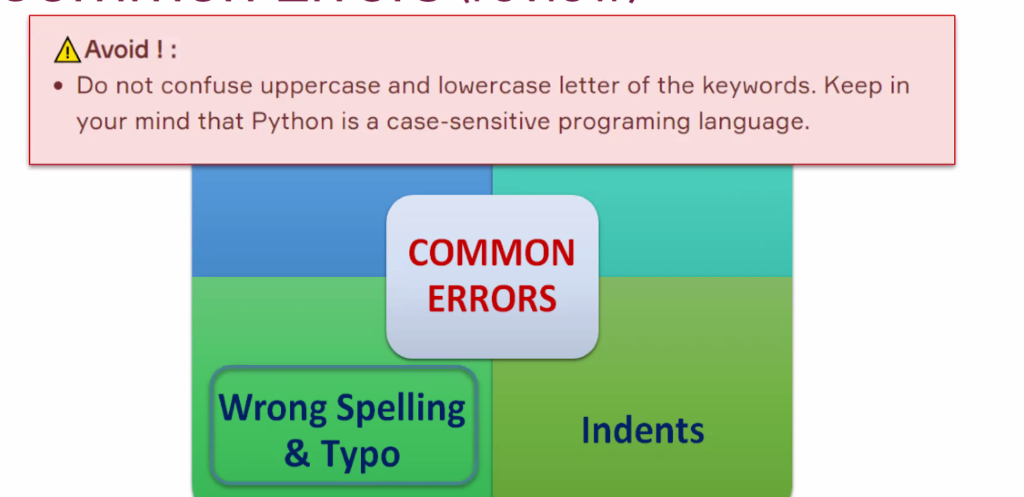
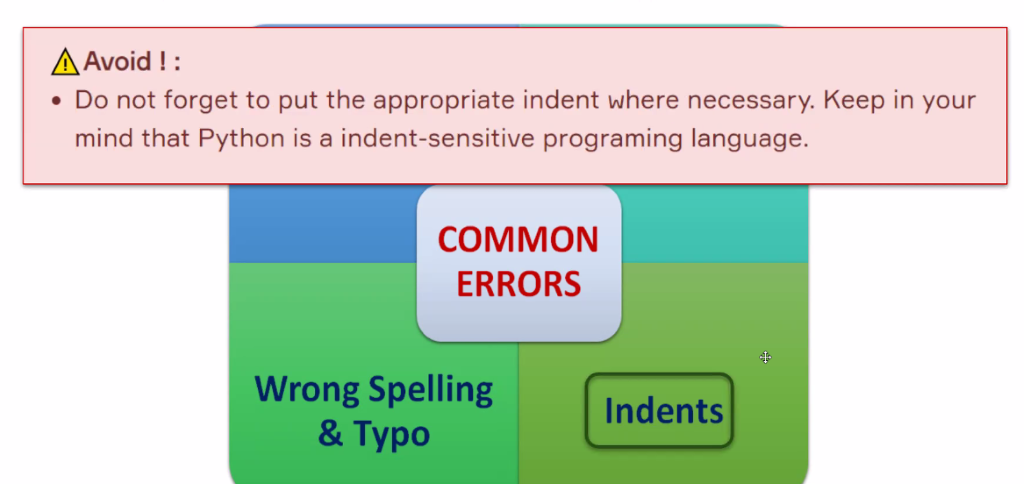
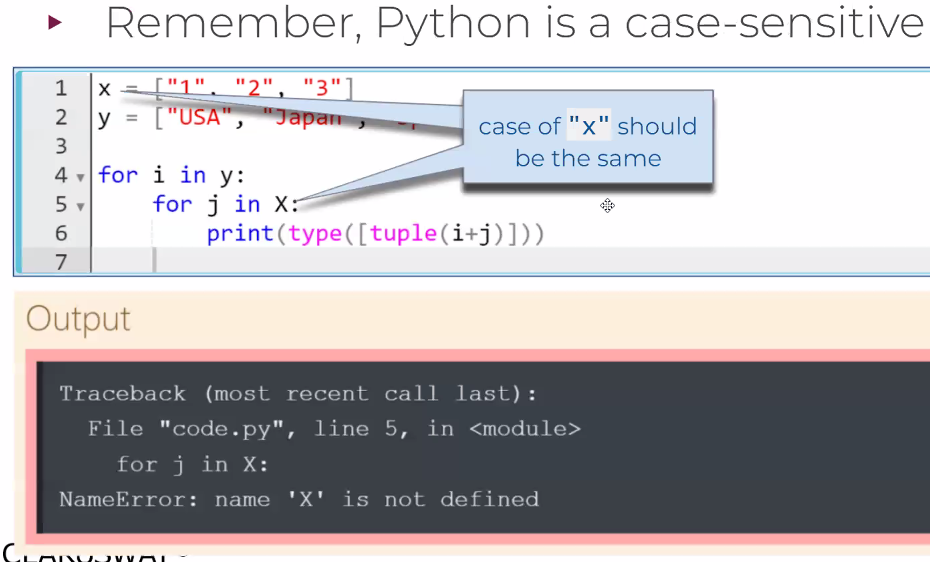


Hatta olduğunda traceback modülü çalışıyor.



