**if\_\_name\_\_ == “\_\_main\_\_” kullanımını aynı zamanda Exception handling yaparak görelim:**

**if\_\_name\_\_ == “\_\_main\_\_” kullanımı hakkında ön bilgi:**

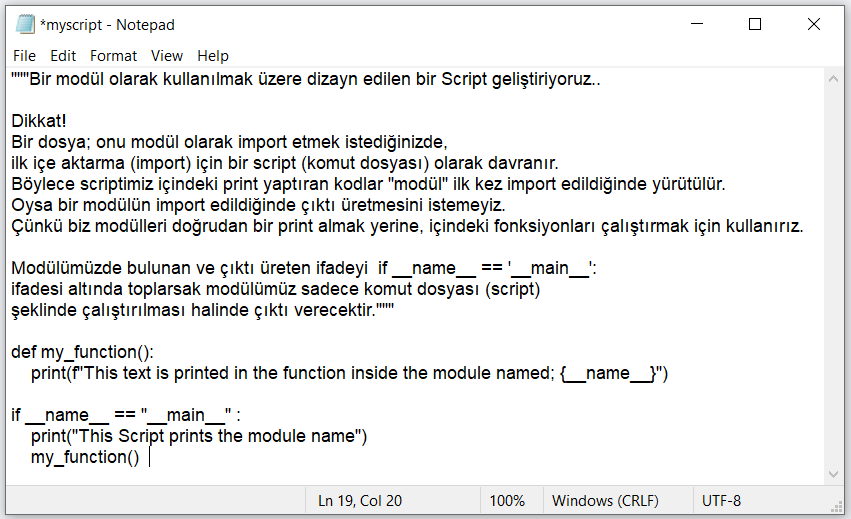
Her Python modülünde \_\_name\_\_ tanımı vardır ve bu ‘\_\_main\_\_’ ise, modülün kullanıcı tarafından bağımsız olarak çalıştırıldığı anlamına gelir ve siz bir if statement ile modülün bu durumda olması durumunda gerçekleştirmek istediğiniz eylemleri gerçekleştirebilirsiniz.

Script'i (komut dosyasını) başka bir scripte "modül" olarak import ederseniz (içe aktarırsanız) \_\_name\_\_, modül olan scriptin adına ayarlanır. Yani artık "\_\_main\_\_" e eşit olmaz.

Python dosyaları (yeniden kullanılabilir) modüller veya bağımsız programlar olarak iki türlü çalışabilir.

**Özetle**: if \_\_name\_\_="\_\_main\_\_": kodunuzun yalnızca import edilmemiş olarak dosyanın doğrudan çalıştırılmış olması halinde yürütmek için kullanılır.

1. Önce minicik bir python modülü (yani python dosyası) oluşturalım. Bunun için Windows'tan Notepad'i açıp içinde Docstring’iyle beraber şöyle küçük bir fonksiyon tanımlayalım. Bu Script’i modül olarak import etmek için geliştirdik.



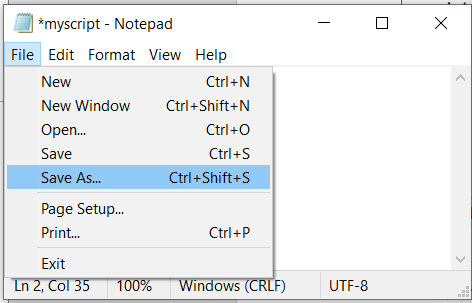
**Modüle neden if \_\_name\_\_ == '\_\_main\_\_': ifadesini ekledik?**

Bu dosya; bir modül olarak import etmek istediğinizde, ilk içe aktarma (import) için bir script (komut dosyası) olarak davranır. Bir modülün içe aktarıldığında çıktı üretmesini istemeyiz.

