Önemli uyarı

Windows için python’u ilk kurarken, kurulum sayfasının ilk bölümünde, altta add path denilen bir kutucuğun işaretlenmesi lazım, yoksa VSC proğramı pyton’u yüklenmemiş olarak görüyor ve işlem yapamıyor.

# # string

# print() # "string" kısaltılmışı"str"dir, tırnak içindeki tüm verilerdir.

# print("Bu ilk dersde öğrendiklerim VSC kullanma ve")

# print("terminalden pyton oluşturma oldu.")

# print("""Noktaların kullanımı konusu işlendi.""") # üç üst tırnak veya virgül olursa,

# # payton arada yazılanların doğruluğuna bakmadan tüm yazılanları kabul ediyor.

# print() # bu şekilde satır atlıyor.

# print(3.3) # sayılarda bu şekilde yapınca matematiksel işlem oluyor.

# print("3.3") # bu şekilde yapınca matematiksel işlem olmuyor,

# # çünkü tırnak içindeki tüm veriler "string" olarak kabul ediliyor.

# # variable and value, değişken ve değer

# print() # "variable", türkçe karşılığı değişken demektir.

# my\_age = 47 # ilk yazdığımız variable=değişken, ona verdiğimiz ise value=değerdir.

# your\_age = 45

# my\_age = your\_age # benim yaşım, senin yaşına eşit dediğimiz için

# print(my\_age) # my\_age yaşı 45 olarak sonuç çıkar.

# print()

# ocak = mart = mayıs = temuuz = eylül = kasım = 31

# şubat = 28

# nisan = haziran = ağustos = ekim = aralık = 30

# print(ocak, mart, mayıs, temuuz, eylül, kasım)

# print(şubat)

# print(nisan, haziran, ağustos, ekim, aralık)

# # aritmetik işlemler

# print() # "sayısal işlemler"

# print(4 + 2) # toplama işlemi

# print(4 / 2) # float division, bölme işleminde "float" olur, yani sonucu 2.0 verir

# print(4 // 2) # integer division, bu bölme işelminde "integer" olur, yani sonucu 2 verir

# print(4 \* 2) # çarma işlemidir

# print(4 - 2) # çıkarma işlemidir

# print(4 \*\* 2) # exponentiation, üslü sayılar yazılırken kullanılır

# print(10 % 3) # modulus, bölme işleminde artık sayıyı verir, yani burada

# # 10, 3'e bölündüğünde artık sayı 1 olduğundan, bu değeri verecektir.

# # aritmetik işlemlerde = işaretinin kullanımı

# -=, sayı = 3 diyelim, x -= 3, demek x = x - 3 demektir.

# +=, x += 3, demek x = x + 3 demektir.

# \*=, x \*= 3, demek x = x \* 3 demektir.

# /=, x /= 3, demek x = x / 3 demektir.

# //=, x //= 3, demek x = x // 3 demektir.

# %=, x %= 3, demek x = x % 3 demektir.

# \*\*=, x \*\*= 3, demek x = x \*\* 3 demektir.

# # aritmetik işlemlerde işlem sırası

# () parantez

# \*\* üslü sayılar

# -3 negatif sayılar

# \*, / çarpma ve bölme

# +, - toplama ve çıkarma

# a = (1 + 3) \*\* (2 \*\* (1 \* 2 / 2) / 2) # işlem sırasına göre en içteki parantez yapılarak devam edilir. / işareti olduğu için sonuç float çıkacaktır.

# print(a)

# # escape sequences, \n, \t, \b (string içinde kullanılırlar)

# \ işareti kendinden sonra gelen karekterin özel bir anlamı veya gücü varsa, onları elinden alır, yani etkisiz hale getirir.

# \n bir alt satıra geçirir.

# \t tab dediğimiz dört boşluk koyar.

# \b boşluğu siliyor, yani kelimeyi bir tık sola alıyor.

# bunlar string içerisinde anlamlıdır, string içerisinde değillerse güçleri yoktur.

# \b aradaki boşluğu kaldırarak yazdırır.

# print("We are", "boosting", "our", "brotherhood")

# print("We are", "\boosting", "our", "\brotherhood")

# \ kendinden sonraki karakterin gücünü etkisiz hale getirir.

# print('it\'s essential to learn Python\'s libraries in IT World') # normalde cümle içinde tek tırnak olduğundan hata vermesi lazım ama \ işareti