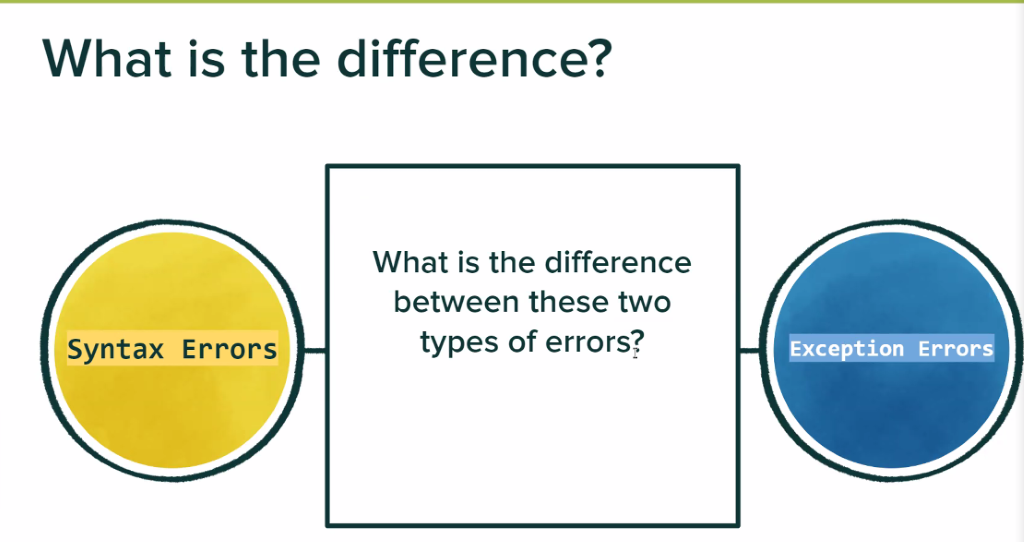


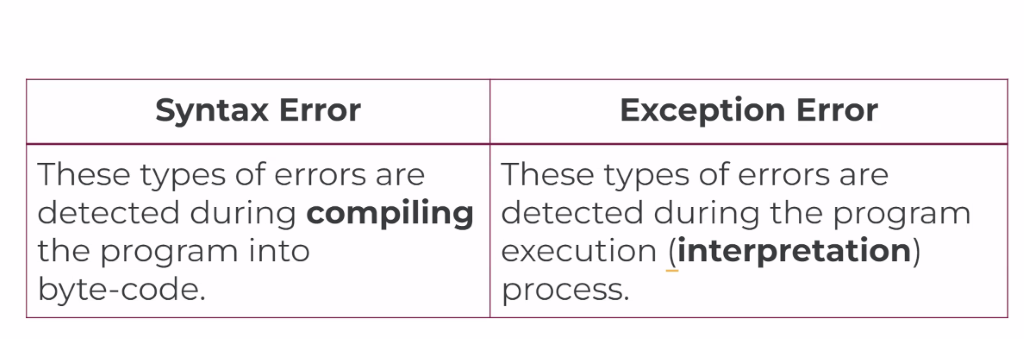




Case sensitive olduğundan dolayı alınan bir hata.

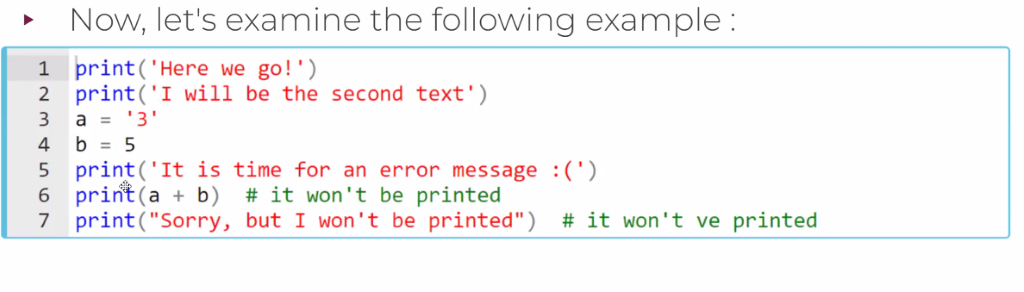
Exception Errors ile Syntax Errors hataları farklı hata tipleri:



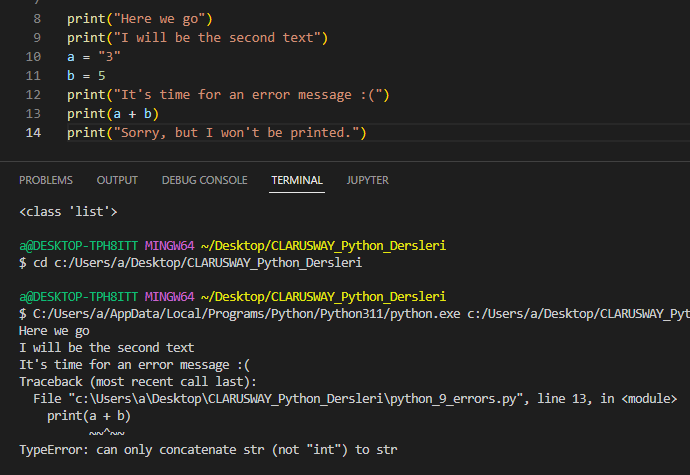


Syntax errors byte kode çevrilirken (compiling) bu hatalar bulunur ve programı hiç çalıştırmaz.

Python interprete edilen bir dil. Kodumuz satır satır okuyor yani. Okuduğu yere kadar program çalışıyor. Hata gördüğü yerde program hatayı verir ve çalışmayı durdurur. Bunlara exception errors dur.



Hatayı gördükten sonraki kısmı yazdırmaycak:

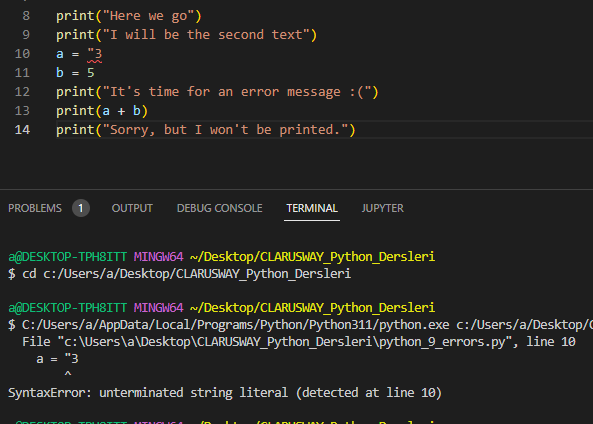


Traceback modülünü çalıştırıyor ve,

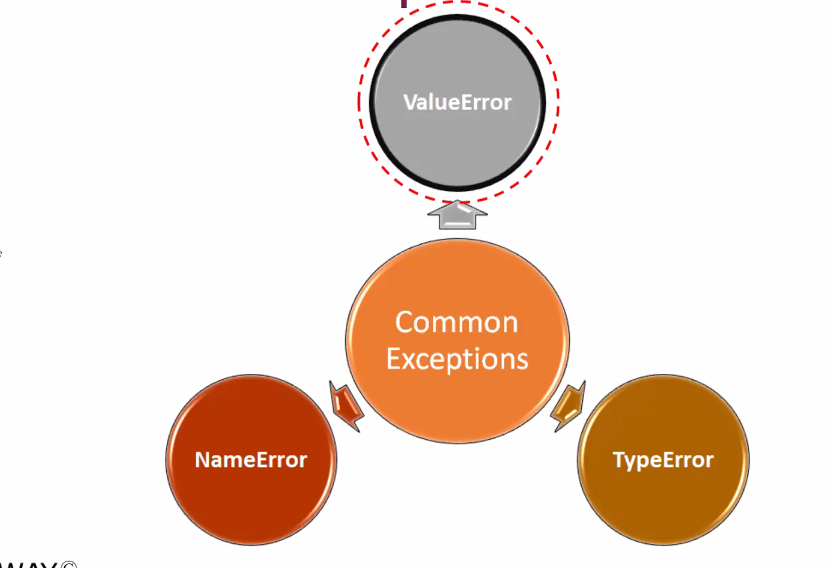
Altta hatanın ne olduğunu hangi satırda olduğunu bize söylüyor.

Bu bir exception hatasıdır. Yani hataya kadar kodu çalıştırır.

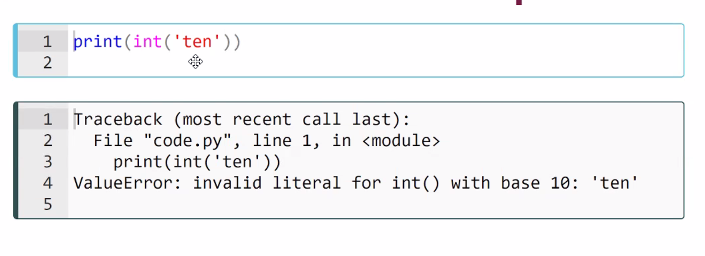
Ancak  bir syntax hatalı yapalım:



Syntax hatasını orta satırlarda yaptık ancak programı hiç çalıştırmadı.