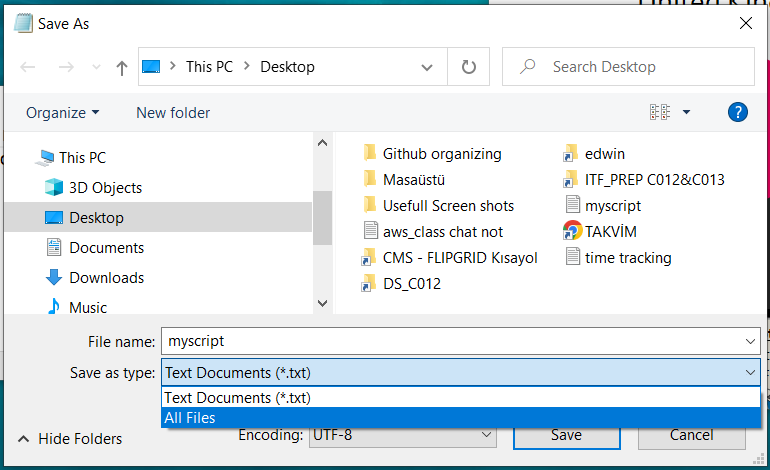
Bir Python kuralı olarak, .py uzantılı oluşturduğunuz dosya bir modül olarak içe aktarıldığında; Python, \_\_name\_\_ değişkenini modülün adına ayarlar. Ancak, dosya bir komut dosyası olarak çalıştırılırsa, \_\_name\_\_ değişkeni '\_\_main\_\_' string değerine ayarlanır.

Modülümüzde bulunan ve çıktı üreten ifadeyi if \_\_name\_\_ == '\_\_main\_\_': ifadesi altında toplarsak modülümüz sadece komut dosyası şeklinde çalıştırılması halinde çıktı verecektir.

1. Sonra Save As yapalım:



3. Dosya tipini ayarladığımız yeri All Files’a getirelim ve File name kısmında dosyamızın ismini uzantısı .py olacak şekilde düzenleyelim. (*Aksi taktirde siz dosyanızın uzantısını .py olarak değiştirseniz bile aslında dosyan myscript.py.txt adında bir text dosyası olarak kaydedilecektir*.)



4. Artık Masaüstünde notebookunuzda import ederek kullanmak üzere küçük bir modülünüz oldu. Hayırlı olsun.

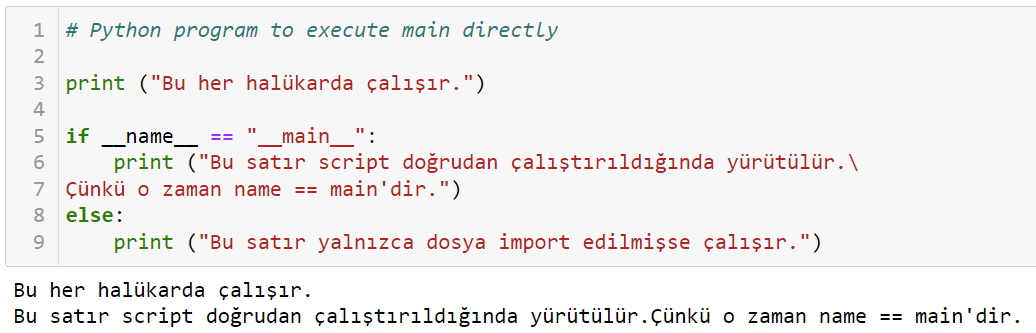
Bu modülü import edebilmek için notebookumuzu çalıştıracağımız working directory içine koymalısınız.

Çünkü Python’un import mekanizması bir modülü import etmeye çalıştığınızda onu önce working directory içinde arar. Bulamaz ise localinizde Python’un package ve modüllerinin bulunduğu directory’e giderek orada arar.