# kendinden sonraki karakterin gücünü elinden aldığı için hata vermiyor.

# \t kılavyede yer alan tab işlevini yapar, yani 4 karakter boşluk bırakır.

# print('first', 'second', 'third', sep='\t') # sep= seperate-ayırmak anlamına gelir.

# print('first', 'second', 'third', sep='\n') # stringleri alt alta yazdırır.

# print('first', 'second', 'third', sep='') # stringleri birleşik yazdırır.

# print('first', 'second', 'third', sep=' ') # stringleri boşluk bırakarak yazdırır.

# print('first', 'second', 'third') # bu şekilde olduğu gibi yazdırır.

# # boolean operations (True and False)

# boolean işlemlerde (and, or, not) işlemleri vardır.

# and ilk gördüğü False döndürür. şayet False yoksa gördüğü son True değerini döndürür.

# or ilk gördüğü True değerini döndürür. şayer True yok ise False değerini döndürür.

# not ise en kolayıdır, True yoksa False verir, False yoksa True verir.

# boolen okumalarında öncelik sırası not, and, or şeklindedir.

# python'da her işlemin bir True veya False değeri vardır.

# 1 değeri True, 0 değeri False'dir.

# print(False and not True) # burda ilk not'a bakar ve False verir, sonra and'e bakar ve False verir.

# bool\_var = False and not True

# print(bool\_var)

# print(True and False or not False or False) # 1. adım not False bakılır, 2. adım True and False, 3. adım False or True, 4. adım True or False ve 5. adım sonuç True çıkar.

# print(True and False or True or False) # 2. adım True and False

# print(False or True or False) # 3. adım False or True

# print(True or False) # 4. adım True or False

# print(True) # 5. adım

# None False değere sahiptir.

# Zero: 0, 0.0, 0j False değere sahiptir.

# "", [], {} False değere sahiptir.

# geri kalan herşey True değere sahiptir.

# print('True' and 'False') # bunlar string değere sahip olduğundan True değere sahiptir, yani True and True olur, and son True'yı yakaladığı için sonuç 'False' string değerini verir.

# print('True' and '') # burada '' False değere sahip olduğundan True and False olur ve and ilk False yakaladığı için '' boşluk değeri verir.

# print(2 and "hello world") # True and True olduğu için ve and son True'yı yakaladığı için sonuç 'hello world' string değerini yazdırır.

# print("" and "hello world") # False and True olur ve and ilk False yakaladığı için sonuç "" boşluk yazdırır.

# print(None and ()) # False and False olur ve and ilk False yakaladığı için None yazdırır.

# print([0] and 1) # True and True olur ve and False değer olmadığında, son True yakaladığından 1 yazdırır.

# And False duyarlidir onu arar buldgu an döndürur

# True and True..      son True

# True and False..      ilk False

# False and False...   İlk False

# False and True...     İlk False

# Or True duyarlidr onu arar ilk buldgunu döndürür.

# False or False...   Son False

# True or True...       İlk. True

# False or True...      İlk True

# True or False...     İlk True

# # Indexing and Slicing Strings-Dizinleme ve dilimleme dizeleri

# [start:stop:step]

# bu işlem iteyribıl yani elemanlarına ayrılabilen stringlerde yapışabilir, rakamlar ve sayılar iteyrıbıl olmadığı için bu işlemler onlarda yapılamaz.

# best = "Clarusway"

# print(best[4]) # tek rakam varsa indexlemedir (dizinlemedir).

# print(best[0:4]) # belli bir parçasını almak istiyorsak slayslamadır (dilimlemedir).

# print(best[::2]) # step'de 2 yazdığı için 0'ı alacak ve 2 atlayarak almaya devam edecek, sonuç Cauwy olarak çıkar.

# burada [start:stop:step] kuralını unutmayalım. bu işlem şu şekilde de yazılabilirdi.

# print("Clarusway"[::2]) # sonuç yine Cauwy olacaktır.

# # len fonksiyonu iteyrıbıl yani string gibi sayılabilen verilerde, karakter sayısını bulmak için kullanılır.

# best = "Clarusway"

# print(len(best)) # sonuç 9 çıkar çünkü 9 karakter vardır.

# # String Formatlama (+, =, \*)

# + işareti stringleri birleştirir.

# print("Alp" + "er")

# \* işareti string sonunda yazılır ve bir sayı verilirse, o belirlenen sayı kadar stringi yazdırır yani kopyasını çoğaltır.