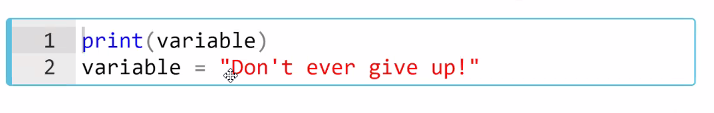


Value error uyarısı: (raised en error)

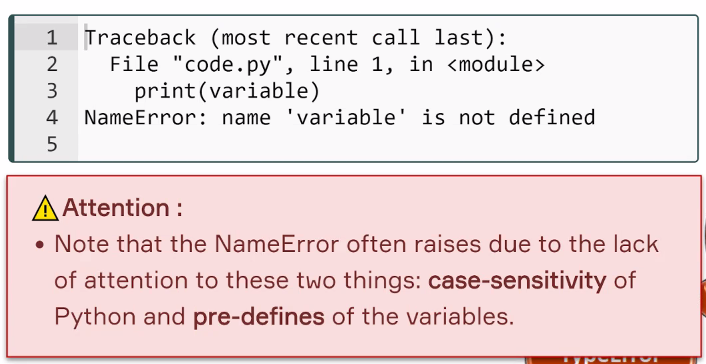


int içinde string ifade olmaz.

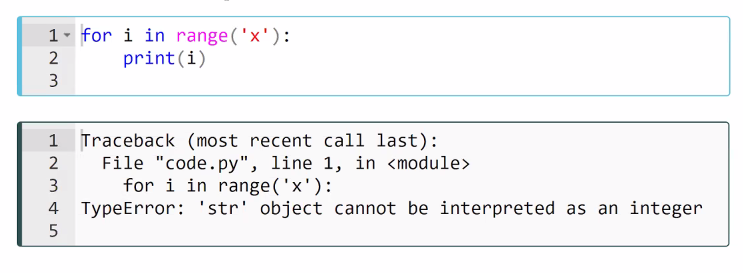
Name error:



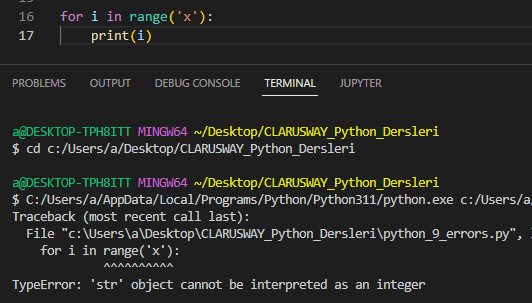
Python üstten aşağı okur bu örnekte print fonk öncesinde variable atanmadığı için hata verir.



Type error:

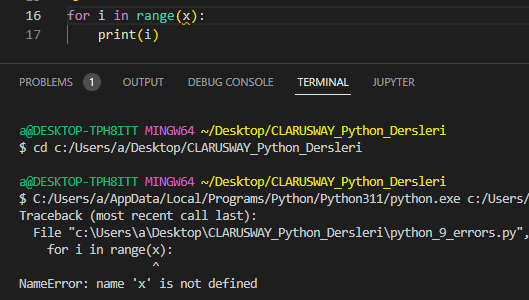


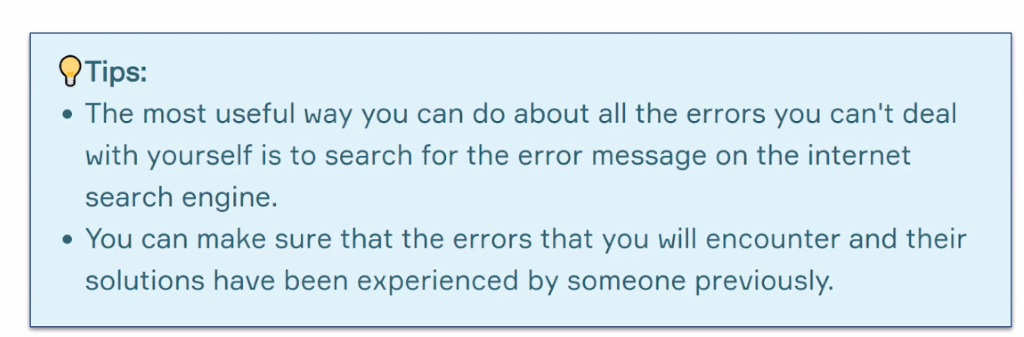
range içinde string ifade kullanılmış. Type error verir:



x i strden çıkaralım:

x i tanımlamadığımız için name error verir:



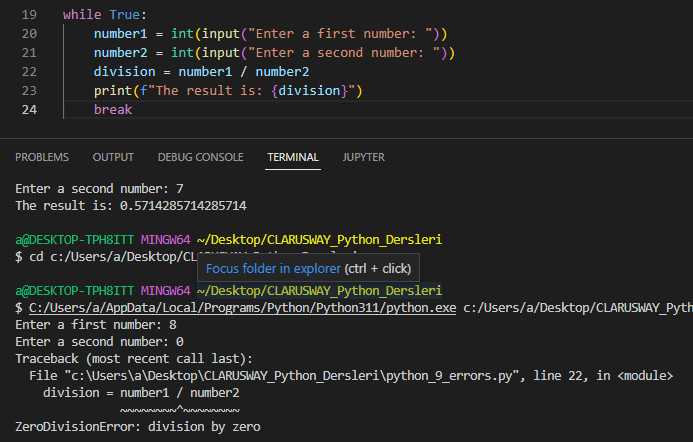


Error mesajlarının hepsini bilemeyiz. Hata aldığımızda internette araştırma yapmalıyız. Başımıza gelen hataların büyük çoğunluğu başkaları tarafından çözülmüştür.

Python un kendi dökümantason sitesinde de detaylı bilgilere sahip olabiliriz.

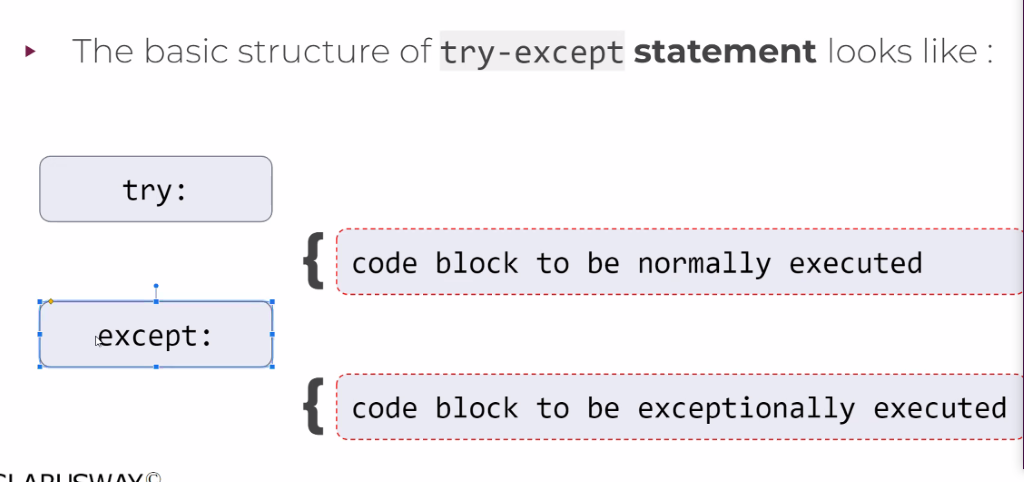
Exception Handling:





yukarıdaki koda çalışıyor ancak 0 yazdığımızda hata verecek.

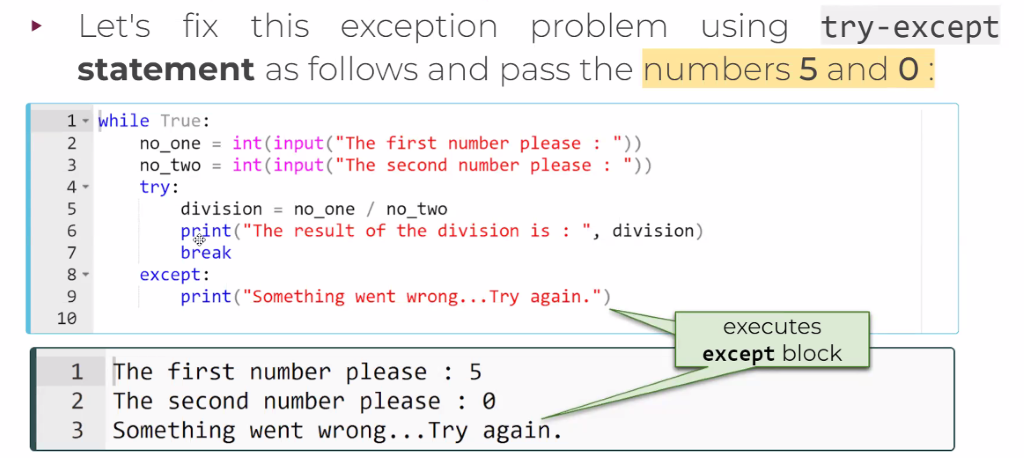
Bunun için yapacağımız yöntem try-expect:



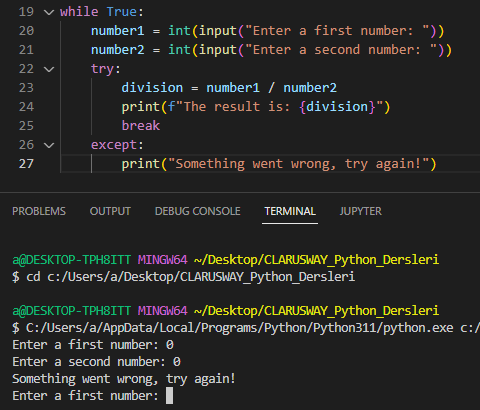
Bunu her örnekte düşünebiliriz.