5. **DİKKAT!**

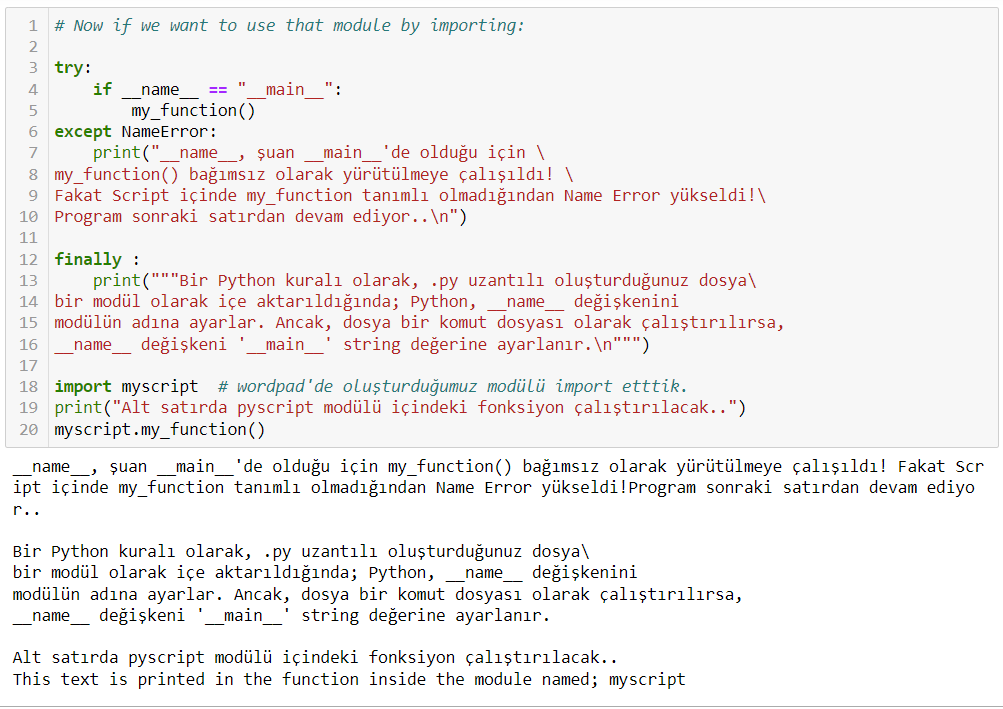
6. Şimdi if \_\_name\_\_ == "\_\_main\_\_" işlevini anlayabilmek için Jupyter’de bir notebook açarak şu kodları yazalım:



Dikkat ederseniz bu script’i herhangi bir modül import etmeden çalıştırdığımız için; import edilmesi halinde yürütülen kod bloğu çalışmadı.

7. Şimdi modülümüzün içinde tanımladığımız fonksiyonu (my\_function) önce (modülü import etmeden) direkt Script’imizin içinde (\_\_name\_\_ == “\_\_main\_\_” iken) çalıştıracağız. Sonra da myscript.py modülünü import ederek çalıştıracağız.

Fonksiyonu daha önce bu notebook’ta çalıştırmadığımız için bu aşamada yükselecek olan Name Error’u Exception Handling ile print ettireceğiz ve sonraki satırda modül içindeki fonksiyonun çalışmasını sağlayacağız.



**Çünkü:**

.py uzantılı oluşturduğumuz dosya ancak bir modül olarak içe aktarıldığında \_\_name\_\_ değişkeni modülün adına ayarlanır. Dosya bir komut dosyası olarak çalıştırıldığında ise \_\_name\_\_ değişkeni '\_\_main\_\_' string değerine ayarlanır.

Not: Outputun son satırındaki modül isminin (myscript), my\_function fonksiyonunun içinde \_\_name\_\_ attribute'ı ile yazdırıldığını hatırlayalım.

8. Gördüğünüz gibi my\_function fonksiyonunu modülden bağımsız olarak yürütmeye çalıştığımızda Name Error exception’ı aldık.

Kodumuzun modülden bağımsız olarak yürütüldüğünden **if \_\_name\_\_ == “\_\_main\_\_”:** condition’ı ile emin olduk.