# print("Alper" \* 3)

# aynı şeyi stringden önce sayı verilip \* yazılırsa da verir.

# print(3 \* "Alper")

# \* işareti string önüne yazılırsa o değeri karakterlerine ayırarak yazdırır.

# print(\* "Alper")

# = işareti + ile kullanılırsa(bu konuyu tam anlamadım)

# a = "Alp"

# b = "er"

# a += b

# d = "Tunga"

# a += d

# print(a)

# = işareti \* ile kullanılırsa (bu konuyu tam anlamadım)

# # string.format(), f-string - stringleri formatlama

# string.format en başta böyle yapılırken

# meyve = "elma"

# sebze = "ıspanak"

# fiyat = 5

# print("Ben {} ve {} aldım. Kilosu {} liraydı.".format (meyve, sebze, fiyat))

# bunun daha kullanışlısı olan f formatı kullanılmaktadır ve daha pratikdir.

# bunun temel nedeni f srtinglerde süslü parantezler içinde oynamalar yapıla bilmesidir.

# meyve = "elma"

# sebze = "ıspanak"

# fiyat = 5

# print(f"Ben {meyve} ve {sebze} aldım. Kilosu {fiyat} liraydı.")

# meyve = "elma"

# sebze = "ıspanak"

# fiyat = 5

# print(f"Ben {meyve} ve {sebze} aldım. Kilosu {3 + 2} liraydı.") # dikkat edin, 5 yerine 3+2 yazdık ve aynı sonucu aldık.

# meyve = "elma"

# sebze = "ıspanak"

# fiyat = 5

# print(f"Ben {meyve.upper()} ve {sebze} aldım. Kilosu {fiyat} liraydı.") # dikkat edin, elma kelimesini büyük yazdırmak için {}

# içinde oynama yaptık ve yine bize sonuç verdi. f string metodunun en büyük avantajı süslü parantez içinde değişiklikler yapmaya izin vermesidir.

# name = "Alper"

# print(f"My name is {name.capitalize()}")

# name = "Alper"

# output = f"My name is {name.capitalize()}"

# print(output)

# name = "Susan"

# age = "young"

# gender = "lady"

# school = "CLRWY IT university." # ev ödevi, bunlardan "Susan is a young lady and she is a student at the CLRWY IT university." şeklinde bir cümle oluşturun.

# # searching a string (Bir dizeyi aramak)

# string.startswith, string.endswith # stringlerde True yada False döndürür.

# text  = "www.clarusway.com"

# print(text.startswith("www"))

# print(text.endswith(".com"))

# print(text.startswith(".ww"))

# print(text.endswith("co"))

# print(text.startswith("w"))

# print(text.endswith("om"))

# str.replace(old, new) # old(değiştirilecek öge yazılır), new(yeni öge yazılır).

# replace de sadece kelime değil, kelinme içinden bir hecede, bir harfte alınıp değiştirilebilir

# str.swapcase() # büyük harflerle yazılmış stringleri küçük, küçük harfle yazılmışları büyük harflere çevirir.

# str.upper() # stringin tamamını büyük harfle yazdırır.

# str.lower() # stringin tamanmını küçük harfle yazdırır.

# str.tiitle() # her kelimenin ilk harfini büyük yapar.

# str.capitalize() # stringin ilk harfini büyük, diğerlerini küçük harfle yazdırır.

# sentence = "I live and work in America."

# print(sentence.replace("America.", "Kanada."))

# print(sentence.replace("Amer", "Kana"))

# print(sentence.replace("i", "+")) # cümledeki bütün "i" leri "+" ile değiştirdi.

# print(sentence.swapcase())

# print(sentence.upper())

# print(sentence.lower())

# print(sentence.title())

# print("I live and work in America.".capitalize())

# print(sentence.capitalize())

# print(sentence)

# stringler invariability yani değişmezlik özelliğine sahiptir. bu nedenle stringde yaptığımız bir değişikliği yeni bir değişkene yani veriable atmazsak, string ilk haliyle sonuç verir.

# aşağıdaki örnekte olduğu gibi.

# cumle = "Pc her zaman console ounlarini dover."

# print(cumle.upper())

# print(cumle)

# cumle = cumle.upper()

# print(cumle)

# string.strip(), string.rstrip(), string.lstrip # strip stringin başındaki ve sonundaki boşlukları siler, rstrip sağındaki, lstrip solundaki boşlukları alır.

# ayrıca rstrip sağındaki ve lstrip solundaki, içine yazılan harf yada heceleri ayrı ayrı ve birlikte arayarak silecektir. strip ise her iki taraftan düzeltme yapacaktır.

# text = "tyou can learn almost everyting in pre-classz" # YOU CAN LEARN EVERYTING IN PRE-CLASS yazdırın.

# print(text.lstrip("t").rstrip("z").upper())

# print(text.strip("tz").upper())

# print(text.upper().strip("TZ"))

# # Collection Types - Koleksiyon Türleri - List, Tuple, Dictionary, Set

# Coolection Types lar birbirinden farklı olan verileri içerisinde barındırabilir. bunlar iteyribildır yani elemanlarına ayrılabilirler.

# list() veya []

# tuple() veya ()

# dict() vaye {}

# set()

# text4 = ["Kuruko no Basket", 36, ["Slam Dunk"], 3.14, True, None] # liste içine string, intecir, float, bool, NoneType gibi farklı veriler yazılabilir.

# print(text4)

# liste1 = "Happy" # list, stringleri elemanlarına ayırarak yazdırır.

# print(list(liste1))

# liste2 = ["Happy", 36, 3.14, ["unhappy"]] # list, köşeli parantez içinde olduğu için tek bir liste halinde verir.

# print(list(liste2))

# list'in bir özelliğide liste içinde liste oluşturulabilmesidir.

# my\_list = ["Alper", "Clarusway", 2022]

# new\_list1 = list(my\_list)

# new\_list2 = [my\_list] # liste içinde liste oluşturulabilir.

# new\_list3 = [my\_list, 36] # list içine yeni bir değer atanabilir.

# print(my\_list)

# print(new\_list1)

# print(new\_list2) # list içinde list yapılabilir.

# print(new\_list3)

# print(len(new\_list1)) # new\_list1'deki sıfırıncı indeks "Alper" iken

# print(len(new\_list2)) # new\_list2'deki sıfırıncı indeks listenin kendisi olacaktır.

# print(len(new\_list3)) # new\_list3'deki sıfırıncı indeks lsitenin kendisi, 1. indeks 36 olacaktır.

# string değerini [] içinde yazarsak, tek bir eleman olarak alacktır. aynı string'i list() fonksiyonu ile yazarsak da herbir objeyi,

# boşluklarda dahil, bir indeks olarak alır. bu durum len() fonksiyonu ile görülebilir.

# my\_list1 = ["2022's hard."]

# my\_list2 = list("2022's hard.")

# print(my\_list1)

# print(my\_list2)

# print(len(my\_list1))

# print(len(my\_list2))

# # Basic Operations with lists - Listelerle Temel İşlemler

# list\_1 = []

# list\_2 = list()

# .append()

# .insert()

# .remove()

# .sort()

# .pop()

# .append() # listelerin sonuna eleman eklemek için kullanılır, eklenen elemanı hep son sıraya atar. append fonksiyonu içine tek değer yazılmalıdır, aksi takdirde hata verecektir.

# bu özellik striglerle yani str.append() şeklinde asla kullanılmaz.

# empty\_list =[]

# empty\_list.append(1)

# empty\_list.append("Multiverse")

# empty\_list.append("Multiuniverse")

# empty\_list.append(3.14)

# empty\_list.append([5])

# print(empty\_list)

# city = ["New York", "Londan", "Berlin", "Seul", "Sydney"]

# city.append("Addis Ababa")

# print(city)

# .insert() # listelerin içine değer eklemek için kullanılır, append'den farklı olarak iki değer girilit. ilki değerin atanmak istendiği indeks sırası,

# ikincisi ise atanacak değerin kendisi yazılır.

# list4 = [1, 4, 7, 9]

# list4.insert(0, "5") # ilk sayı atanacak indeks sırası, ikinci değer atayacağımız değerdir.

# print(list4)

# list4 = [1, 4, 7, 9]

# list4.insert(0, "5") # liste içine insert ile arka arkaya değerler atayabiliriz.

# list4.insert(3, 5)

# print(list4)

# .remove() # list içinden bir eleman silmek için kullanılır. silinecek değer () içine yazılır.

# list5 = [1, 4, 7, 9]

# list5.remove(4)

# print(list5)

# .pop() # () iken list içindeki son elemanı siler, () içine yazılan indeksdeki değeri siler.

# list6 = [1, 4, 7, 9]

# list6.pop()

# print(list6)

# list7 = [1, 4, 7, 9]

# list7.pop(0)

# print(list7)

# .sort() # list içindeki değerleri numaratik sıraya göre düzenler. sort aynı veri tiplerinde çalışır, yani numara ve harf olan bir liste de çalışmaz.

# harfleri ise aski kod değerine göre düzenler.

# python'da her bir küçük veya büyük harfin aski kod değeri vardır ve bu print.(ord("A")) ve print(ord( a)) ile öğrenilebilir.

# print(ord("A"))

# print(ord("a"))

# list8 = [1, 45, 7, 9, 0, 22]

# list8.sort()

# print(list8)

# list içinde, listeye ve bunlara ulaşma yöntemi.

# list\_1 = ["one", "four", "nine"]

# list\_2 = ["@", "\*-", "False"]

# list\_3 = [True, False, None]

# list\_4 = [[3], [44], [-12]]

# list\_5 = [[1, 3], [44, -40], [-12, 1]]

# print(list\_1)

# print(list\_2)

# print(list\_3)

# print(list\_4)

# print(list\_5)

# print(len(list\_1))

# print(len(list\_2))

# print(len(list\_3))

# print(len(list\_4))

# print(len(list\_5))

# list\_5 = [[1, 3], [44, -40], [-12, 1]] # bu list de 3 indeks bulunmaktadır. (0, 1, 2)

# print(list\_5[0][0]) # ilk önce ulaşmak istediğimiz indeksi sonra onun içindeki indeksi yazarak istediğimiz değere ulaşabiliriz.

# print(list\_5[1][0])

# print(list\_5[2][0])

# city\_list = [["New York", "Londan", "Berlin", "Seul", "Sydney"]] # srtingler de ise iç içe giderek, istediğimiz harfe kadar gidebiliriz.

# print(city\_list[0][2][4]) # 0 listenin kendisini, 2 liste içindeki değeri, 4 değer içindeki harfi getirir.

# # Slicing a list - Bir listeyi dilimleme (start:stop:step)

# animals = ['elephant', 'bear', 'fox', 'wolf', 'rabbit', 'deer', 'giraffe']

# print(animals[:])

# animals = ['elephant', 'bear', 'fox', 'wolf', 'rabbit', 'deer', 'giraffe']

# print(animals[3:])

# animals = ['elephant', 'bear', 'fox', 'wolf', 'rabbit', 'deer', 'giraffe']

# print(animals[:5])

# animals = ['elephant', 'bear', 'fox', 'wolf', 'rabbit', 'deer', 'giraffe']

# print(animals[::2])

# # Tuples # tuples invitibildır yani içindeki verileri değiştiremezsiniz. list ile arasındaki en büyük fark budur.

# değişmesini istemediğimiz değerlerde kullanılır. banka hesap numarası, kimlik numarası gibi.

# Tuple'da tıpkı list gibi birbirinden farklı değerleri içerisinde barındırabilir. str, int, float, list, tuple birarada olabilir.

# ()

# tuple()

# my\_tuple = ("wakabayashi") # tek elemanlı bir değer tuple olmaz, onu tuple yapmak için mutlaka parantezden önce , koymalıyız. virgül koymadığımız zaman türünü string olarak verecektir.

# my\_tuple1 = ("wakayabashi",) # burada parantezden önce virgül koyduğumuz için türünü(type) tuple olarak verecektir.

# my\_tuple2 = "wakabayashi"

# my\_tuple2 = tuple("wakayabashi")

# print(my\_tuple, type(my\_tuple)) # değeri str gördüğü için sadece kelimeyi yazdırır.

# print(my\_tuple1, type(my\_tuple1)) # değeri tuple gördüğü için parantezle birlikte içindekileri yazdırır.

# print(my\_tuple1, type(my\_tuple1), sep="\n") # \n new line yani yeni çizgi manasında olduğundan, bilgileri ayrı satırlarda yazdırarak çıktı verir.

# print(my\_tuple2)

# my\_tuple = (1, 2, 3, 4, 564, 7, 863)

# my\_list = list(my\_tuple) # tuple list fonksiyonuna alarak, list fonksiyonu olan [] içine aldık.

# print(my\_tuple, type(my\_tuple), sep="\n") # burada türü(type) tuple'dır.

# print(my\_list, type(my\_list), sep="\n") # burada türü(type), [] içinde olduğundan list'dir.

# print(my\_tuple[3]) # tuple içindeki değerlere indeks sayısını yazarak ulaşabiliriz.

# print(my\_tuple[3:5]) # ayrıca (start:stop:step) kuralıyla, birden fazla indeks yazarak da değerlere ulaşabiliriz.

# sehirler = ["Istanbul", "Izmir", "Ankara", "Van", "Erzurum", "Sivas", "Gonya", "Ssinop", "Mugla", "Gaziantep"]

# print(sehirler)

# sehirler\_tuple = tuple(sehirler)

# sehirler[9] = "Yozgat"

# print(sehirler)

# sehirler\_tuple[0] = "Mus"  # Bu çalışmaz.

# # Dictionaries - Sözlükler

# {} içinde yazarken key değişkeni 'str' olarak yazılır ve sonrasında : kullanılır; grocer1 = {'fruit' : 'apple', 'drink' : 'water'} şeklinde yazılırken

# dict() içinde yazarken key değişkeni tırnaksız ve sonrasında = kullanılır; groser2 = dict(fruit='apple', drink='water') şeklinde yazılır. bu ayrıma dikkat edilmelidir.

# bir anahtar(key) bir değer(value) çiftinden oluşur ve bu iki çift bir eleman olarak kabul edilir, {} içinde yazılır ve değerler virgül(,) ile ayrılır.

# sözlükte yer alan bir kelime ve onun açıklaması şeklinde düşünülebilir. bu nedenle sözlük manasına gelen dictionari denmiştir.

# Tuple ve list'de olduğu gibi birbirinden farklı değerleri içerisinde barındırabilir. str, int, float, list, tuple birarada olabilir.

# {key1 : value1,

# key2 : value2}

# first\_dict = {"kola" : 25, "ekmek" : 5, "makarna" : 5} # {} süslü parantez içerisinde yazım şekli.

# second\_dict = dict(kola=25, ekmek=5, makarna=5) # dict() şeklinde yazım şekli.

# state\_capitals = {  "Arkansas" : "Little Rock", # 0. indeks

#                     "Colorado" : "Denver",      # 1. indeks

#                     "California" : "Sacramento", # 2. indeks

#                     "Georgia" : "Atlanta"        # 3. indeks

# }

# print(state\_capitals["Arkansas"]) # dick içinden bilgi çekmek için [] içine key'i olduğu gibi yazmamız gerekir, value(değer) veya list'lerde olduğu gibi rakam yazarsak hata verir.

# dict içine atama yani ekleme yapmak için

# state\_capitals["Virgina"] = "richmond" # key ["..."] içinde = value "..." yazılmalıdır.

# state\_capitals["Virgina"] = "Richmond"

# print(state\_capitals) # yapılan son eklemeyi yazdırır.

# state\_capitals = ["Virginia" : "Richmond"] # bu şekilde hata verir.

# .items() # dict içindeki bütün herşeyi verir.

# .keys() # dict içindeki bütün key değişkenlerini verir.

# .values() # dict içindeki bütün value değerlerini verir.

# .update({}) # dict içine yeni bir key=value eklememizi sağlar ve yeni değeri en sona atar. ({....}) içine yazılmalıdır.

# del # dict içinden bir key silmek için kullanılır. diğerlerinden farklı olarak öne yazılır ve sonra dict eklenir. [...] içine silinmek istenen key yazılır, value asla yazılmaz.

# mix\_values = {"animal" : ("dog", "cat"),  # tuple

#                  "planet" : ["Neptun", "Pluton", "Jupyter"],  # liste

#                  "number" : 40,  # integer

#                  "pi" : 3.14,  # float

#                  "is\_good" : False  # boolean

# }

# mix\_keys = {22 : "number",

#             3.14 : "float",

#             True : "boolean",

#             "key" : "string"

# }

# my\_dictionary = dict(animal =("dog", "cat"), planet = ["Neptun", "Pluton", "Saturn"], number = 55)

# print(mix\_values.items(), "\n") # dict içindeki bütün herşeyi verir.

# print(mix\_values.keys(), "\n") # dict içindeki bütün key değişkenlerini verir.

# print(mix\_values.values(), "\n") # dict içindeki bütün value değerlerini verir.

# print(mix\_keys.items(), "\n")

# print(mix\_keys.keys(), "\n")

# print(mix\_keys.values(), "\n")

# mix\_values.update({"is\_bad" : True}) # dict içine yeni bir key=value eklemek için mutlaka ({...}) şeklinde yazılmalıdır.

# print(mix\_values)

# del mix\_values["is\_bad"] # diğerlerinden farklı olarak, noktasız bir şekilde, en öne yazılır ve sonra dict eklenir. silinmek istenen key [...] içine yazılmalıdır.

# print(mix\_values)

# # Nested Dictionari - iç içe sözlük

# dict'lerin içi içe geçmiş halidir.

# school\_records={

#   'personal\_info':                                      # "personel\_info" key'dir, "kid" ve "teen" value'dir.

#       {'kid':{'tom':{'class':'intermediate', 'age':10}, # "kid" key, "tom" ve "sue" value'sidir.

#           'sue':{'class':'elemantary', 'age':8}

#           },

#       'teen':{'joseph':{'class':'college', 'age':19},   # "teen" key, "joseph" vr "marry" value'sidir.

#           'marry':{'class':'high school', 'age':16}

#           },

#       },

#     'grades\_info':                                      # "grades\_info" key'dir, "kid" ve "teen" value'dir.

#       {'kid':{'tom':{'math':88, 'speech':69},           # "kid" key, "tom" ve "sue" value'sidir.

#           'sue':{'math':90, 'speech':81}

#           },

#       'teen':{'joseph':{'coding':80, 'math':89},        # "teen" key, "joseph" vr "marry" value'sidir.

#           'marry':{'coding':70, 'math':96}

#           },

#       }

# }

# print(school\_records["personal\_info"]["teen"]["marry"]["age"])

# print(school\_records["grades\_info"]["kid"]["sue"]["speech"])

# family = {

#     "erkek" : {

#         "baba" : {"yas" : 40, "meslek" : "eyt emeklisi" },

#         "kardes" : {"yas" : 22, "meslek" : "ogrenci"}

#     },

#     "kadin" : {

#         "anne" : {"yas" : 40, "meslek" : "emekli albay"},

#         "abla" : {"yas" : 27, "meslek" : "influencer"}

#     }

# }

# print(family["kadin"]["anne"]["meslek"])

# # Creating a set -

# {} # set = {} şeklinde boş set üretilemez, çünkü bu sadece dictionari de dict = {} şeklinde kullanılmaktadır.

# set()

# .add() # set içine eleman ekler.

# .remove() # set içerisinden eleman siler.

# set unordered yani içeriğini her yazdırdığında farklı şekillerde yazdırır, belli bir düzene göre yazdırmazi her seferinde değiştirecektir.

# kümelerde yer alan birleşim, kesişim, fark kümesi şeklinde çalışır.

# set\_1 = {'red', 'blue', 'pink', 'red'}

# colors = 'red', 'blue', 'pink', 'red'

# set\_2 = set(colors)

# print(type(set\_1))

# print(set\_2)

# bos\_set = {} # bu şekilde kullanılamaz.

# flower\_list = ["rose", "orchid", "cactus", "violet", "ginger", "rose", "orchid", "tulip", "tulip"]

# flower\_set = set(flower\_list)

# print(flower\_list) # list özelliği ile yazdırdığımızdan, sırasına göre yazdırır.

# print(flower\_set) # set özelliği ile yazdırdığımızdan, her seferinde karışık şekilde yazdırır.

# .intersection() # kesişim kümesidir, iki setin ortak elemanlarını yazdırır.

# .union() # birleşim kümesidir, iki setin ortak elemanları dışında kalanları verir (kesişim kümesi dışında kalanlar).

# .defference() # iki set kümesi arasındaki farkları verir.

# a = set("Ankara")

# b = set("Istanbul")

# c = set("Izmir")

# d = set("Afyonkarahisar")

# print(a, b, c, d, sep= "\n")

# print(a.difference(b))  # == print(a - b)

# print(b.difference(a))  # == print(b - a)

# print(a.union(b))  # == print(a | b)

# print(b.union(a))  # == print(b | a)

# print(a.intersection(b))  # == print(a & b)

# print(b.intersection(a))  # == print(b & a)