Kod bloğunu try: kısmına yazıyoruz, hata gördüğünde çalışmayı bırakması yerine except: bloğuna yazdığımız kodu çalıştırır.

hatayı aldığımız kısım division = number1 / numbr2 kısmı;

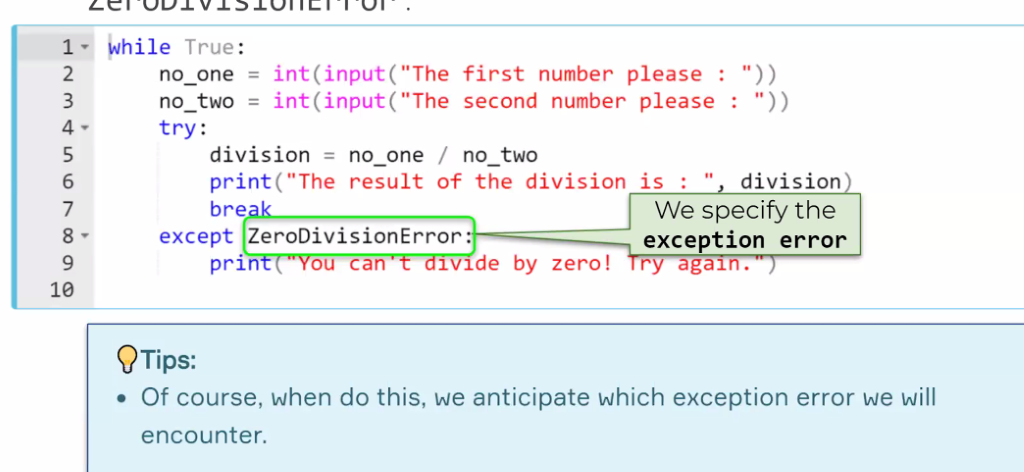


Vs code da yazıyoruz:

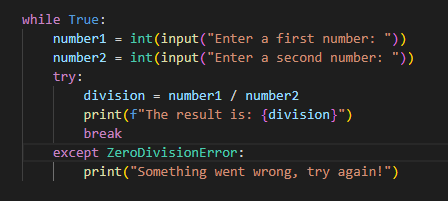


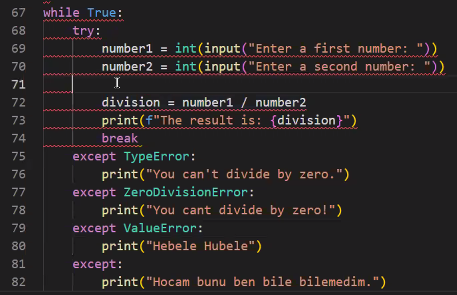
Amacımız hatanın görülmesi değil, kodun çalışması. Bu kez 0 a bölündüğü halde koddan çıkarmadı ve except bloğundaki kodu çalıştrıdı.

Eğer hatanın ismini biliyorsak:



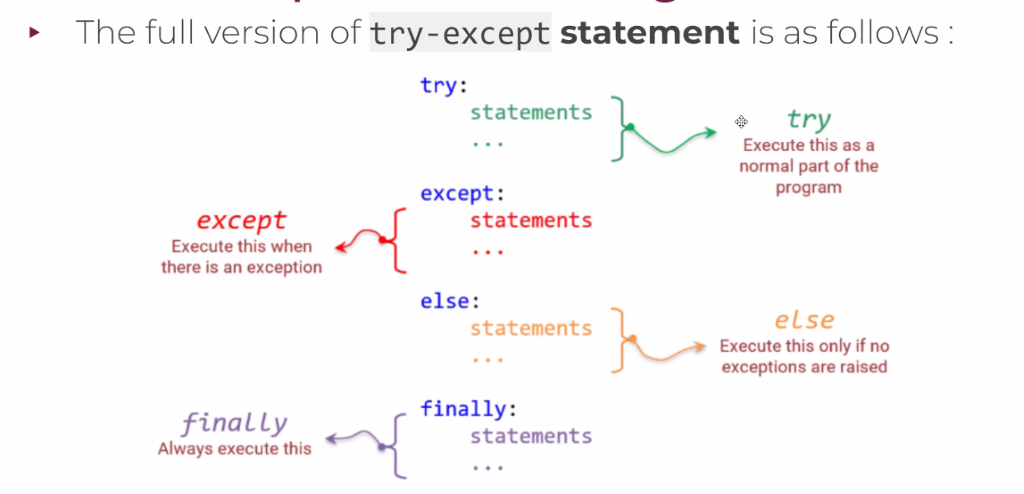
except karşısına yazarız, farklı hatalar vereceği zaman farklı kodlar alacaksak bunu yapabiliriz.



