTERSECTION-DIFFERENCE IN SETS**

**# Difference, A'da olup B'de olmayan, yani A-B**

**# B'de olup A'da olmayan da B-A**

**# set\_5-set\_6 ya da set\_5.difference(set\_6) seklinde yazilabilir**

**# set\_5 & set\_6 ve set\_5. intersection(set\_6) intersectioni gosterir.**

**# set\_5+ set\_6 ile olmaz ama set\_5.union(set\_6) ile union olur.**

#**symmetric\_difference**

**# set\_7.symmetric\_difference(set\_8) ile ve**

**# set\_8^ set\_7 ile yapilabilir/**

**#difference update**

**# set\_5.difference\_update(set\_6) ile yapilabilir.**

**#Union Update**

**# union update dogrudan update ile yapilir.**

**# set\_6.update(set\_5) demek set\_5 ile set\_6 union demek**

**# set\_6 artik set\_5'le set\_6 nin union'u oldu.**

**# update oldu set\_6 boylece**

**#add**

**# set icine list ve dict ekleyemeyiz ama tuple ekleyebiliriz.**

**# list ve dict tipinde bir veri, set icinde saklanmaz.**

**#clear, issubset, remove, discard, pop**

**# clear bos set haline getirir**

**# set\_9.issubset(set\_10) bu da set\_9, set\_10'un alt kumesi midir demek**

**# set9 <= set10 boyle de sorulur.**

**# remove veya discard ile set’ten eleman silebilirsin**

**# ama remove ile icinde yoksa hata alirsin**

**# pop da rastgele birini siler**

**WHILE AND FOR LOOPS**

**# dongu halinde yapmamiz gereken islemler icin,**

**# diger elemanlara da ulasmak istedigimiz zaman,**

**# birden fazla islemi ayni anda yapmak istedigimde kullanirim.**

**WHILE LOOP**

**# butun elemanlarina ulasmasi gerek. naparim? bunun icin indexlerine ulasmam gerek. []**

**# bunun icin dongunun eleman sayisi kadar calismasi gerek (len)**

**# condition dogru oldugu surece sonsuza kadar gider. o yuden stopper koyman lazim**

**# While dongusu True oldugu surece devam eder. Ya False'a donecek bir stopper girmen gerek**

**# ya da break girmen gerek**

**# durdurucuyu while'in icine yazmazsam da sonsuza kadar devam eder.**

**# random modulu**

**# eklemezsek kullanamayiz. notebook'a indirip methodlarindan faydalanabilirsin.**

**# kutuphane degil modul**

**# import random as rd ile import edebilirsin.**

**# random.random() # her calistirdigimizda farkli sayi random olarak getiriyor**

**# seed**

**#Initialize internal state from a seed. sayilari o noktaya referansla uretir.**

**# seed eklersek hep ayni sayiyi uretir. ayni sayiya ulasmak istersen seed belirleyeceksin.**

**# randint**

**# Return random integer in range [a, b], including both end points.**

**# girilen sayilar arasinda sayi uretir.**

**## Min ve Max**

**# min max icine str yazarsam asciiye gore siralar.**

**# max(["hello", "sedat", "Techpro"], key=str.upper) # hepsini buyuge cevirip ona gore siraliyor**

**# hepsini upper yap icinden max olani dondur demek**

**# key bir func.dir. hepsine tak tek ugrar, sonucu karslilastirir.**

**FOR LOOPS**

**# calismasini istedigimiz sure boyunca kodlari calistirmaya yarar while gibi**

**# While dongusu kosul True oldugu surece calisir. Ta ki False'a cevrilene kadar**

**# Break kullanirim ya da.**

**# For donguleri de ayni mantikla calisir. Dongu devam ettigi surece**

**# kosul ile degil iterable degerle calisir farkli olarak while'dan**

**# for dongusu iterable'in sayisi kadar doner**

**# str, set, tuple, dict hepsi iterable oldugundan for dongusune uygundur.**

#for degisken in iterable :

islem satirlari

**# elinizde bir liste var bunun tum elemanlarina islem yapmak istiyorsun for kullanirsin**

**# mesela bankada sifre giriyorsun 3 hakla sinirlandirilmissin, while kullaniliyordur**

**# bir durum True oldugu surece, gibi bir durumsa while kullaniri**

ornek. hava soguk oldugu surece her eve komur gonderilsin

**# bir iterable(Lists, tuples, dictionaries, and sets) uzerine is yapmak istedigimiz zaman for loop kullaniriz**

ornek. siniftaki ogrencilerin her birisiyle bir calisma yapmak istiyorsak for kullaniriz

**Break and Continue**

**# break nerde duracagini belirler**

**# continue da neyi atlayacagini belirler.**

**# k+= i #bu toplama islemi yaptiriyor.her seferinde i'yi ekle "continue dedigimiz eleman" haric demis oluyoruz**

**Zip and enumarate**

**# zip, len'i en dusuk olan kadarla eslestiriyor.**

**# enumerate de start noktasi vermezsen 0’dan verirsen verdigin sayidan baslayip siralandiriyor**

**List Comprehension**

**# for dongulerini cok hizlandirir, 20 dak'dan 30 sn'ye kadar hizlandirabiliyor**

**#**[ i \*\* 2 for i in liste\_1 ] #normal comprehension kullanimi

# sonuc ve for dongusu seklinde yaziyorum.

**if ile list comprehension kullanimi:**

**## sadece if kullanacaksan for dongusunun sagina yazman gerekir** [i for i in liste\_2 if i %2 ]

**# if i %2==0 ciftler icin, if i %2 de tekler icin yazarim**

#(2'ye bolumunden kalan ve 1'i True olarak algiladigi icin)

**# if True: calisir.**

**If else list comprehension kullanimi:**

**## hem if hem else olursa for dongusunun soluna yazarim if else’i**

**# ne yazdiracagini ise if' in soluna, else' in sağına yazıyoruz hep**

[i if i %2==1 else i\*\*2 for i in liste\_3]

**Nested Loops**

**#ilk for dongusunun ilk elemani icin, ikinci for dongusunun tum elemanlari calisir once**

**# sonra ilk for dongusunun ikinci elemanina gecer.**

**# iki ile de sinirli olmak zorunda degil ama, daha fazla dongu varsa da ayni mantikla calisir.**

**# matematikteki kartezyen carpim gibi**

**# bunun da kullanim alani yaygin**

**# while ile for'u da ic ice kullanabiliriz.**

**FUNCTIONS**

**# def ile yapiyorum.**

**# notebook'da tanimladiktan sonra istedigim yerde cagirabilirim.**

**# gomulu bir fonk olarak calisir.**

**# return dedigim zaman farkli bir isleme tanimlayip daha sonra kullanabilirim. print ise bir gosterimdir**

**#my\_len1([1,2,3,4,5]) # bu sekilde func'da cagirirken ([1,2,3,4,5])'a arguman deriz.**

**# func'i tanimlarken yazdigim iter'e ise parametre deriz. func cagirirken tanimladigim iter arguman olmus olur.**

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

**#args**

**## args \* ile yapilir. iterable hale getirir.**

**# def icinde tanimladigim func'daki eleman sayisindan az ya da cok girdigim zaman hata vermemesini saglar.**

**# yani istedigimiz kadar eleman girebiliriz.**

**# benim girdigim elemanlarin hepsini degerlendirsin istedigimiz zaman bunu kullaniriz.**

**#kwargs**

**## \*\* kwargs olunca bir dict'e denk gelir. keywords arguman vermen gerekir icinde.**

**FILTER**

**# bir func ve bir iterable girmem gerekir filter icine**

**## Filter icine yazdigimiz iterable'lari, tanimladigimiz func acisindan True olup olmadigina gore filtreler.**

**# icine func yerine None yazsaydim tum elemanlari dondururdu.**

**## none dersem hepsini True aldigi icin hepsini dondurur.**

**# filter iterable bir eleman**

**# o yuzden gorebilmemiz icin o yuzden list gibi tuple gibi collection type'larla cagirmamiz gerek.**

**# return yerine print dersek olmaz.**

**# return yaparsam farkli degiskenlere tanimlayip kullanabiliyorduk.**

**# bir kere kullanacaksak print de diyebilirdik ama sonra kullanacaksak return demem lazim**

**Lambda ile fonksiyon olusturma**

**# Lambdayi direkt olusturup kullanabiliyorsunuz. Python'da hafizada yer edinmiyor degiskene atamadikca.**

**MAP**

**# icinde bir func ve iterable var bunun da**

**# sonunda map object dondurur.**

**# icindei func'i kullanip birden cok islem yapmamizi saglar**

**# filter'dan farki, filterin True False ile calismasi**

**# Lambda'dan farki birden cok elemanla calismasi**

**# map de bir func oldugu icin, ic ice bir func kullanmis oluruz (nested func)**

**Recursive Function**

**# tanimladiginiz bir func'i kendi icinde tekrar kullanmak bu da**

**# nested ise bir func tanimlayip onu baska bir func icinde kullanmak**

**# sanki bir for dongusu gibi calisacak ve bitirmek icin bir sey gerekecek**

**EXCEPTION HANDLING**

**# TRY EXCEPT  
# hata alsan da yine de sunu yap demek istedigimiz zaman kullanabiliriz.  
# yanlislikla yazdigim bir seyde mesela, hatayi ortadan kaldirmak ya da hataya ragmen kodu calistirmak icin kullanilir.  
# hata varsa except calisir.   
# except'e bir sey yazmazsam tum hatalari kontrol eder.  
# ama hata TypeError iken, except'e, TypeError yerine NameError yazarsam calismaz.  
# hatayi onlemek icin hatanin neler olabilecegini ongorerek girmem lazim kodu  
# except ile birden fazla olasi hata tanimlayabilirim (TypeError, ValueError, ZeroDivisionError)  
# boylelikle hatalari kisisellestirmis oluyorum.## try-except-else-finally  
# try calisirsa else calisir.  yani tyr'i dene, olursa else'i calistir demek  
# yani except'lere yazdiginiz hatalar yoksa, else'i calistirir.**

**# python satir satir gidiyor hata aldigi yere kadar calisir.**

**# hata aldigi yerde except'ler calisir**

**# finally ise her halukarda calisir sonunda.**

**# artik finally dan sonra, except ve else yazamayiz.**

**# except ile ozel bir hata belirteceksek oncelikle onu belirtmemiz gerekir.**

**MODULS**

**# daha once python'da yuklu olmayan bir modul kullanacaksan onu install etmen gerekir.**

**# conda kullananlar conda install ile, python kullananlar pip install ile install edebilirsin.**

**# pip install rembg gibi, conda install rembg gibi bir modul yukleyeceksin diyelim. bunu install ederek pythonin icine yuklemis olursun.**

**# ama random yuklu bir modul pythondaki, sadece import etmen yeterli**

**# import' la cagirabilirsin gomulu bir func varsa, yoksa install etmen gerek once.**

**# pyautogui mouse'un durdugu yerin koordinatlarini veriyor. sonra koordinatlara git click yap diyebilirsin mesela.**

**##Create modul**

**# kendi modulunu yaparken kullanirsin**

**# modullerin icinde func lar degiskenler ayri ayri var**