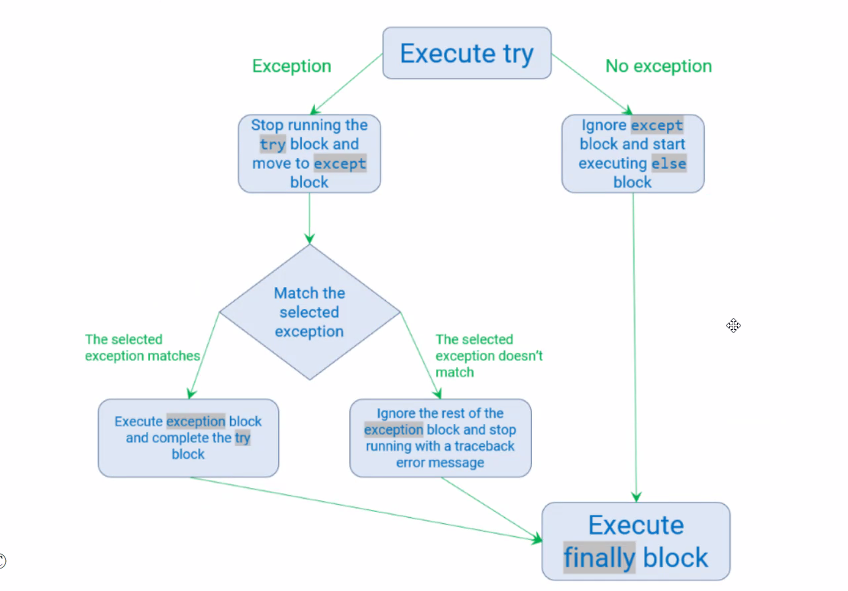


except bloğunu farklı farklı hatalar içinde alt alta yazabiliriz.

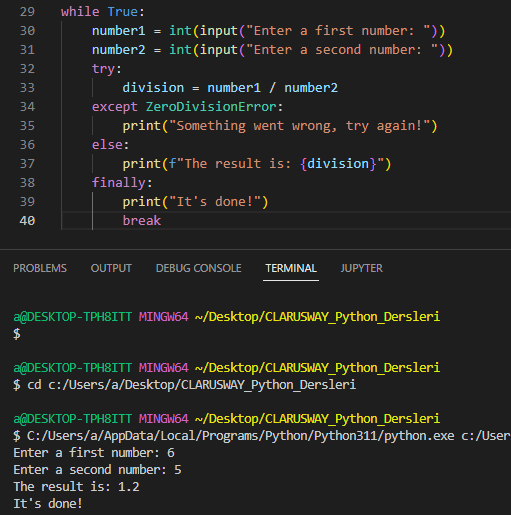
Full exception Handling Block:



Adım adım şeması:

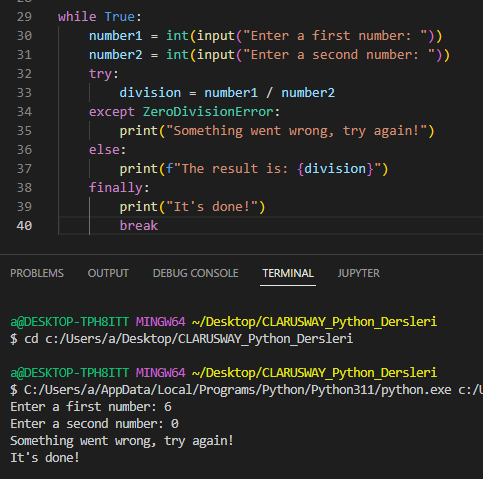


Vs Code:



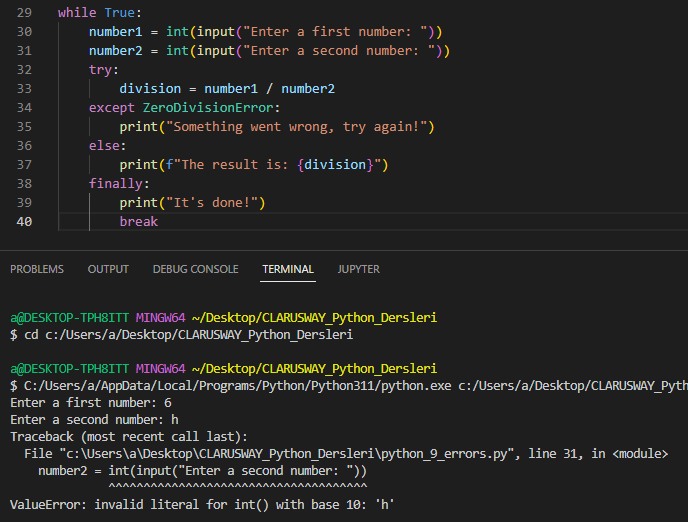
Bu kodu al try ı çalıştır, hata verirse except i çalıştır, vermezse else i çalıştır, en sonunda finally ı her koşulda çalıştır.

hata verdirelim:

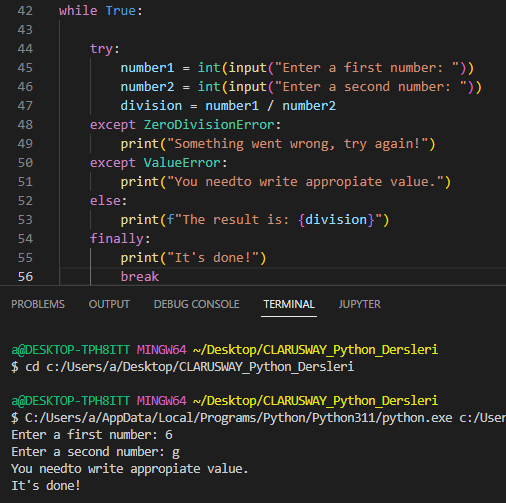


finally i yine de yazdırdı.

int yerine başka değer girdğimizde number1 ve number2 try ın dışında olduğu için altta hata veriyor:



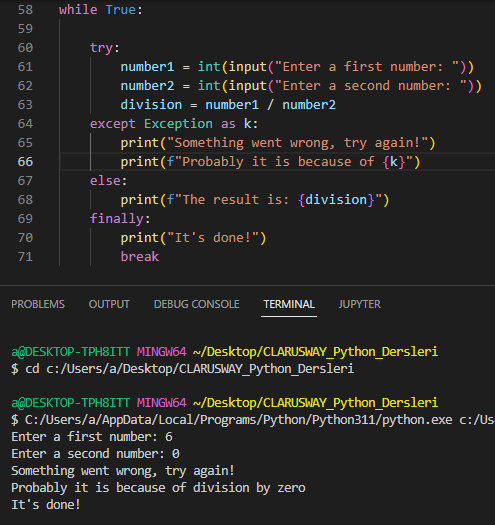
number1 ve number2 variable ları try ın altına alalım:

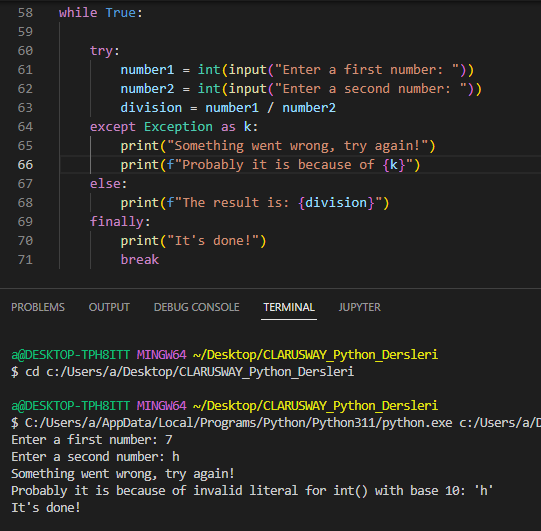


yeni bir except daha ekledik ve çıktı olarak onu aldık.

Bir de spesifik bir hata ismi vermek yerine "Exception as e" ile hatayı terminale yazdıralım:

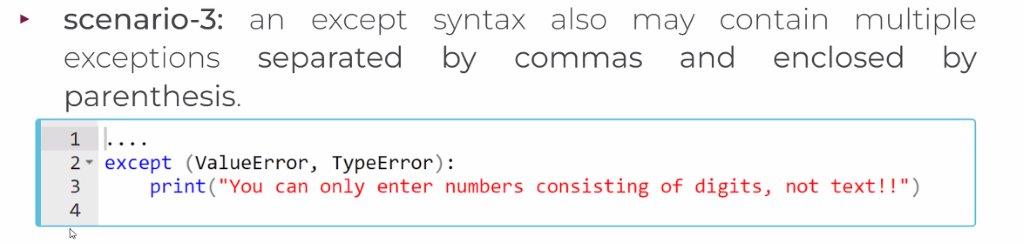
(e yerine herhangi bir harf kullanabiliriz)





hata olduğu halde kod çalıştı ve hatanın ismini istediğimiz çıktıda yazdı

except i şu şekilde de yazabilriz:



exception ArithmeticError: yazdığımızda şu hatalara bakar:



