PYTHON PROGRAMLAMA

(ANSYS SCRIPTING)

1. GİRİŞ

Bu notlarda Ansys içerisindeki işlemleri otomatik olarak gerçekleştirmek, değişik optimizasyon uygulamaları yapmak için nasıl programlanacağı anlatılacaktır. Bu konuda önceliklee Ansys'nin kullandığı Phyton (IronPhyton) anlatılacak.

Python genel amaçlı bir **yorumlama tabanlı** (derleme değil), interaktif, nesne tabanlı ve yüksek seviye bir programlama dilidir. Bu dil **Guido Van Rossum** tarafından 1985-1990 yıllarında geliştirilmiştir. Genel halk lisansı altında kullanılmaktadır. Okunabilirliği yüksek bir dil olarak tasarlanmıştır. Diğer dillere nazaran daha az sözdizimsel yapılara sahiptir. Özelliklerini şu şekilde sıralayabiliriz.

* **Python Yorumlanır**: Python yorumlayıcısı tarafından çalışma esnasında yorumlanarak işlenir. Çalıştırmadan önce derlemeye ihtiyaç yoktur. Bu haliyle PERL ve PHP ye benzer.

*** Python etkileşimli bir dildir**: Program yazım esnasında yorumlayıcı ile etkileşim halinde program yazılabilir.

*** Python nesne tabanlıdır**: Python nesne tabanlı stil ve teknikleri içermektedir.

*** Başlangıç seviyesi dilidir**: Python başlangıç seviyesindeki programcılar için büyük bir dildir. Basit metin işleme ile çok geniş bir yelpazede uygulama sunan bir dildir.

*** Kolay öğrenilen bir dildir**: Python bir kaç anahtar kelimeye sahiptir, basit yapısı vardır, açıkca tanımlanmış bir sözdizimine sahiptir. Bu öğrenene dili hızlıca kavrama imkanı sunar.

*** Kolay okunan bir dildir**: Kodlar açık olarak tanımlanmıştır. Gözle takibi kolaydır.

*** Geniş bir standart kütüphanesi vardır:** Python'nun kütüphanesi taşınabilir ve çapraz platformlar (Unix, Windows, Macintosh) için uygundur

*** Portatiftir**: Python geniş bir donanım platformu üzerinde çalışır. Tüm platformlarda aynı arayüze sahiptir.

*** Genişletilebilir**: Python'a düşük seviyeli modüller ekleyebilirsiniz. Bu modüller programcılara kendi araçlarını eklemeleri için imkan sağlar.

*** Veritabanları**: Phyton bütün büyük ticari databaseler için arayüzler sağlar.

*** Kullanıcı Grafik Arayüzünden (GUI) Programlama**: Python Windows, Macintosh ve Unix gibi bir çok sistemin kütüphane ve pencere sistemlerini çağırıp kullanabilir.

*** Ölçeklenebilir**: Büyük programlar oluşturmak için kabuk programlamaya nazaran daha iyi yapı ve destek sunar.

Phyton işletim sistemleri üzerinde basit ve hızlı çalışan bir programdır. Dosya işlemleri ve metinsel datalar üzerinde işlem yapmak için çok uygundur. Kullanıcı grafik arayüzlü programlar yada oyun programları için pek uygun değildir. Phyton'nun kullanımı basittir. Gerçek bir programlama dilidir. Phyton C ye nazaran daha fazla hata kontrolü sunar. Yüksek seviyeli data tipleri kullanımına imkan tanır. Örneğin esnek dizi ve klasör kullanımları gibi. Temel olarak kullanabileceğiniz geniş bir standart kod kolleksiyonuna sahiptir. Bu modüllerin bazıları I/O dosyaları, sistem çağırma, soketler, GUI (kullanıcı grafik arayüzü) araçlarıdır.

Python'nun en önemli özelliklerinden birisi Yorumlama dili olmasıdır. Bu durum program geliştirme esnasında oldukça zaman kazandırır. Çünkü hiçbir derleme ve bağlantılara gerek duymaz. Yorumlayıcı interaktif olarak kullanılabilir. Bu durum, yaz-at yada aşağıdan-yukarı doğru geliştirme gibi özellikleri denemeyi sağlar. Aynı zamanda kullanışlı bir hesap makinasıdır.

Python diğer üst seviye dillere nazaran (C,C++,Java) daha kompact ve okunabilir şekilde yazılabilir. Bunun bir kaç sebebi vardır:

* Üst seviye data tiplerini, karmaşık işlemleri tek bir satırda anlatmaya imkan tanır.

* İfadeleri gruplandırma işlemi başlangıç ve sona konulan parantezlerle değil, girinti verilerek yapılır.

* Hiç bir değişken yada argüman bildirimlerine ihtiyaç yoktur.

Python büyük-küçük harf duyarlıdır. **Adsoyad** ile **adsoyad** farklı olarak değerlendirilir. Program yazmında bloklar için {} gibi parantezler kullanılmaz. Satır başı girinti verilerek bloklar gösterilir.

IronPython ile Python Arasındaki Fark

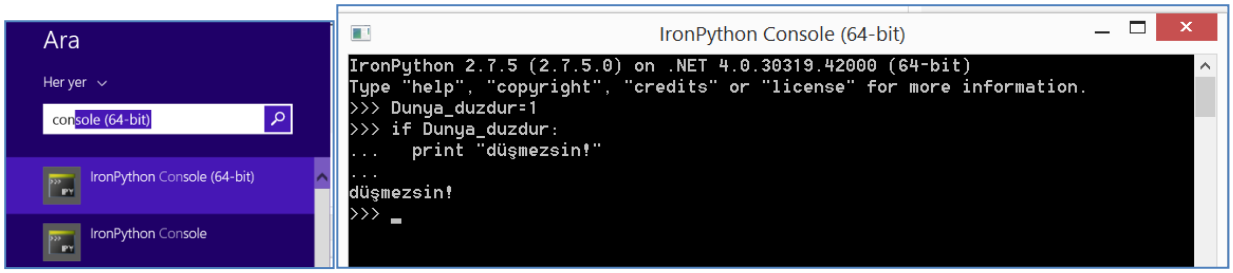
IronPython ile Python birbirinin benzeri programlar olsa da aralarında kullanım sahası olarak farklılıklar vardır. IronPython temel Python kütüphanelerini kullanabilmenin yanında .net kütüphanelerini de kullanabilir. Ayrıca IronPython en iyi görsel IDE'ye (Integrated Development Interface=Bütünleşik geliştirme arayüzü) sahiptir. Fakat IronPython ile yapılan uygulamalar sadece Windows işletim sistemlerinde çalışır. .Net Framework yüklü olmayan bilgisayarlarda çalışmaz. Python ise platformdan bağımsızdır, tüm sistemlerde çalışabilir (Windows ve Unix). Exe'ye çevrilerek Windows'ta python yorumlayıcısına gerek kalmadan direk çalışabilir. Windows ve .Net platformlarında çalışıyorsanız IronPython'u tercih etmelisiniz.

IronPhyton ile ilgili detaylı bilgiler edinmek istiyorsanız www.python.org, www.ironpython.net, www.ironpython.info https://ironpython-test.readthedocs.io/en/latest/tutorial/index.html adreslerinden faydalanabilirsiniz.

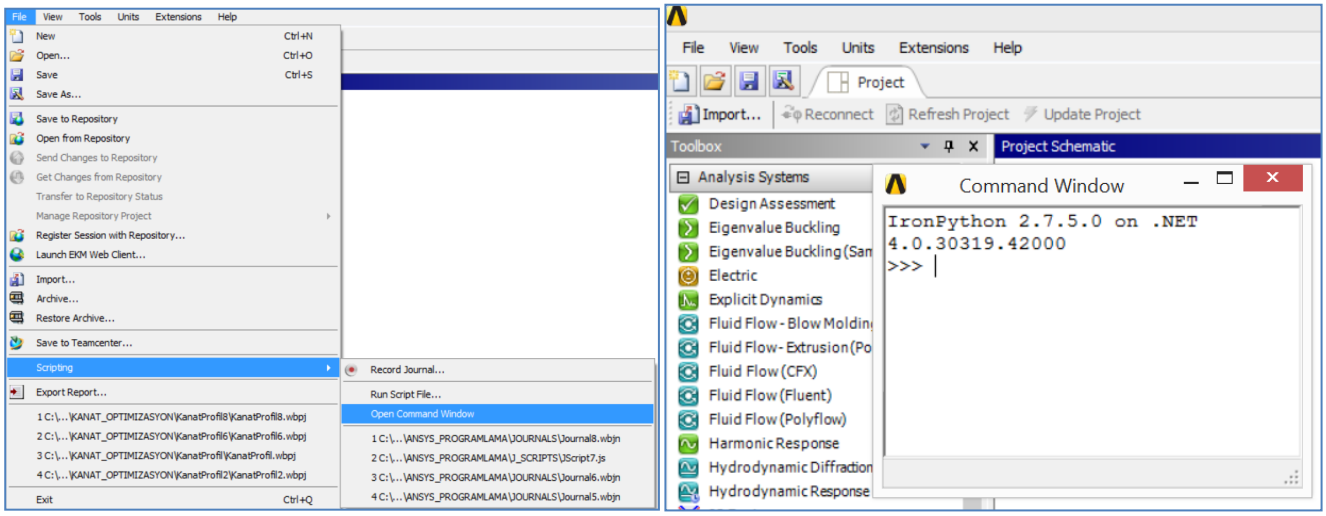
Yüklemek için https://www.python.org/downloads/windows/ adresini, ilgili dökümanlara ulaşmak için https://www.python.org/doc/ adresini kullanabilirsiniz.

Komutları Çalıştırma (Console ekranı)

Bu dokümanda IronPython komutlarını Ansys içerisinde programlama yapmak için öğreniyoruz. Bu amaçla Ansys konsol ekranını kullanacağız. Arzu edersek Windows içerisinde de IronPython komutlarını çalıştırabiliriz. Windows işletim sisteminde IronPhyton'u çalıştırmak için arama kısmına aşağıdaki gibi Console kelimesini yazarsak ona ait konsol ekranını açabiliriz.



IronPython dilini Ansys Workbench (WB) içinde kullanmak için aşağıdaki menü den ("Command Window") faydalanılır.



Komut Satırı (Prompt->>>)

Komutların yazıldığı prompt (istek satırlarında) başlangıç istek olarak (primary prompt) **">>>"** işaretleri kullanılır. Daha sonraki ikincil istek satırlarında (secondary promtp) ise **"..."** noktaları kullanılır. Kodlar yazıldıktan sonra basılan enter tuşu ile komutlar yorumlanır ve ona göre çıktı verilir. Açıklama satırları ise "#" karekteri ile başlar.

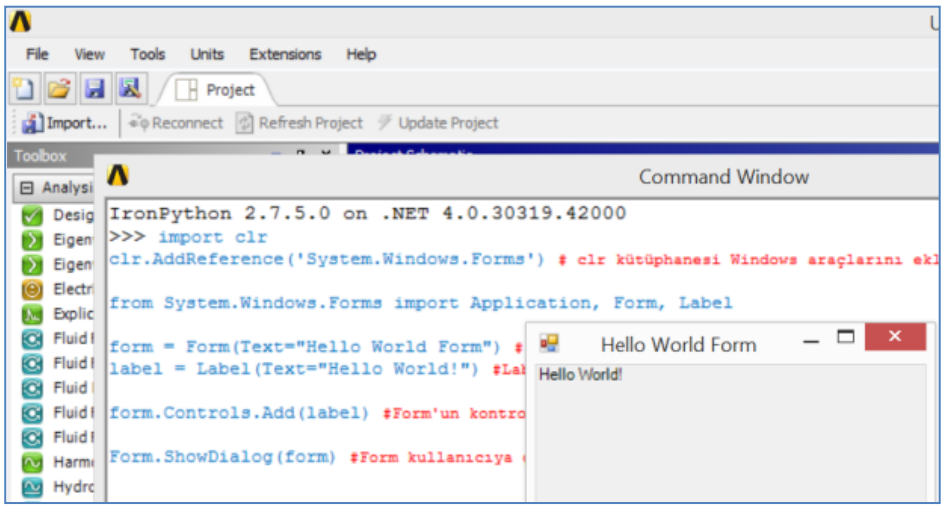
Aşağıdaki örnekte birinci satır yazıldıktan sonra entera basılmıştır. Komutla ilgili yapılacaklar bittiği için tekrar yeni bir birincil istek satırına (>>>) geçmiştir. Ardınan if komutu yazılmış ve devamı olacağı beklendiği için ikincil istek satırına (...) geçmiştir. "print..." ile devam eden satırın içerinden (en az boşluk) yazılması gerekir ki, if blok yapısı ortaya çıkmış olsun. Girinti verilmezse hata oluşabilir. print satırı yazıldıktan sonra tekrar entera basıldığında if'in devam eden komutları olabileceği için (ör:else gibi) tekrar yine (...) ikincil satır gösterilmiştir. Fakat burada yeniden entera basıldığında artık çalıştırılacak satır beklentisi kalmadığı için kodlar yorumlanıp sonuç gösterilmiştir. Bu kodları kopyalayıp denemek istediğinizde başındaki >>> ve .... işaretlerini kaldırmalısınız.

```
>>> Dunya_duzdur=1 #Bir değişken tanımlıyor ve içine true (1) ifadesini atıyor
>>> if Dunya_duzdur: #Eğer dünya düz ise yani true ise
...     print "O zaman düşmezsin!" # o zaman ekrana " " arasını yazıyor.
...   # burada entera tekrar basıldı.
O zaman düşmezsin! #Burası programın çıktısıdır.
>>> #burada yeni bir komut bekleniyor
```

Örnek: "Hello World" örneği

Aşağıdaki kodları Ansys'deki Komut satırına yapıştırıp Entera tıklayın. # Kare işaretinden sonra gelen satırları görmeyecektir. Bunlar yorum satırlarıdır.

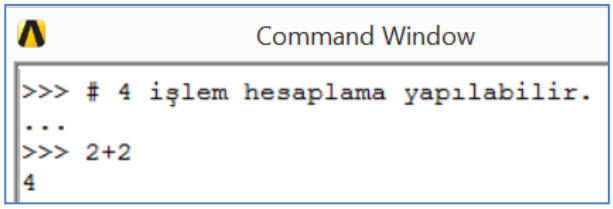
```
import clr
clr.AddReference('System.Windows.Forms') # clr kütüphanesi Windows araçlarını eklemek için
kullanılıyor.

from System.Windows.Forms import Application, Form, Label

form = Form(Text="Hello World Form") # Form nesnesi oluşturuluyor
label = Label(Text="Hello World!") #Label nesnesi oluşturuluyor.

form.Controls.Add(label) #Form'un kontrolleri arasına Label nesnesi ekleniyor.

Form.ShowDialog(form) #Form kullanıcıya gösteriliyor.
```

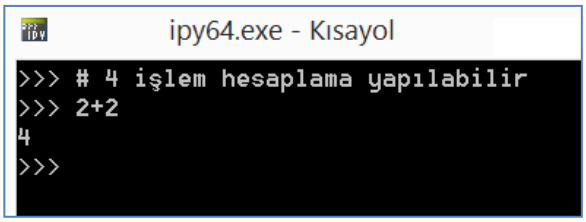


Hesap makinesi olarak kullanımı

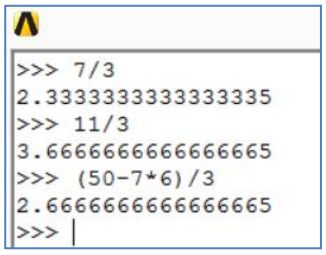
Komut satırı (promt) basit bir hesap makinası olarak kullanılabilir. Ansys'deki konsol ile IronPython (IPY) konsolundaki kullanım biraz farklı gerçekleşti. Ansys'de komut satırından sonra ikincil satır geldi (...).

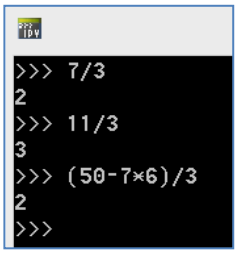
Bundan sonraki kullanımlarda Beyaz ekranlar Ansys'nin konsol ekranı, Siyahlar ise IPY konsol ekranını göstermektedir.

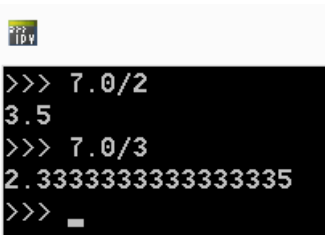




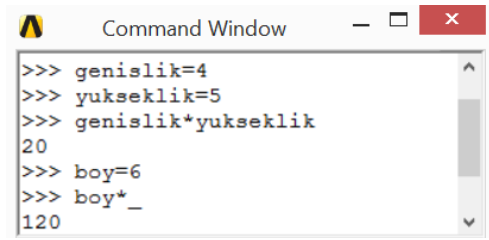
Küsüratlı bölme işlemlerinde Ansys komut satırı tam olarak sonuç verirken IPY ekranı alta yuvarlamaktadır. IPY ekranında tam sonucu göstermesi için pay kısmında double değer kullanılmalıdır. örneğin (7.0/3=2.3333) gibi.

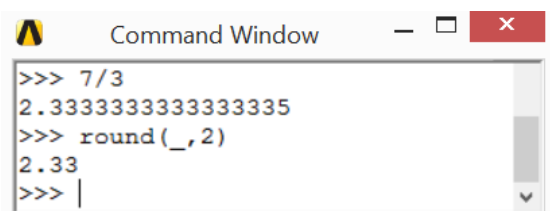






Değişkenlere atama "=" ile yapılır bunlar matematiksel işlem olarak kullanılabilir. Hesaplanan en son değer bir sonraki işlemlerde kullanılacaksa "_" işareti ile o değer tekrar kullanılabilir. Değerleri yuvarlatmak için `round(yuvarlanacak sayi, yuvarlama miktarı)` fonksiyonu kullanılabilir. Örneği inceleyin;





2. STRING İŞLEMLERİ

String ifadelerinde tek tırnak (' ') yada çift tırnak (" ") ikisi de kullanılabilir. Bu ifadeleri birleştirmek için arada + işareti kullanılabilir. * çarpım ifadesi ile çoğaltılabilir.

```
>>> kelime="karabuk" + "78" #Çift tırnaklı iki stringi birleştiriyor.
>>> kelime  #string kelime değişkeni içine atıldı. entera tıklayınca içindeki bilgi gösteriliyor.
'karabuk78'
>>> kelime='karabuk' + '78' # Tek tırnaklı stringlerde aynı şekilde çalışır.
>>> kelime
'karabuk78'  #Çıktı
>>> kelime*5  #Bir string değişkeni çarpım işlemi ile çoğaltma
'karabuk78karabuk78karabuk78karabuk78karabuk78'  #Sonuç
>>> 'karabuk''78'  #Araya '+' işareti koymadanda iki string birleşik olarak gösterilebilir.
'karabuk78'  # sonuç
```

String içerisindeki karakterler bir diz şeklinde olup bunların belli kısmını görüntülemek için aşağıdaki gibi dizi sınırları kullanılabilir.

```
>>> kelime="karabuk"  #kelime içerisine string ifade atılıyor.
>>> kelime[2] # baştan 3 karekter görüntüleniyor. 2 ci indisteki karekter
'r'  #sonuç
>>> kelime[1:3] # 1 ile 3 karekter arası görüntüleniyor. 1. indis dahil 3 indis dahil değil.
'ar'  #sonuç
>>> kelime[4:] # 4. indis ten itibaren sona kadar görüntülüyor. 4. indis dahil.
'buk'
```

Çoklu satıra ifadelerin yazımı

Komutların devamını alt satırlara yazabilmek için (\) ters slash kullanılır.

```
>>> kelime1="Karabuk"
>>> kelime2="Universitesi"
>>> kelieme3="MekatronikBolumu"
>>> cumle = kelime1 +\
kelime2 +\
kelime3
>>> print cumle
KarabukUniversitesiMekatronikBolumu
>>>
```

Eğer ifadeler [],{},() şeklinde parantezler içinde ise alt satırlara yazılabilir. Örnek;

```
>>> gunler=['Pazt', 'Sali', 'Cars',
... 'Pers', 'Cuma']
>>> print gunler
['Pazt', 'Sali', 'Cars', 'Pers', 'Cuma']
>>>
```

Tırnakların kullanımı

String ifadelerde tek tırnak (' ') çift tırnak ("...") ve üçlü tırnak ("""....""") kullanılabilir. Bunlardan üçlü tırnak alt satırada devam eden stringleri gösterirken kullanılır. Diüer durumda tek tırnak ve çift tırnak arasında bir fark yoktur.

```
>>> kelime = 'Kelime'
>>> cumle = "Bu bir cumledir"
>>> paragraf="""Bu bir coklu satıra sahip
... paragraftir"""
>>> print kelime
Kelime
>>> print cumle
Bu bir cumledir
>>> print paragraf
Bu bir coklu satıra sahip
paragraftir
```

```
>>>
```

Alt Satıra İnme

string ifadelerde alt satıra inme veya boş satır eklemek için \n işaretleri kullanılır. Örnek;

```
>>> print("\n\n Cikmak icin bir tusa basiniz")


 Cikmak icin bir tusa basiniz
>>>
```

Bir satırda çoklu ifade kullanımı

Aynı satırda çoksayıda blok ifade kullanılmak istenirse ifadelerin sonlarında noktalı virgül (;) kullanılır. Örnek;

```
>>> Universite = "Karabuk Universitesi" ; Bolum = "Mekatronik Muhendisligi" ; print
Universite + " " + Bolum; #Burada entera basıldı.
Karabuk Universitesi Mekatronik Muhendisligi #Bu ifade çıktıdır
>>>
```

3. DEĞİŞKENLER

Değişkenler değerleri saklamak için hafızada ayrılmış alanlardır. Oluşturulan her değişkenin kullanım zamanı içerisinde hafızada belli bir bölge ayrılmış demektir.

Phyton'da bir çok dilin aksine değişkenler için hafızada yer ayarmak amacıyla açık bir şekilde tanımlama yapmaya gerek yoktur. Değişkenlerin beyanı, değer atandığı esnada otomatik olarak yapılır. Eşittir işareti (=) değişkene değer atamak için kullanılır. İşaretin sol tarafındaki değişken, sağ tarafınadaki ise değeridir.

```
>>> sayac = 100      # integer atandı
mesafe   = 1000.0    # Ondalık sayı atandı
isim     = "Ahmet"    # Bir string atandı

print (sayac)
print (mesafe)
print (isim) #Burada entera tıklandı
100
1000.0
Ahmet
>>>
```

Phyton bir çok değere aynı değerin atanmasına müsaade eder. Örnek;

```
>>> a, b, c = 1, 2, "Ahmet"
>>> print (a,b,c)
(1, 2, 'Ahmet')
>>>
```

Hafızada verileri saklamak için 5 tane standart tip kullanılır. Bunlar;

- Numbers (Sayısal ifadeler)
- String (Metinsel ifadeler)
- List (Diziler)
- Tuple (Sabit diziler)
- Dictionary (.....)

Numbers (Sayılar)

Number (sayısal) veri tipi bir sayısal değer atandığında oluşturulur. del komutu ile değişken hafızadan silinebilir.

```
>>> A=1  #A değişkenine 1 atandı.
>>> B=10  #B değişkenine 10 atandı.
>>> print (A,B) #A ve B değişkenlerinin içeriği gösteriliyor.
(1, 10)  # çıktı
>>> del A,B  #A ve B değişkeni silindi
>>> print (A,B) #tekrar gösterilmeye çalışıldığında hata verdi.
UnboundNameException: name 'A' is not defined
>>>
```

Nümerik değişkenler 3 türdür. Bunlar;

- int (işaretli (+/-) tam sayılar)
- float (ondalık gerçek sayılar)
- complex (kompleks sayılar(real+imaginer))

Python'da bütün integer sayılar Long olarak tanımlanır. Bu nedenle integer'ın türünü tanımlamaya gerek yoktur. Sayısal değişkenler ile ilgili bazı örnekler verilmiştir.

int	float	complex
100	15.20	45.j
-0490	-90.	-.6545+0J
-0x260	-32.54e100	3e+26J
0x69	70.2-E12	4.53e-7j

Strings (Metin)

String ifadeler tırnaklar içinde gösterilen bir dizi karekterdir. Tek ('...') ve çift ("...") tırnakların her ikiside kullanılabilir. Stringin içerisinden bir karekteri okumak için [..] ifadesi, belli bir bölgeyi okumak içinse [.. : ..] kullanılır. Artı (+) işareti stringleri toplamak için çarpı işareti ise (*) tekrarlama yapmak için kullanılır.

```
>>> str = 'Merhaba Dünya!'

print (str)           # tüm ifadeyi yazar
print (str[0])        # ilk karakteri yazar, 0. indisli karekter
print (str[2:5])      # 3. karekter ile (2.indisli) 6. karekter arasını (6 dahil değil)
print (str[2:])       # 2. nolu indisli karekterden sona kadar yazar
print (str * 2)       # ifadeyi iki kez yazar
print (str + "Nasilsiniz") # ifadenin yanına Nasılsınız ekler
Merhaba Dünya!    #Çıktılar
M
rha
rhaba Dünya!
Merhaba Dünya!Merhaba Dünya!
Merhaba Dünya!Nasilsiniz
>>>
```

Lists (Listeler)

Listeler çok yönlü veri tipleridir. Köşeli parantez ([...]) içinde araya virgül (,) koyarak oluşturulurlar. Diğer dillerdeki dizilere benzer fakat, tek farkla. Diğer dillerde içerisindeki her data tipi aynı olmak zorundadır. Burada ise farklı tiplerde datalar liste içinde bulunabilir.

Stringlerde olduğu gibi Listelerdede içerisinden bir elemanı okumak için [..] ifadesi, belli bir aralığı okumak içinse [.. : ..] kullanılır. Artı (+) işareti listeye yeni eleman ekler, çarpı işareti ise (*) tekrarlama yaparak listeyi çoğaltır. Örneği inceleyin.

```
>>> ListeA = [ 'abcd', 786 , 2.23, 'Ali', 70.2 ]
ListeB = [123, 'Oya']

print (ListeA)          # tamamını gösterir.
print (ListeA[0])       # ilk karekterini gösterir
print (ListeA[1:3])     # 2. eleman ile 4. eleman arasını gösterir. 4 dahil değil.
print (ListeA[2:])      # 3. elemandan sona kadar gösterir.
print (ListeB * 2)      # B listesini iki kez çoğaltır
print (ListeA + ListeB) # A ve B listesini toplar, birleştirir.
['abcd', 786, 2.23, 'Ali', 70.2]  # Çıktılar
abcd
[786, 2.23]
[2.23, 'Ali', 70.2]
[123, 'Oya', 123, 'Oya']
['abcd', 786, 2.23, 'Ali', 70.2, 123, 'Oya']
>>>
```

Tubles (Sabit Listeler)

Bir diğer liste türü Tubles olarak adlandırılır. Bunlarda tıpkı Lists gibidir fakat iki farkı vardır. Burada parantez olarak köşeli parantez yerine normal parantez (...) kullanılır. Diğer bir fark ise bu dizilerin elemanları değiştirilemez, sabittir. Üzerinde yapılacak işlemlerde bir önceki örnekle aynıdır.

```
>>> ListeA = ( 'abcd', 786 , 2.23, 'Ali', 70.2 )
ListeB = (123, 'Oya')

print (ListeA)          # tamamını gösterir.
print (ListeA[0])       # ilk karekterini gösterir
print (ListeA[1:3])     # 2. karekter ile 4. karekter arasını gösterir. 4 dahil değil.
print (ListeA[2:])      # 3. karekterden sona kadar gösterir.
print (ListeB * 2)      # B listesini iki kez çoğaltır
print (ListeA + ListeB) # A ve B listesini toplar, birleştirir.
('abcd', 786, 2.23, 'Ali', 70.2)
abcd
(786, 2.23)
(2.23, 'Ali', 70.2)
(123, 'Oya', 123, 'Oya')
('abcd', 786, 2.23, 'Ali', 70.2, 123, 'Oya')
>>>
```

Kullanımı listeler ile aynı fakat bilgi eklenemediği için aşağıdaki kullanımda hata verecektir.

```
>>> listA = [ 'abcd', 786 , 2.23, 'Ali', 70.2  ]
tupleB = ( 'abcd', 786 , 2.23, 'Ali', 70.2  )

listA[2] = 1000     # List diziler için geçerli bir yazım. Listeye ekleme yapılabilir
tupleB[2] = 1000    # Tuple listeler için geçersiz bir yazım. Ekleme yapılamaz.
TypeErrorException: 'tuple' object does not support item assignment
>>>
```

Dictionary (Birleşik Diziler)

Dizi içerisindeki ifadeler hem "Degisken" hemde onun "Degeri" şeklinde sıralanırsa bu tip bir dizi oluşturulmuş olur. Örnekleri inceleyin; Elemanların belli bir sıralaması yoktur.

```
>>> diziC = {}             # Boş bir A dizisi tanımlıyor
diziC['bir'] = "Bu birdir"# 'bir' adında bir anahtar eleman ekliyor. Değeri 'Bu birdir' şeklinde
diziC[2] = "Bu ikidir" # C dizisine 2 adında anahtar bir eleman ekliyor. Değeri 'Bu ikidir' şeklinde

print (diziC['bir'])        # 'bir' elemanının değerini yazıyor.
print (diziC[2])            # 2 adındaki elemanın değerini yazıyor.
print (diziC)               # C dizisini komple yazıyor.

diziD = {'Ad': 'Ali','OkulNo':6734, 'Bolum': 'Mekatronik'}  #İçinde anahtar:degerler olan başka
bir D dizisi tanımladı
```

```
print (diziD)          # D dizisini komple yazıyor.
print (diziD.keys())   # D dizisinin anahtar elemanlarını yazıyor.
print (diziD.values()) # D dizisinin değerlerini yazıyor.

Bu birdir  # Çıktılar
Bu ikidir
{2: 'Bu ikidir', 'bir': 'Bu birdir'} #dikkat, ikinci sırada eklenen elemanı birinci sırada gösterdi
{'OkulNo': 6734, 'Ad': 'Ali', 'Bolum': 'Mekatronik'}
['OkulNo', 'Ad', 'Bolum']
[6734, 'Ali', 'Mekatronik']
>>>
```

Veri Tiplerini Dönüştürme

Bazen verit tipleri arasında dönüşüme ihtiyaç olabilir. Bu durumda aşağıdaki tabloda verilen bazı fonksiyonlar kullanılabilir.

Function	Description
int(x)	x değerini integer'a dönüştürür.
float(x)	x değerini ondalık sayıya dönüştürür.
str(x)	x object türünü stringe dönüştür.
repr(x)	x object türünü string ifadeye dönüştürür.
eval(str)	string ifadeyi object türüne dönüştürür.
tuple(x)	x ifadesini tuple türüne dönüştürür.
list(x)	x ifadesini list türüne dönüştürür.
dict(x)	x ifadesini dictionary türüne dönüştürür. x ifadesi (key, value) şeklinde olmalıdır.
chr(x)	x integer sayısını bir karaktere dönüştürür.

Bunlardan başka oct(x), hex(x), ord(x), unichr(x), frozenset(s), set(s) şeklinde diğer dönüşüm fonksiyonları da vardır.

4. TEMEL OPERATÖRLER

Operatorler, işlenen değerler üzerinde işlem yapmayı sağlar. Örneğin 4 + 5 = 9 ifadesinde 4 ve 5 işlenen, + ise operatör olarak adlandırılır.

python aşağıdaki operatör tiplerini destekler

- Aritmetik operatörler
- Karşılaştırma (ilişkisel) operatörler
- Atama operatörleri
- Mantıksal operatörler
- Bitsel operatörler
- Üyelik Operatörleri
- Kimlik Operatörleri

Operatörleri toplu olarak tek tabloda verelim.

Operator	Description
**	Üs alma

*, /, %, //	Çarpma, bölme, mod alma, alta yuvarlayarak bölme.
+, -	Toplama ve çıkarma
<=, <, >, >=	Kıyaslama operatörleri, küçük-eşit, küçük, büyük, büyük-eşit
<>, ==, !=	Eşitlik operatörleri,
=, %=, /=, //=, -=, +=, *=, **=	Atama operatörleri
is, is not	Kimlik operatörleri
in, not in	Üyelik operatörleri
not, or, and	Mantıksal operatörler

5. KARAR YAPILARI

Karar yapıları içerisinde mantıksal bir ifade bulunduran, bu ifadenin true yada false durumuna göre ilgili dal altındaki kodları çalıştıran yapılardır. Karşılaştırma yaparken Boş-olmayan yada Sıfır-olmayan değerler True sonuç üretir. Eğer değer Boş yada Sıfır ise False değer üretir.

if... yapısı

Yapısı şu şekildedir.

```
if (kosul_ifadesi):
   Çalıştırılaak_komutlar
```

Örnek 1: İçinde değer bulunan değişkenin mantıksal olarak True değere sahip olması

```
>>> degisken1 = 100
if degisken1:
   print "degisken1 mantiksal true degere sahiptir"
   print degisken1
# Cikti-------------
degisken1 mantiksal true degere sahiptir
100
>>>
```

if... elif... else yapısı

Karar içinde çok sayıda koşul varsa bu yapı kullanılır. Koşullar çoğaltılırken "elif" ifadesi kullanılır. Üstteki hiç bir koşul gerçekleşmese en son olarak "else" yapısı ile kalan durum çalıştılır. else nin yanına koşul yaşılmaz.

Örnek 1: İçindeki değer 0 olan değişkenin mantıksal olarak Frue değere sahip olması

```
>>> degisken2 = 0
if degisken2:
   print "degisken2 mantiksal True degere sahiptir=", degisken2
else:
   print "degisken2 mantiksal False degere sahiptir=", degisken2
# Cikti-------------
degisken2 mantiksal False degere sahiptir= 0
>>>
```

```
>>> puan=78
if (puan>=60 and puan<=100):  # Eğer puan 60 dan büyük ve 100 den küçükse 'Geçti' yaz.
   print("Gecti")

elif (puan>=0 and puan<60):  # Yok eğer puan 0 dan büyük ve 60 dan küçükse 'Kaldı' yaz.
   print("Kaldi")

else :  # Yukarıdaki iki şart da false ise (gerçekleşmese) 'Yanlış değer' yaz.
   print ("Yanlis değer")

Gecti :  # sonuç
>>>
```

6. DÖNGÜLER

Döngüler bir grup komutun bir çok kez çalıştırılmasını sağlar. Döngüde kullanılan koşul true olduğu müddetçe döner. False olursa döngüden çıkar.

for ... döngüsü

Sınırları yada tekrar adedi belli olan durumlarda kullanılması uygundur. Genel yapısı şu şekildedir.

```
for iterasyon_degiskeni in aralik:
   Calistirilacak_komutlar
```

Aralık belirtirken **range()** yada **xrange()** fonksiyonu kullanılır. range() fonksiyonu içerisinde sayılar bulunan list özellikli bir dizi oluşturur. xrange() fonksiyonu ise bir sayı üretecidir. Daha çok xrange() kullanılır. Bunun kullanımının tek sebebi hafıza kullanımını azaltmaktır. Az sayıda döngüde fark anlaşılmaz ama yüksek sayılı döngülerde xrange() hız farkı ortaya çıkar.

Örnek 1: Belli bir sayıda döngüyü çalıştırma

```
>>> for x in range(1,5): # 1 ile 5 arasında sayıları sırayla x atacak (5 dahil degil)
...     print (x)    # her döndüğünde x değerini yazacak.
...  #Çıktı için entera tıklandı
1
2
3
4
>>>
```

Örnek 2: İç içe döngüler

```
>>> for x in xrange(1, 3): # dikkat! xrange() fonksiyonu kullanıldı.
    for y in xrange(1, 4):
        print '%d * %d = %d' % (x, y, x*y)
# Cikti-------------
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
>>>
```

Örnek 3: Döngüden erken çıkma

```
for x in xrange(3):
```

```
    print(x)
    if x == 1:
        break
# Cikti-------------
0
1
>>>
```

Örnek 4: String ifade içindeki karekterleri listeleme

```
>>> kelime = "Merhaba"
for x in kelime:
    print x
# Cikti-------------
M
e
r
h
a
b
a
>>>
```

Örnek 5: Liste içindeki elemanları görüntüleme

```
>>> kelimeler = ['Ali', 5, 'Oya']
for x in kelimeler:
    print x
# Cikti-------------
Ali
5
Oya
>>>
```

Örnek 6: Liste içinde Listeleme

```
>>> buyukListe = [ [1, 2, 3], [4, 5, 6], [7, 8, 9]]
for kucukListe in buyukListe:
    print (kucukListe)
    for x in kucukListe:
        print x
# Cikti-------------
[1, 2, 3]
1
2
3
[4, 5, 6]
4
5
6
[7, 8, 9]
7
8
9
>>>
```

Örnek 7: İç içe for kullanımı ve mod almaya bir örnek. Ayrıca for'un else ile kullanımına da bir örnek.

```
>>> for x in range(10,20):  #10 ile 20 arasinda x degeri alacak (20 dahil değil)
   for y in range(2,x): #2 ile x degeri arasında donecek.
      if x%y == 0:      #ilk karekteri belirlemek için mod kontrolu yapiyor.
         z=x/y          #z degerini yi hesapliyor
         print '%d * %d = %d' % (y,z,x)
         break #ikinci For dan cikmak için break atiyor.
   else:                # else ikinci for'un bir parcasidir.
      print x, 'sonraki sayi'
# Cikti-------------
2 * 5 = 10
11 sonraki sayi
2 * 6 = 12
```

```
13 sonraki sayi
2 * 7 = 14
3 * 5 = 15
2 * 8 = 16
17 sonraki sayi
2 * 9 = 18
19 sonraki sayi
>>>
```

Örnek 8:

```
>>> meyveler= ['elma',  'erik', 'uzum']
for x in range(len(meyveler)): #meyveler dizisi 3 elemanli. 0,1,2 sayilari ni sirayla alacak
   print 'Suanki meyve :', meyveler[x]

print "Gule gule!"
# Cikti-------------
Suanki meyve : elma
Suanki meyve : erik
Suanki meyve : uzum
Gule gule!
>>>
```

While... döngüsü

While döngüsü sonsuz bir döngüdür. Şart gerçekleştiği sürece (koşul true iken çalışır, false olana kadar döner) dönmeye devam eder. Yapısı şu şekildedir.

```
while (kosul_ifadesi):
   calistirilacak_komutlar
```

Örnek 1: Döngünün belli sayıda dönmesini sağlamak.

```
>>> sayac = 0
while (sayac < 5):
   print 'sayac degeri:', sayac
   sayac = sayac + 1
# Cikti-------------
sayac degeri: 0
sayac degeri: 1
sayac degeri: 2
sayac degeri: 3
sayac degeri: 4
>>>
```

Örnek 2: Döngünün sonsuz olarak dönmesini sağlamak. Programdan çıkmak için **CTRL+C** basın (Ansys de çalışmaz).

```
>>> x = 1
while x == 1 :  # x daima bir olacağı için sonsuz döngü oluşturur.
   sayi = raw_input("Bir Sayi Giriniz :")
   print "Girdiginiz Sayi: ", sayi
# Cikti-------------
Bir Sayi Giriniz :11
Girdiginiz Sayi:  11
Bir Sayi Giriniz :12
Girdiginiz Sayi:  12
Bir Sayi Giriniz :13
Girdiginiz Sayi:  13
Bir Sayi Giriniz :
```

Örnek 3: Döngünün else ifadesi ile kullanımı. Döngü bittikten sonra çalıştırılır yada döngü False durumunda ise çalıştırılır. Diğer dillerde bu özellik yoktur. Döngü bittikten sonra zaten sıradaki komutlar çalıştırılacağı için yapılacak işlemler oraya yazılır.

```
sayi = 0
while sayi < 5:
   print sayi, " 5 den kucuktur"
   sayi = sayi + 1
else:
   print sayi, " 5 den kucuk degildir"
```

7. SAYILAR (NUMBERS)

Sayısal bilgileri tutan veri tipidir. Sayı nesnesi (değişkeni) ona bir değer atandığında oluşturulur. Örneğin;

```
deg1=1
deg2=10
```

Değişkeni hafızdan silmek için del ifadesi kullanılabilir.

```
del deg1
del deg_a, deg_b
```

Python 4 çeşit sayısal veri tipini destekler

- **int** (işaretli tam sayı): İçerisinde ondalık nokta bulunmayan eksi yada artı tam sayıdır. (Örn: 10, -78)
- **long** (uzun tam sayı): Sınırsız uzun tam sayıları tanımlamak için kullanılır. Sonuna küçük yada büyük L harfi kullanılır. (Örn: 519L,-2343543533L)(Büyük L harfi kullanılması tavsiye edilir, çünkü 1 rakamı ile karıştırılır).
- **float** (ondalık noktalı gerçek sayılar): İçerisinde ondalık nokta bulunan gerçek sayılardır. Bilimsel üstlü gösterminde E yada e harfi kullanılabilir. (0.0, 15.20, 32.3+e18, 70.2-E12)

Sayıları Dönüştürme

Python kendisi otomatik olarak bir ifadeyi bilinen yaygın bir tipe dönüştürür. Ama bazen belli bir tipe dönüştürme zorunluluğu olabilir. Böyle bir durumda aşağıdaki dönüşüm fonksiyonları kullanılabilir.

- **int(x)**: x ifadesini düz integer'a dönüştürür.
- **long(x)**: x ifadesini long integer'a dönüştürür.
- **float(x)**: x ifadesini noktalı sayıya (ondalık) dönüştürür.

8. NESNE TABANLI PROGRAMLAMA (object-oriented language)

Python nesne tabanlı bir dildir. Bu yüzden Sınıf (Class) ve Nesne (Object) oluşturma kolaydır. Programcılıkta önemli bir konu olduğu için öncelikle terminolojisine bir bakalım.

Class (Sınıf): Sınıfa ait herhangi bir nesnenin özelliklerini (attributes) içeren kullanıcının tanımladığı bir prototip koddur. Bu özellikler members (elemanları) ve methods (fonksiyonları) olarak geçer ve bunlara erişim sağlanırken nesneden sonra bir nokta kullanılır.

Class variable (Sınıf değişkeni): Bu değişken sınıftan türetilmiş tüm örnekler tarafından kullanılan ortak değişkendir. Class'ın içerisinde tanımlanır. Class'ın dışında onun metodları (alt fonksiyonları) tarafından tanımlanamaz. Class değişkenleri ondan türetilmiş örnekler (instance) tarafından kullanılamaz.

Instance variable (Örnek değişkeni): Bu değişken bir metodun içinde tanımlanan ve sadece sınıfın o anki örneğine ait olan değişkendir.

Instance (örnek): Belli bir sınıftan türetilmiş bağımsız bir birey nesnedir. Bir nesne hangi sınıfa ait ise onun bir örnek nesnesidir.

Method (metod): Bir sınıf içinde tanımlanan özel bir fonksiyon türüdür.

Object (Nesne): Class tarafından oluşturulan örnek bir veri yapısıdır. Yani bir kopyasıdır.

Inheritance (Miras): Bir Class'ın özelliklerinin başka bir Class'a aktarılmasıdır.

Sınıfların (class) oluşturulması

Class tanımlaması aşağıdaki kod yapısı ile oluşturulur.

```
class SinifAdi:
   'döküman açıklama stringi (documentation string)'
   class_suite
```

Class bir "documantaion string" sahiptir. Bu ifadeye erişmek için `ClassName.__doc__`. komutu kullanılır.

`class_suite` kelimesi ise class members (sınıf elemanlarını), data attributes (veri özelliklerini) ve functions (fonksiyonları) tanımlayan bir ifadedir.

Örnek: Aşağıdaki kodlar basit bir Python class örneğidir.

```
class Personel:
   'Bu kodlar tum calisanlar icin ortak bir temel siniftir'
   persSayisi = 0 # class degiskenidir. Bunun degeri bu class'in tum ornekleri (instances) arasinda paylasilir. Gerek class icerisinde, gerekse class disindan bu degere erismek icin Personel.persSayisi ifadesi kullanilir.

   def __init__(self, ad, maas): #bu metod (fonksiyon) ozel bir metoddur. Class olusturan yada baslatan metoddur. Bu sinifin yeni bir ornegi (instance) olusturuldugunda bu metod cagrilir (call).
      self.ad = ad
      self.maas = maas
      Personel.persSayisi += 1

   def persSayisi(self): #Bu metod (fonksiyon) ilk metoddan farkli olarak normal bir fonksiyondur. Bu metodlarin ilk argumani/parametresi (argument) self olmalidir. Bu metodu cagirirken bu argumani eklemeye gerek yoktur. Python tarafindan otomatik olarak eklenir.
     print "Toplam calisan %d" % Personel.persSayisi

   def persListeleme(self):
      print "Isim : ", self.ad,  ", Maas: ", self.maas
```

Örnek (Instance) nesnelerin (objects) oluşturulması

Sinifin (class) bir ornegini olusturmak icin class adini ve argumanlarini yazmak gerekir. Bu durumda calistirilan metod __init__ ozel metodudur.

```
pers1 = Personel("Ali", 2000) #Personel sinifindan ilk nesnemizi olusturduk.
pers2 = Personel("Oya", 5000) #Bu da ikinci olusturulan nesnedir.Dikkat edersek class icerisindeki ilk metod olan __init__ metoduna uygun argumanlar gonderildi. Bu metodun argumanlari (self, ad, maas) idi self ozel bir arguman oldugu icin metod tarafindan otomatik olarak atanmakta ve bu argumanin degerini gondermeye gerek yoktur. Diger iki arguman olan ad ve maas gonderilmistir. Buna gore birinci olusturulan pers1 nesnemizin icerisindeki pers1.ad=Ali, pers1.maas=2000 olacaktır. Diger nesnemizin degerleri ise pers2.ad=Oya, pers2.maas=5000 olacaktir. Personel.persSayisi class seviyesinde global bir degisken oldugu icin degeri 2 olacaktır.
```

***KODLAR ÇALIŞTIRILMADI. ÜZERİNDE DENE ***

ANSYS PROGRAMLAMA

Ansys ve programlama ile ilgili sorularınızı officehours@padtinc.com adresine gönderebilirsiniz.

Programlamanın Temelleri

Orjinal Workbench herhangi bir programlama (scripting) yada günlük tutmaya (Journaling) sahip değildir. APDL (eski klasik Ansys) den büyük oranda farklıdır. Yeni Workbench hem Jounaling hemde programlamayı kullanır. Herşeyi dosyaya kaydeden bir arayüzle yapabilirsiniz. Kendi dosyanızı oluşturabilir ve onu tekrar çalıştırabilirsiniz.

Python Tabanlı Programlama

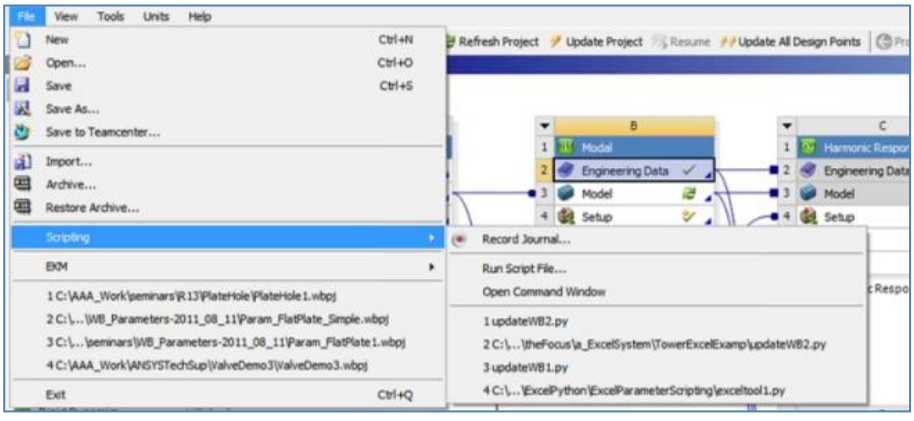
Workbench arayüzü Python programlamayı kullanır. Bu program; a) Bilgisayar Destekli Mühendislikte kullanılan kişiye özel olmayan en yaygın programlama dilidir. b) Nesne tabanlıdır. c) Dokumantasyonu iyidir. d) Tonlarca kütüphanesi ve eklentisi vardır. e) Çok geniş ve derinliktedir. f) Diğer gelişmiş kütüphane/dillerle entegre olabilir. (.Net, C, C#, C++, vs)

Eğer bir araç python'u kullanırsa, bu araç kendisiyle etkileşime geçen bir python kabuğu oluşturur. Kullanıcılar bu kabuğa python komutlarını gönderebilir ve bu kabuk vasıtasıyla programı çalıştırabilir.

Journaling (Günlük Tutma)

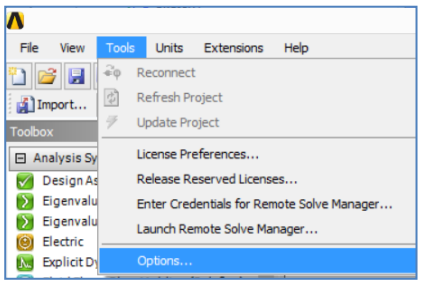
Ansys-Workbench, GUI (Graphical User Interface-kullanıcı arayüzü-Workbench kullandığımız ekranlar) vasıtasıyla yaptığımız aktiviteleri kaydetme imkanı sunar. Bu işleme "Journaling" (Günlük kaydetme) denir. Journals daha sonra tekrar oynatılabilecek bir kayıttır. Journals kayıt işlemini Python tabanlı yapar. Bu kayıt dosyasını açarak içerisinde değişiklik yada yeni eklemeler yapılabilir. Bu işleme "Scripting" (program yazma) denir. Bu özellikler size hızlı ve kolay bir şekilde tekrar eden analizler yapma imkanı sunar. Analizler "Batch Mode" da (arka planda çalıştırma-ekran görüntülerini göstermeden) çalıştırılarak tekrarlı ve hızlı bir şekilde gerçekleştirilebilir.

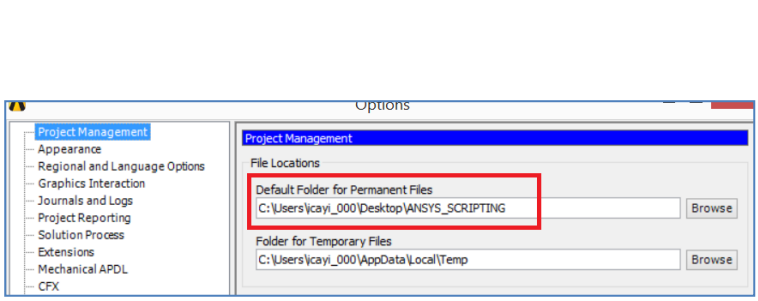
Ansys'de yapılanları Journaling dosyasına kaydetmek için şu yolu kullanabilirsiniz. File ->Scripting ->Record Journal. Açılan pencereden dosya tipi olarak (*.wbjn) seçilir. Kayıt işlemini durdurmak için ise yine aynı yerden File ->Scripting ->Stop Recording Journal

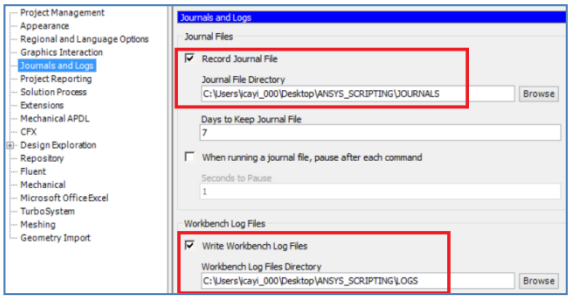


9. Varsayılan Klasörlerin Ayarlanması

Ansys'in kayıt yaptığı varsayılan klasörleri ayarlamak için Tools -> Options kısmından açılan pencereden resimlerde verilen ilgili bölümlerde ayarlama yapılır.

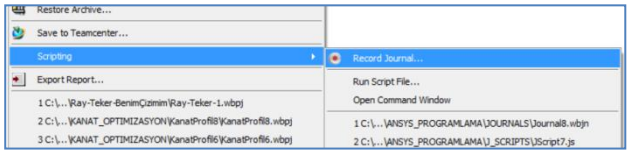


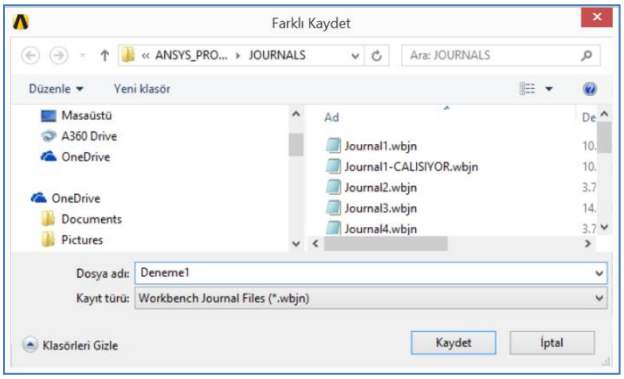




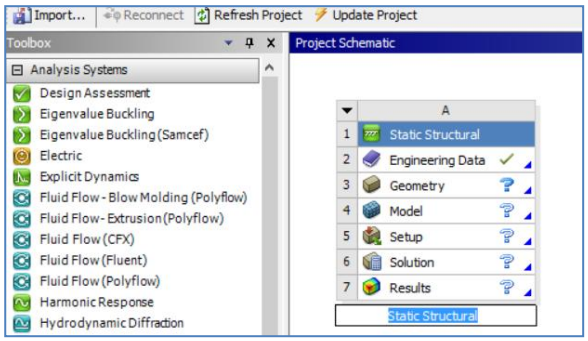
1. ÖRNEK: Malzeme özelliğini programla değiştirme (Journaling)

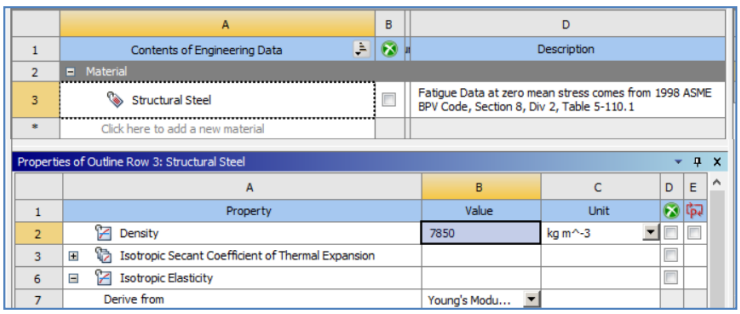
Ansys içerisindeki default olarak atanmış çelik malzemenin yoğunluğunu 8000 yapalım. Önce Journal kayıt dosyasını oluşturalım.

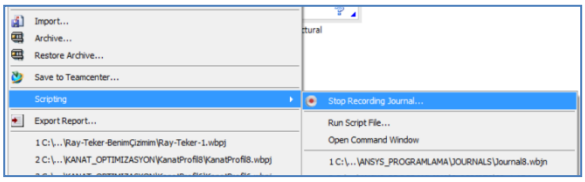


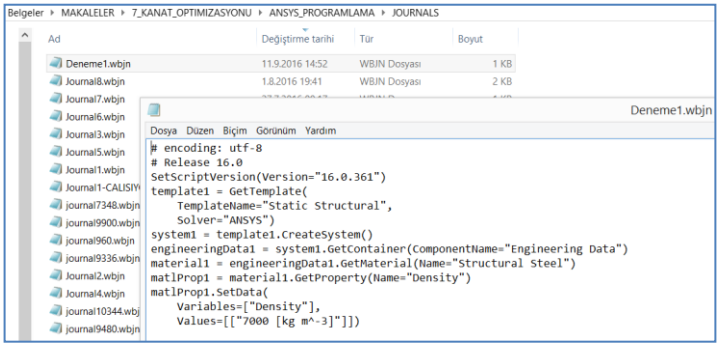


Panele modülü sürekleyip malzeme özelliğini değiştirelim. Yoğunluğu önce 7000 yapalım. Ardından Journal kayıdını durduralım. ve Journal dosyasının içeriğine bakalım.



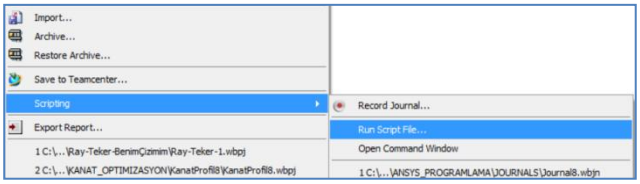


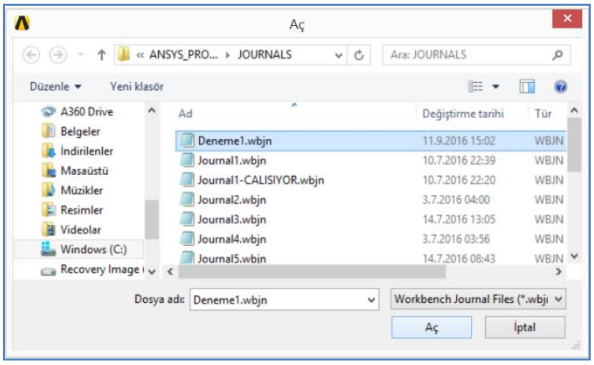


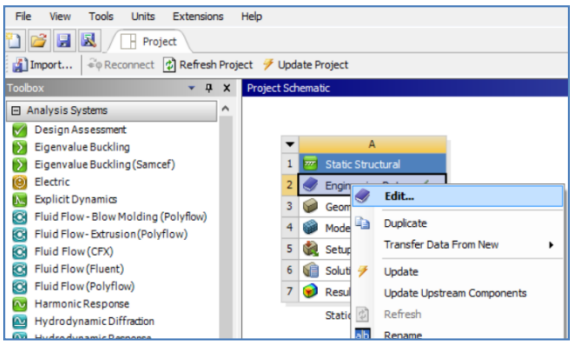


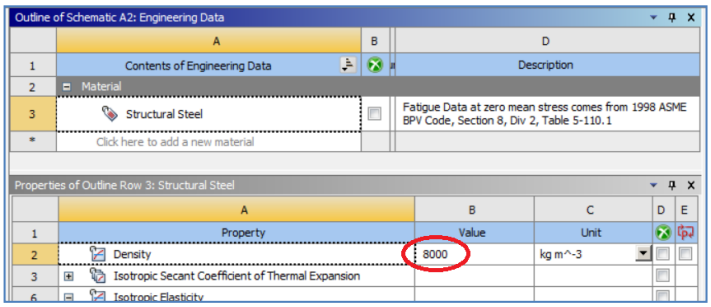

```
# encoding: utf-8
# Release 16.0    #Dikkat bu kodlar yorum gibi gözüksede bulunması gerekiyor.

SetScriptVersion(Version="16.0.361")

sablon1 = GetTemplate(TemplateName="Static Structural", Solver="ANSYS")
sistem1 = sablon1.CreateSystem()

muhendislikVerisi1 = sistem1.GetContainer(ComponentName="Engineering Data")
malzeme1 = muhendislikVerisi1.GetMaterial(Name="Structural Steel")

matProp1 = malzeme1.GetProperty(Name="Density")   #Burası matProp olarak kalmalıdır. Turkcesi
olmuyor.
matProp1.SetData(Variables=["Density"],Values=[["8000 [kg m^-3]"]])
```

Ansys kapatıp (journal dosyasınad değişiklik yapabilmek için, yada farklı isimle kaydedebilirsiniz) Not defterinde açtığımız Journal kodları içindeki yoğunluk değerini 8000 yapalım. Tekrar Ansys çalıştıralım. Şimdi tüm işlemleri Journal üzerinden gerçekleştirelim. Journal'daki komutlar çalışınca modül kendiliğinden açıldı ve malzeme özelliklerine baktığımızda yoğunluğun 8000 olduğunu görebiliyoruz.





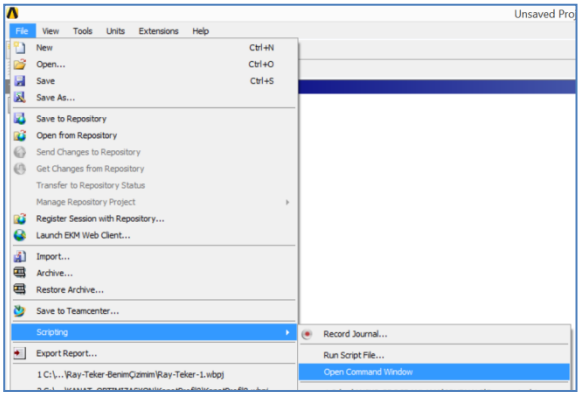




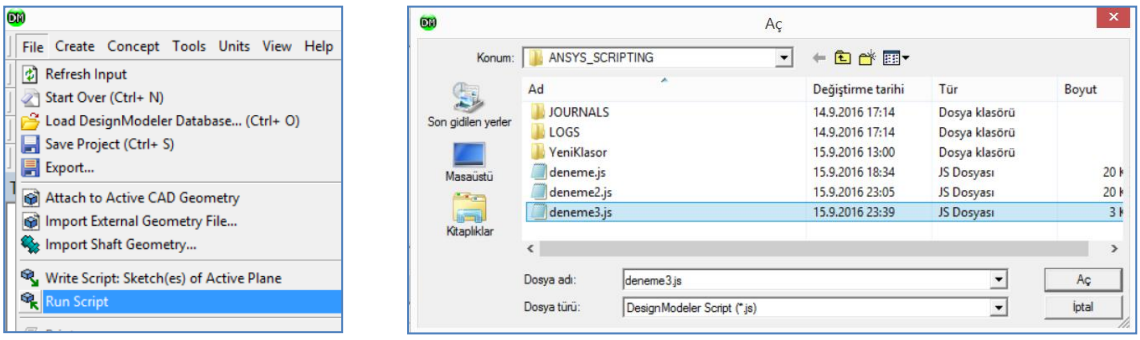
Basit bir örnek fakat proses genel olarak bu şekildedir.

IronPhyton ve JavaScript Komutlarını Çalıştırma

Daha önceden de bahsedildiği gibi Workbench içerisinde IronPhyton komutlarını çalıştırmak için File -> Scripting -> Open Command Window yolunu kullanabiliriz. Burada açılan pencerede komutlarımızı çalıştırabiliriz.



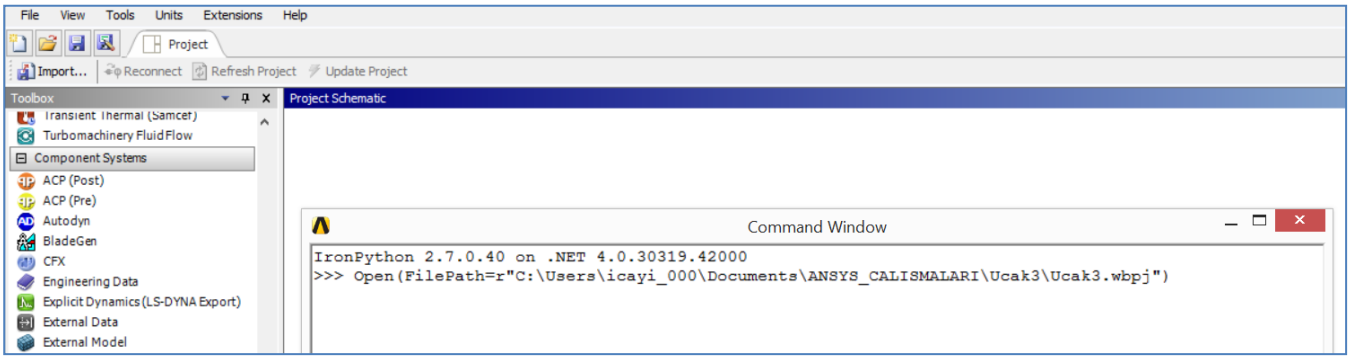
IronPyhton komutları Workbench düzeyinde çalışırken DesignModeller içerisinde ise geometriyi oluşturmak için JavaScript kodları çalıştırılır. Her ikisi ile ilgili örnekler aşağıda verilmiştir.



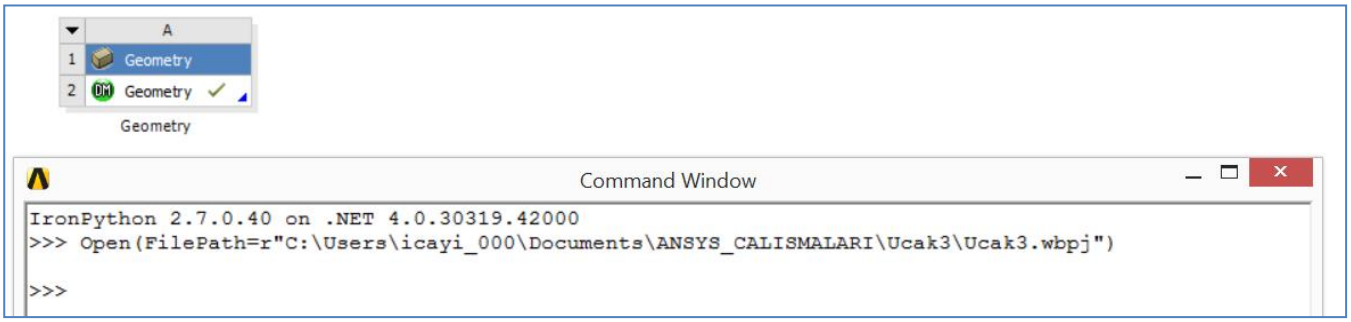
2. ÖRNEK: Ansys projesini komut satırından çalıştırma (IronPyhton)

Projenin bulunduğu yerin yolu ve adı alınır. Command penceresi açılıp buraya aşağıdaki kodlar yazılır.

```
Open(FilePath=r"C:\Users\icayi_000\Documents\ANSYS_CALISMALARI\Ucak3\Ucak3.wbpj")
```



Entera tıklayınca proje açılır ve panelde görülür.



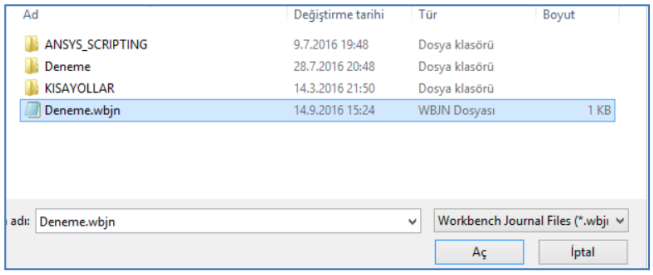
Proje yolu aşağıdaki komutla baştan ayarlanırsa her seferinde yolu yazmaktan kurtulabiliriz.

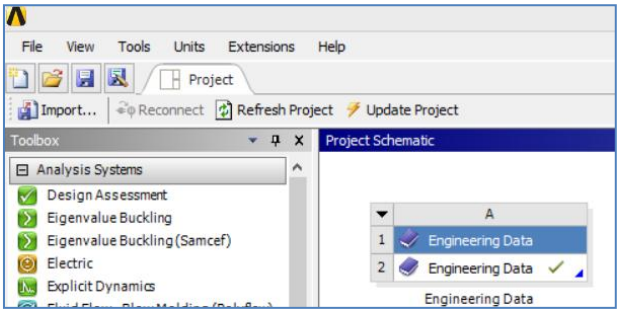
```
SetUserPathRoot(DirectoryPath = "C:\Users\icayi_000\Documents\ANSYS_CALISMALARI\Ucak3") #Yolu
atandı
Open(FilePath=AbsUserPathName("Ucak3.wbpj")) #Direk dosya adı yazıldı
```

3. ÖRNEK: Projenin birim sistemini malzeme özelliklerini ayarlama (IronPyhton)

Son satır çalışmadı. Journal dosyası olarak çalıştıralım.

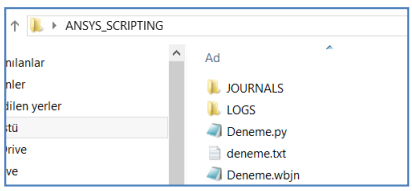
```
SetProjectUnitSystem(UnitSystemName="SI")

sicakliklar1 = [200,400,600,800,1000] #Diziye sıcaklıklar atandı
alfaDegerleri1 = [6.3e-6, 7.0e-6, 7.46e-6, 7.8e-6, 8.04e-4] #Diziye değerleri atandı

sablon1 = GetTemplate(TemplateName="EngData")
sistem1 = sablon1.CreateSystem()
MuhendislikVerisi = sistem1.GetContainer(ComponentName="Engineering Data")
celik1 = MuhendislikVerisi.GetMaterial(Name="Structural Steel")
alfa1 = celik1.GetProperty(Name="Coefficient of Thermal Expansion")

alfa1.SetData(Variables=["Tempertature","Coefficient of Thermal Expansion"], Values =
[sicakliklar1, alfaDegerleri1])
```

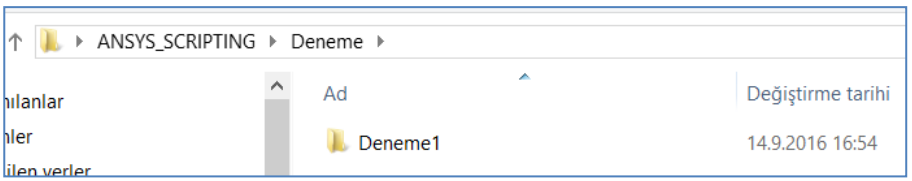




4. ÖRNEK: Otomatik dizin oluşturma (IronPyhton)

```
# encoding: utf-8
# Release 16.0    #Dikkat bu kodlar yorum gibi gözüksede bulunması gerekiyor.

#Operating system kutuphanesini yukluyor
import os

#Varsayilan dizini atiyor
SetUserPathRoot(DirectoryPath = "C:\Users\icayi_000\Desktop\ANSYS_SCRIPTING")

#varsayilan dizinin içerisine ic ice 2 tane dizin olusturacak
yeniDizin = AbsUserPathName("Deneme/Deneme1")

#Eger dizin yoksa, dizini olustur
if not os.path.exists(yeniDizin):
   os.makedirs(yeniDizin)
```

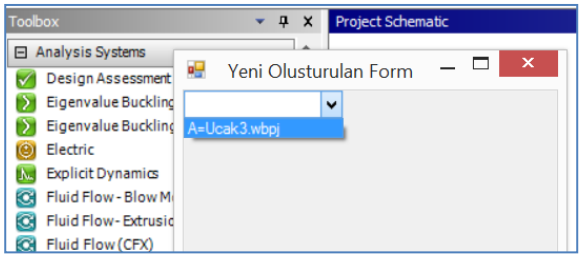




5. ÖRNEK: Klasörün içindeki dosyaları bulma ve comboBox'a ekleme (IronPyhton)

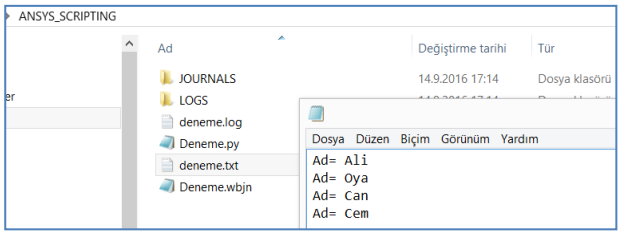
```
# encoding: utf-8
# Release 16.0
```

```
import os  #Operating-system kutuphensini ekliyor. Dizinlerle ilgili islem yapmak için

#Varsayilan dizini atiyor
SetUserPathRoot(DirectoryPath = "C:\Users\icayi_000\Documents\ANSYS_CALISMALARI")

#Varsayılan dizinin altındaki yolu gösteriyor.
BakilacakDizin = AbsUserPathName("Ucak3")

projeListesi = []

for dosyaAdi in os.listdir(BakilacakDizin):
   if dosyaAdi.endswith(".wbpj"):
      projeListesi.append(dosyaAdi)
#-----------------------------------------------------
import clr
clr.AddReference('System.Windows.Forms') #clr kütüphanesi Windows araçlarını eklemek için
kullanılıyor.

from System.Windows.Forms import Application, Form, ComboBox

form1 = Form(Text="Yeni Olusturulan Form") # Form nesnesi oluşturuluyor
comboBox1 = ComboBox()        #ComboBox nesnesi oluşturuluyor.

for x in projeListesi:
   comboBox1.Items.Add("A=" + x)

form1.Controls.Add(comboBox1) #Form'un kontrolleri arasına ComboBox nesnesi ekleniyor.
Form.ShowDialog(form1)      #Form kullanıcıya gösteriliyor.
```



6. ÖRNEK: Dosyaya bilgi yazma (IronPyhton)

```
SetUserPathRoot(DirectoryPath = "C:\Users\icayi_000\Desktop\ANSYS_SCRIPTING")
txtDosyasi = open(AbsUserPathName("deneme.txt"),"w")
isimler =["Ali","Oya","Can","Cem"]
for isim in isimler:
   txtDosyasi.write("Ad= " + isim + "\n")
txtDosyasi.close()
```

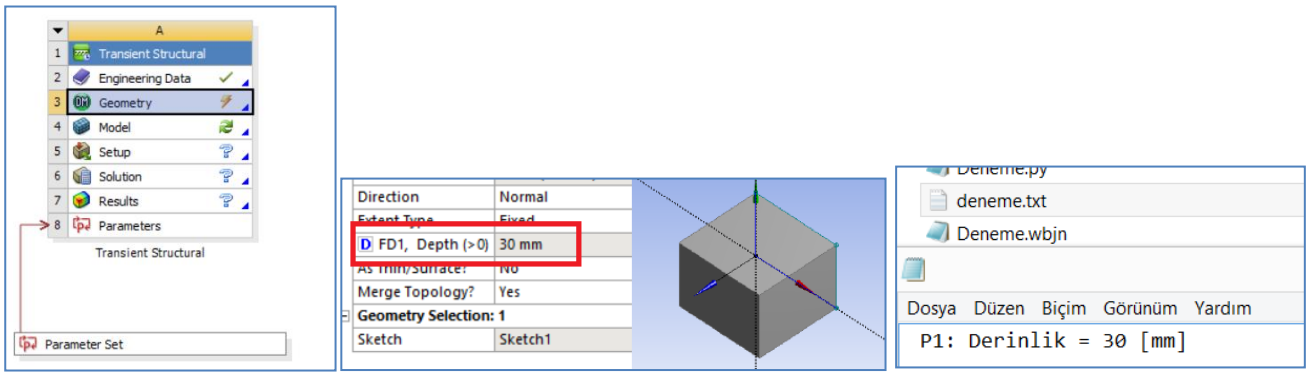


7. ÖRNEK: Projedeki parametrelerin dosyaya yazdırılması (IronPyhton)

Parametrelerin yazdırılabilmesi için öncelike projede parametre'nin oluşturulmuş olması gerekir. Aşağıdaki örnekte prizmanın Extrude derinliği parametrik olarak belirlenmiştir. Kodlar bu parametre ve değerini dosyaya yazmıştır.

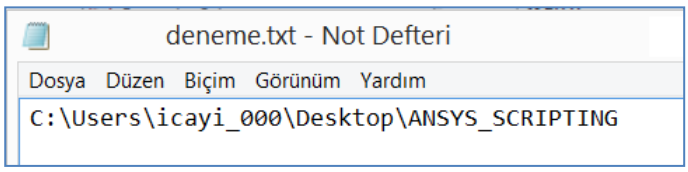
```
SetUserPathRoot(DirectoryPath = "C:\Users\icayi_000\Desktop\ANSYS_SCRIPTING")
txtDosyasi = open(AbsUserPathName("deneme.txt"),"w")
```

```
for param in Parameters.GetAllParameters():
   prmString = " " + param.Name + ": " + param.DisplayText + " = " + param.Value.ToString()
   txtDosyasi.write(prmString + "\n")
txtDosyasi.close()
```



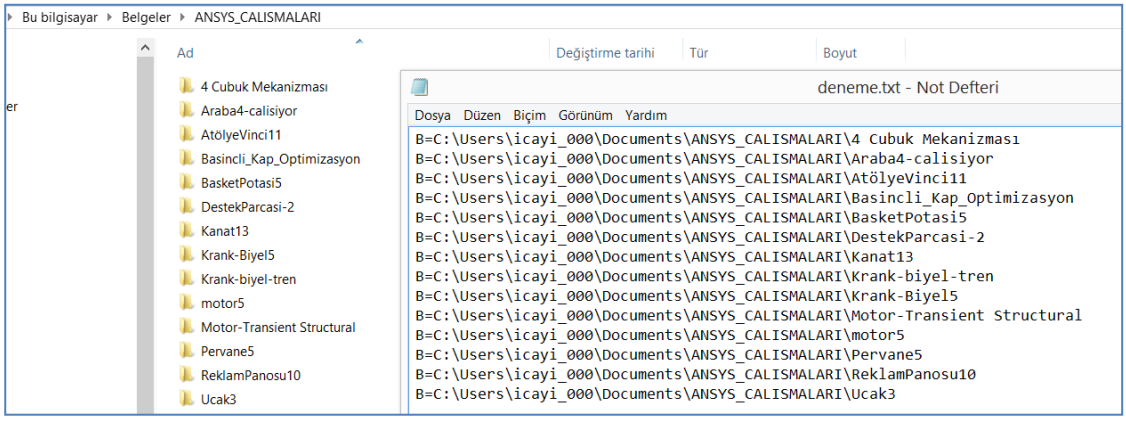
8. ÖRNEK: Varsayılan dizinin yolunu görüntüleme (IronPyhton)

```
SetUserPathRoot(DirectoryPath = "C:\Users\icayi_000\Desktop\ANSYS_SCRIPTING") #Varsayılan dizini ayarladı
txtDosyasi = open(AbsUserPathName("deneme.txt"),"w")
searchDir = GetUserPathRoot() #Varsayılan dizini görüntüledi
txtDosyasi.write(searchDir + "\n")
txtDosyasi.close()
```



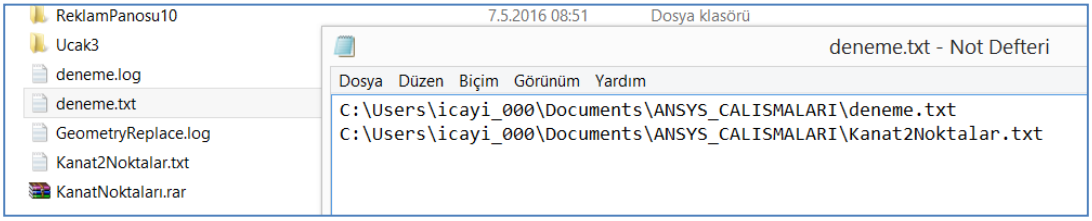
9. ÖRNEK: Dizin içindeki alt dizinleri görüntüleme (IronPyhton)

```
SetUserPathRoot(DirectoryPath = "C:\Users\icayi_000\Documents\ANSYS_CALISMALARI") #Varsayilan dizini ayarladi
aranacakDizin = GetUserPathRoot() #Varsayilan dizinin adini okuyor
txtDosyasi = open(AbsUserPathName("deneme.txt"),"w") #ciktinin kaydedilecegi dosya
dizinListesi = []  #bilgilerin tutulacagi diziyi tanimladi

for dizinAdi in os.listdir(aranacakDizin): #dizin adini okuyor
   dizintamAdi = os.path.join(aranacakDizin, dizinAdi)
   if os.path.isdir(dizintamAdi):
      dizinListesi.append("B=" + dizintamAdi)

for dizin in dizinListesi: #dosyaya yaziyor
   txtDosyasi.write(dizin + "\n")
txtDosyasi.close()
```



10. ÖRNEK: Dizin içindeki dosya adlarını görüntüleme (IronPyhton)

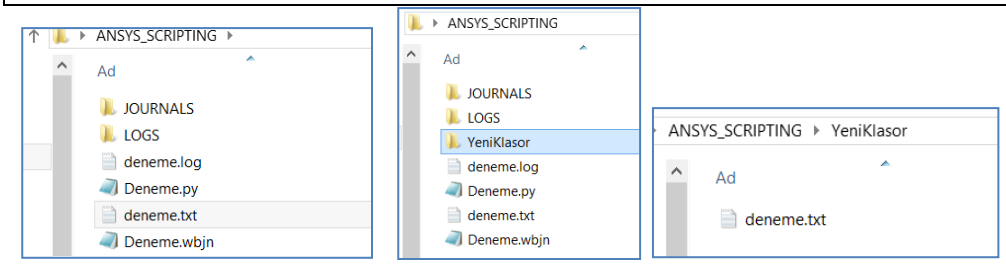
```
SetUserPathRoot(DirectoryPath = "C:\Users\icayi_000\Documents\ANSYS_CALISMALARI") #Varsayilan
dizini ayarladi
aranacakDizin = GetUserPathRoot() #Varsayilan dizinin adini okuyor
txtDosyasi = open(AbsUserPathName("deneme.txt"),"w") #dosyanin yazilacagi dosyayi belirliyor
dosyaListesi = []

for dosyaAdi in os.listdir(aranacakDizin):
   dosyatamAdi = os.path.join(aranacakDizin, dosyaAdi) #dosyanın tam adını getiriyor
   if dosyaAdi.endswith(".txt"):
      dosyaListesi.append(dosyatamAdi)

for dosya in dosyaListesi: #sonucu dosyaya yaziyor
   txtDosyasi.write(dosya + "\n")
txtDosyasi.close()
```



11. ÖRNEK: Bir dosyayı hedef klasöre kopyalama (IronPyhton)

Hedef klasöre kopyalarken o klasör orada yok ise, oluşturur.

```
CopyFile(SourceFilePath=r"C:\Users\icayi_000\Desktop\ANSYS_SCRIPTING\deneme.txt",
   DestinationDirectoryPath=r"C:\Users\icayi_000\Desktop\ANSYS_SCRIPTING\YeniKlasor",
   Overwrite=True)
```



12. ÖRNEK: Malzeme özelliklerini tablo şeklinde atama (IronPyhton)

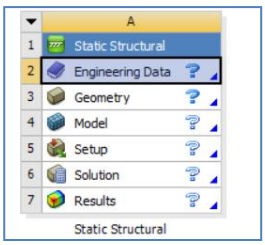
Atanacak Veriler txt dosyası içine kaydediliyor (GerilmeGerinimVerileri.txt)

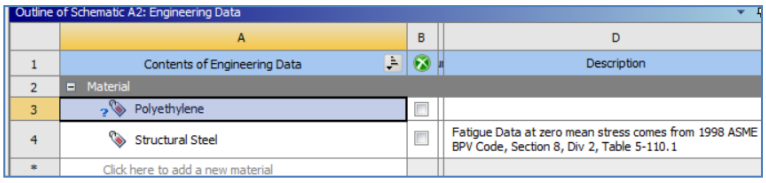
```
#
# Stress Stain Data for the Material Properties scripting example.
#
# The data is Total Strain (in m m^-1), Stress (in MPa)
#
7.33E-02, 80.6
1.80E-01, 88.0
6.30E-01, 142.5
7.53E-01, 168.0
8.70E-01, 187.0
```

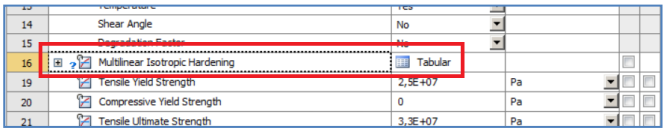
Script kodları

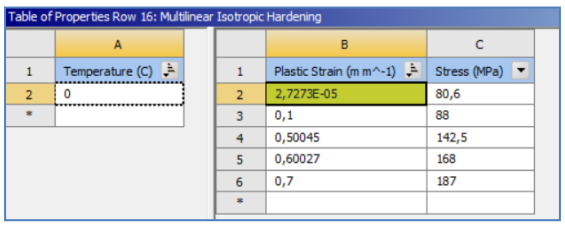
```
# "Static Structural" sistemi olusturuluyor ve onun icindeki "Engineering Data" ya erisiliyor
sablon1 = GetTemplate(TemplateName="Static Structural",Solver="ANSYS")
sistem1 = sablon1.CreateSystem()
muhendislikVerisi1 = sistem1.GetContainer(ComponentName="Engineering Data")

# "Polyethylene" malzemeyi ekliyor
```

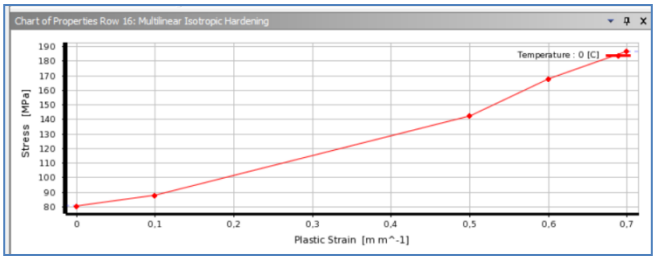
```
malzeme1 =
muhendislikVerisi1.ImportMaterial(Name="Polyethylene",Source="General_Materials.xml")

# degisken degerlerini saklamak icin dizileri olusturuyor
sicakliklar1 = []
gerinimler1 = []
gerilmeler1 = []

elastiklik1 = malzeme1.GetProperty(Name="Elasticity")
E1 = elastiklik1.GetData(Variables="Young's Modulus")

#----------------------------------------
# Sicaklik,Gerinim ve Gerilme degerlerini diziye atmak icin dosyadan okuyor.

dosyaAdi = AbsUserPathName("GerilmeGerinimVerileri.txt")
veriDosyasi = open(dosyaAdi,"r")

for satir in veriDosyasi:
    if satir.startswith("#"): #yorum satirlarini atliyor
        continue

    (gerinim2, gerilme2) = satir.split(",") #gerilme ve gerinim virgul ile ayriliyor

    gerinim2 = Quantity(float(gerinim2),"m m^-1") #degerlerin tiplerini ayarliyor (float) ve
birimlerini atiyor
    gerilme2 = Quantity(float(gerilme2),"MPa")

    sicakliklar1.append(0) #okunan degerleri diziye ekliyor. Gerinim degerini toplam gerinimi
hesaplayarak ekliyor
    gerinimler1.append(gerinim2 - gerilme2/E1)
    gerilmeler1.append(gerilme2)

# Malzeme ozelliklerini atiyor
malzemeIzotropik = malzeme1.CreateProperty(Name="Isotropic Hardening",
Definition="Multilinear")
malzemeIzotropik.SetData(SheetName="Isotropic Hardening", SheetQualifiers={"Definition
Method": "Multilinear"}, Variables = ["Temperature","Plastic Strain","Stress"], Values =
[sicakliklar1, gerinimler1, gerilmeler1])

# Projeyi kaydediyor
Save(FilePath=AbsUserPathName("deneme.wbpj"), Overwrite=True)
```







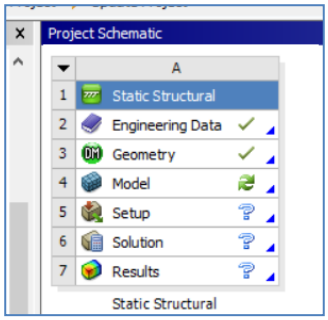


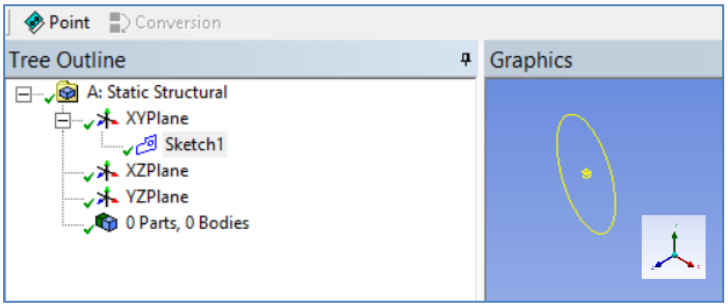


13. ÖRNEK: Sketch içerisinde Elips çizdirme (IronPyhton+JavaScript)

Pyhton Kodları içerisinde JavaScript komutlarını kullanmak için "**SendCommand**" komutu kullanılmıştır. JavaScript kodları ile skecth içerisinde elips çizdiriliyor. JavaScript kodlarının sonunda (;) vardır ve onun bölgesi 3 tane çift tırnak ile ("""...""") ile belirlenmiştir.

```
template1 = GetTemplate(TemplateName="Static Structural",Solver="ANSYS")
system1 = template1.CreateSystem()
geometry1 = GetDataContainer("Geometry")

geometry1.SendCommand(Command = """
var ps1 = new Object();
ps1.Plane = agb.GetActivePlane();
ps1.Origin = ps1.Plane.GetOrigin();
ps1.XAxis = ps1.Plane.GetXAxis();
ps1.YAxis = ps1.Plane.GetYAxis();
ps1.Sk1 = ps1.Plane.NewSketch();
ps1.Sk1.Name = "Sketch1";

with (ps1.Sk1)
{
ps1.El7 = Ellipse( 8.0, 10.0, 9.0, 6.0, 5.0, 12.0);
}
agb.Regen();
""")
```

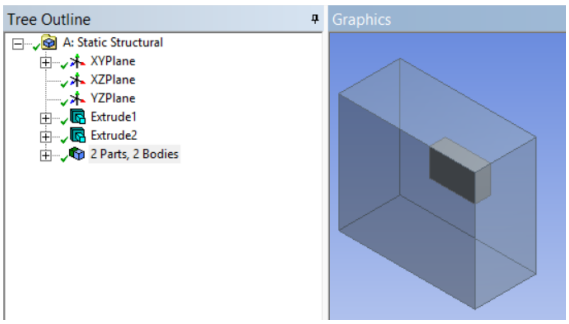




14. ÖRNEK: İç içe 2 tane küp çizdirme

```
# encoding: utf-8
template1 = GetTemplate(TemplateName="Static Structural",Solver="ANSYS")
system1 = template1.CreateSystem()
geometry1 = GetDataContainer("Geometry")

geometry1.SendCommand(Command = """
function planeSketchesOnly (p)
{

//----------------------Plane1
p.Plane  = agb.GetActivePlane();
p.Origin = p.Plane.GetOrigin();
p.XAxis  = p.Plane.GetXAxis();
p.YAxis  = p.Plane.GetYAxis();

//----------------------Sketch1
p.Sk1 = p.Plane.NewSketch();
p.Sk1.Name = "Sketch1";

//----------------------Edges1
with (p.Sk1)
{
  p.Ln1 = Line(-50, 30, 50, 30);
  p.Ln2 = Line(50, 30, 50, -30);
  p.Ln3 = Line(50, -30, -50, -30);
  p.Ln4 = Line(-50, -30, -50, 30);
}
```

```
//----------------------------Sketch2
p.Sk2 = p.Plane.NewSketch();
p.Sk2.Name = "Sketch2";

//----------------------------Edges2
with (p.Sk2)
{
  p.Ln7 = Line(-150, 130, 150, 130);
  p.Ln8 = Line(150, 130, 150, -130);
  p.Ln9 = Line(150, -130, -150, -130);
  p.Ln10 = Line(-150, -130, -150, 130);
}

p.Plane.EvalDimCons();

var ext1 = agb.Extrude(agc.Add, p.Sk1, agc.DirNormal, agc.ExtentFixed, 35.0, agc.ExtentFixed,
0.0, agc.No, 0.0, 0.0);
var ext2 = agb.Extrude(agc.Frozen, p.Sk2, agc.DirNormal, agc.ExtentFixed, 135.0,
agc.ExtentFixed, 0.0, agc.No, 0.0, 0.0);

return p;
}

var ps1 = planeSketchesOnly (new Object());

agb.Regen();

""")
geometry1.Update()
```

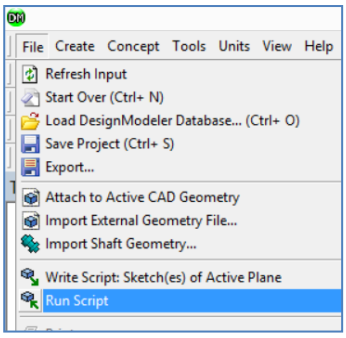


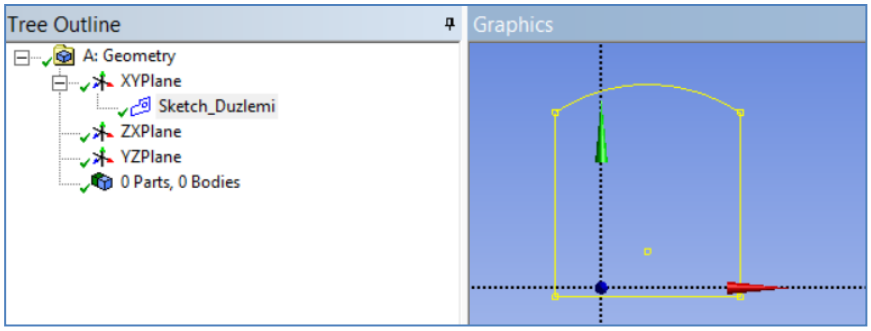
15. ÖRNEK: DesignModeller (DM) da Çizgi ve Yay çizdirme (JavaScript)

DesignModeller üzerinde çizgi çizebilmek için JavaScript kodları kullanmalıyız. Bu kodlar Not defterine yazıldıktan sonra .js adıyla biryere kaydedilir. DM açıldığında içerisindeki File -> RunScript ->DosyaAdi.js yolu kullanılarak program çalıştırılır. Koordinat sistemi (plane), sketch (çizim düzlemi) ve çizgiler kodla aşağıdaki şekilde oluşturulur.

```
// Plane ve sketch leri olusturuyor.
var plane1=agb.GetXYPlane(); //XY plane cagiriyor.
var sketch1 = plane1.NewSketch(); //plane1 düzlemi üzerinde yeni bir cizim alanı (sketch)
olusturuyor.
sketch1.Name = "Sketch_Duzlemi";  //olusturulan sketch duzleminin adini atadi.

// Noktalarin koordinatlarini tanimliyor.

var x1=-5.0;
var y1=-1.0;

var x2=15.0;
var y2=-1;

var x3=-5;
var y3=19;
```

```
var x4=15;
var y4=19;

var x=(x1+x2)/2; // yayin merkezinin x koordinati, x=5 olur
var y=4.0;  // yayin merkezinin y koordinati
var r=Math.sqrt((x4-x)*(x4-x)+(y4-y)*(y4-y)); //yayin yaricapini pisagor teoreminden hesapliyor.

//Kenarlar cizgilerini sketch uzerinde olusturuyor.

var altCizgi = sketch1.Line(x1, y1, x2, y2);
var solCizgi = sketch1.Line(x1, y1, x3, y3);
var sagCizgi = sketch1.Line(x2, y2, x4, y4);

var yay= sketch1.ArcCtrEdge(x, y, x4, y4, x3, y3); //yayin merkez noktasi, baslangic ve bitis
noktalari tanimlandi

agb.Regen();
```



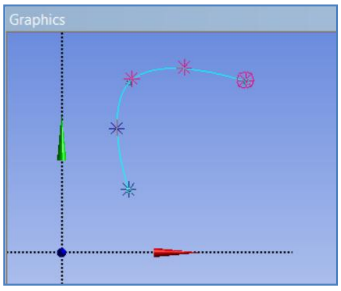


16. ÖRNEK: Spline çizimi (JavaScript)

Aşağıdaki JavaScript komutları DesignModeller->File->RunScript yolu kullanarak çalıştırın.

```
function CizimFonksiyonu (p)
{
//Plane (duzlem) olusturuyor.
p.Plane  = agb.GetActivePlane();
p.Origin = p.Plane.GetOrigin();
p.XAxis  = p.Plane.GetXAxis();
p.YAxis  = p.Plane.GetYAxis();

//Sketch (Cizim alanını) olusturuyor.
p.Sk10 = p.Plane.NewSketch();
p.Sk10.Name = "Sketch10";

//Kontrol noktalarinin koordinaltarini degisken olarak belirledi
var x1=14.54908802;
var y1=13.44270761;

var x2=14.65067138;
var y2=36.80687665;

var x3=39.84334394;
var y3=37.01004335;

//Edges (Cizgileri) olusturuyor
with (p.Sk10)
{
 //Construction Points (olusturma  noktalari) tanımladı.
  p.Pt48 = ConstructionPoint(x1, y1);
  p.Pt49 = ConstructionPoint(x2, y2);
  p.Pt50 = ConstructionPoint(x3, y3);

 //Spline (egri) cizimine geciyor
  p.Sp23 = SplineBegin();
```

```
  with(p.Sp23)
  {
    SplineXYW(x1, y1, 1.00000000);  //formati: SplineXYW(x,y,kalinlik)
    SplineXYW(12.16642630, 20.85741538, 1.00000000);
    SplineXYW(8.00167038, 44.07794833, 1.00000000);
    SplineXYW(31.87815208, 39.67073448, 1.00000000);
    SplineXYW(x3, y3, 1.00000000);

    SplineKnot(0.00000000);  //formati: SplineKnot(dugum)
    SplineKnot(0.00000000);
    SplineKnot(0.00000000);
    SplineKnot(0.00000000);
    SplineKnot(0.48116576);
    SplineKnot(1.00000000);
    SplineKnot(1.00000000);
    SplineKnot(1.00000000);
    SplineKnot(1.00000000);

    SplineCtrlPtEnd(0.00000000, 1.00000000, 4, 0, 0, 0);  //formati:
SplineCtrlPtEnd(baslangicParametresi, sonParametre, sirasi, kapali, dönel, periyodik)

    SplineFlexibility = agc.Yes;
  }
}

//constraints (sabitler)
with (p.Plane)
{
  CoincidentCon(p.Sp23.Base, x1, y1, p.Pt48, x1, y1);
  CoincidentCon(p.Sp23.End, x3, y3, p.Pt50, x3, y3);

  CoincidentConLock(p.Sp23, x1, y1, p.Pt48, x1, y1, 1);
  CoincidentConLock(p.Sp23, x2, y2, p.Pt49, x2, y2, 1);
  CoincidentConLock(p.Sp23, x3, y3, p.Pt50, x3, y3, 1);
}

p.Plane.EvalDimCons(); //son olarak duzlem uzerindeki butun boyut ve sabitleri degerlendiriyor

return p; //fonksiyon geri yine aldigi p nesnesini gonderiyor.
}

//Cizimi yapacak alt fonksiyonu cagiriyor.
var ps1 = CizimFonksiyonu (new Object());

//son
agb.Regen();
```



Aynı kodları Anahtar noktalar tanımlamadan (direk koordinat girerek) yazalım.

```
function planeSketchesOnly (p)
{

p.Plane  = agb.GetActivePlane();
p.Origin = p.Plane.GetOrigin();
p.XAxis  = p.Plane.GetXAxis();
p.YAxis  = p.Plane.GetYAxis();
```

```
//Sketch
p.Sk4 = p.Plane.NewSketch();
p.Sk4.Name = "Sketch4";

//Points
var x1=17;
var y1=16;

var x2=10;
var y2=23;

var x3=16;
var y3=45;

var x4=39;
var y4=42;

var x5=47;
var y5=39;

//Edges
with (p.Sk4)
{
  p.Sp10 = SplineBegin();

  with(p.Sp10)
  {
    SplineXYW(x1, y1, 1.00000000);
    SplineXYW(x2, y2, 1.00000000);
    SplineXYW(x3, y3, 1.00000000);
    SplineXYW(x4, y4, 1.00000000);
    SplineXYW(x5, y5, 1.00000000);

    SplineKnot(0.00000000);
    SplineKnot(0.00000000);
    SplineKnot(0.00000000);
    SplineKnot(0.00000000);
    SplineKnot(0.45516103);
    SplineKnot(1.00000000);
    SplineKnot(1.00000000);
    SplineKnot(1.00000000);
    SplineKnot(1.00000000);

    SplineCtrlPtEnd(0.00000000, 1.00000000, 4, 0, 0, 0);
    SplineFlexibility = agc.Yes;
  }
}

p.Plane.EvalDimCons();

return p;
}

var ps1 = planeSketchesOnly (new Object());

agb.Regen();
```

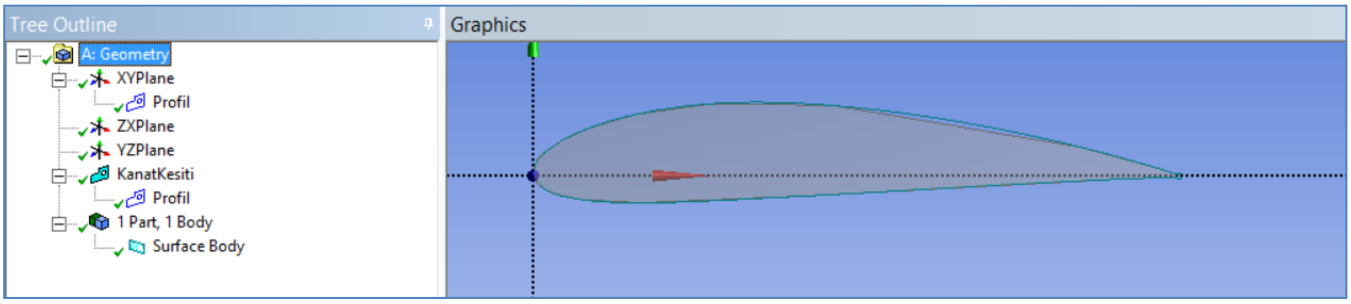
17. ÖRNEK: Spline ve Surface olusturma (JavaScript)

Aşağıda 40 tane noktadan oluşan bir kanat profili spline komutu ile oluşturulmuştur ve içerisi yüzey olarak tanımlanmıştır. Burada spline oluşturma tekniği yukarıdaki örenklerden farklıdır.

```
ag.m.ClearAllErrors();
ag.gui.setUnits(ag.c.UnitMillimeter, ag.c.UnitDegree, ag.c.No);

function CizimFonksiyonu(p)
{
  //----------Koordinat Sistemini Ayarliyor----------
```

```
    p.Plane = agb.GetActivePlane();
    p.Origin = p.Plane.GetOrigin();
    p.XAxis = p.Plane.GetXAxis();
    p.YAxis = p.Plane.GetYAxis();

  //----------Sketch Olusturuyor ve Adini Profil veriyor----------
    p.Sk1 = p.Plane.NewSketch();
    p.Sk1.Name = "Profil";

  //----------Sketch Olusturuyor----------
    with (p.Sk1)
    {
   //----------Spline Olusturuyor----------
        p.Sp1 = SplineBegin();
        with (p.Sp1)
        {
            SplineFlexibility = agc.Yes; //Buradaki spline kapali dongu. Ucu acik olsaydi esnek
yapilabilecekti. Yani bir noktasini cekince gerisi ona göre ayarlanacaktir
             SplineXY(1.0000000000,0.0000000000);
             SplineXY(0.9941909457,0.0034443422);
             SplineXY(0.9762644498,0.0089525701);
             SplineXY(0.9468056337,0.0176798496);
             SplineXY(0.9064463270,0.0290036332);
             SplineXY(0.8560738767,0.0421506502);
             SplineXY(0.7968252313,0.0562643111);
             SplineXY(0.7300720100,0.0704606995);
             SplineXY(0.6573966117,0.0838679746);
             SplineXY(0.5805610801,0.0956533030);
             SplineXY(0.5014701995,0.1050478667);
             SplineXY(0.4221285534,0.1113815755);
             SplineXY(0.3434626760,0.1136979610);
             SplineXY(0.2682640305,0.1106278449);
             SplineXY(0.1991460457,0.1024074256);
             SplineXY(0.1381012612,0.0897548199);
             SplineXY(0.0868274179,0.0737240165);
             SplineXY(0.0466519884,0.0555674396);
             SplineXY(0.0184988244,0.0365553796);
             SplineXY(0.0028945720,0.0177828502);
             SplineXY(0.0000000000,0.0000000000);
             SplineXY(0.0164708748,-0.0204615910);
             SplineXY(0.0419484020,-0.0307583977);
             SplineXY(0.0779358751,-0.0375744924);
             SplineXY(0.1234161612,-0.0411497131);
             SplineXY(0.1772540651,-0.0418875255);
             SplineXY(0.2382427318,-0.0403737798);
             SplineXY(0.3051191235,-0.0373591222);
             SplineXY(0.3765503309,-0.0336972848);
             SplineXY(0.4521273682,-0.0298757348);
             SplineXY(0.5295399342,-0.0254591443);
             SplineXY(0.6064679100,-0.0208983254);
             SplineXY(0.6810562717,-0.0165856201);
             SplineXY(0.7514644539,-0.0127586173);
             SplineXY(0.8159169236,-0.0095192837);
             SplineXY(0.8727539388,-0.0068773511);
             SplineXY(0.9204809134,-0.0047995127);
             SplineXY(0.9578162855,-0.0032481123);
             SplineXY(0.9837374474,-0.0022006917);
             SplineXY(0.9975224016,-0.0016512322);
             SplineXY(1.0000000000,0.0000000000);
            SplineFitPtEnd(); //Spline dogrulamasini yapiyor
        }
    }

    ag.selectedFeature = ag.gui.TreeviewFeature(p.Sk1.Name, 0);

    var Surf1 = ag.gui.CreateSurfSk();  //sketch den yuzey olusturuyor
    Surf1.Name = "KanatKesiti";
    Surf1.Operation = ag.c.Frozen;
    Surf1.WithPlaneNormal = ag.c.Yes;
```

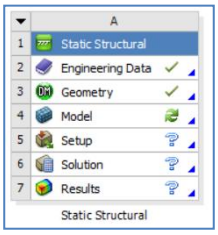
```
    p.Plane.EvalDimCons(); //plane nin boyut ve sabitlenirini dogruluyor

    return p; //plane nesnesini geri gonderiyor
}

var DuzlemYuzey1 = CizimFonksiyonu(new Object());

agb.Regen();
```

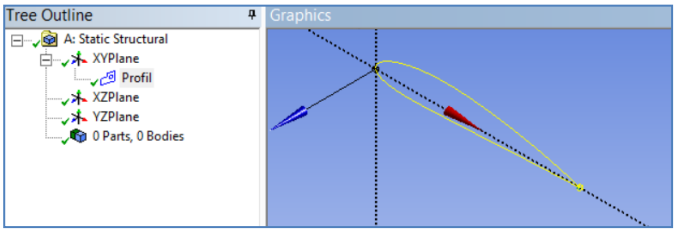


18. ÖRNEK: IronPython ve JavaScript iç içe Spline oluşturma

```
template1 = GetTemplate(TemplateName="Static Structural",Solver="ANSYS")
system1 = template1.CreateSystem()
geometry1 = GetDataContainer("Geometry")

geometry1.SendCommand(Command = """

function CizimFonksiyonu(p)
{
    p.Plane = agb.GetActivePlane();
    p.Origin = p.Plane.GetOrigin();
    p.XAxis = p.Plane.GetXAxis();
    p.YAxis = p.Plane.GetYAxis();

    p.Sk1 = p.Plane.NewSketch();
    p.Sk1.Name = "Profil";

    with (p.Sk1)
    {
        p.Sp1 = SplineBegin();
        with (p.Sp1)
        {
            SplineFlexibility = agc.Yes;
             SplineXY(1.0000000000,0.0000000000);
             SplineXY(0.9941909457,0.0034443422);
             SplineXY(0.9762644498,0.0089525701);
             SplineXY(0.9468056337,0.0176798496);
             SplineXY(0.9064463270,0.0290036332);
             SplineXY(0.8560738767,0.0421506502);
             SplineXY(0.7968252313,0.0562643111);
             SplineXY(0.7300720100,0.0704606995);
             SplineXY(0.6573966117,0.0838679746);
             SplineXY(0.5805610801,0.0956533030);
             SplineXY(0.5014701995,0.1050478667);
             SplineXY(0.4221285534,0.1113815755);
             SplineXY(0.3434626760,0.1136979610);
             SplineXY(0.2682640305,0.1106278449);
             SplineXY(0.1991460457,0.1024074256);
             SplineXY(0.1381012612,0.0897548199);
             SplineXY(0.0868274179,0.0737240165);
             SplineXY(0.0466519884,0.0555674396);
             SplineXY(0.0184988244,0.0365553796);
             SplineXY(0.0028945720,0.0177828502);
             SplineXY(0.0000000000,0.0000000000);
```

```
                SplineXY(0.0164708748,-0.0204615910);
                SplineXY(0.0419484020,-0.0307583977);
                SplineXY(0.0779358751,-0.0375744924);
                SplineXY(0.1234161612,-0.0411497131);
                SplineXY(0.1772540651,-0.0418875255);
                SplineXY(0.2382427318,-0.0403737798);
                SplineXY(0.3051191235,-0.0373591222);
                SplineXY(0.3765503309,-0.0336972848);
                SplineXY(0.4521273682,-0.0298757348);
                SplineXY(0.5295399342,-0.0254591443);
                SplineXY(0.6064679100,-0.0208983254);
                SplineXY(0.6810562717,-0.0165856201);
                SplineXY(0.7514644539,-0.0127586173);
                SplineXY(0.8159169236,-0.0095192837);
                SplineXY(0.8727539388,-0.0068773511);
                SplineXY(0.9204809134,-0.0047995127);
                SplineXY(0.9578162855,-0.0032481123);
                SplineXY(0.9837374474,-0.0022006917);
                SplineXY(0.9975224016,-0.0016512322);
                SplineXY(1.0000000000,0.0000000000);
            SplineFitPtEnd();
        }
    }

    p.Plane.EvalDimCons();

    return p;
}

var DuzlemYuzey1 = CizimFonksiyonu(new Object());

agb.Regen();

""")
```





19. ÖRNEK: Aktif Koordinat sistemi yada kendi belirlediğimiz koordinat sistemi üzerinde sketch oluşturma

Aktif plane üzerinde sketch oluşturmak için

```
p.Plane  = agb.GetActivePlane();
p.Origin = p.Plane.GetOrigin();
p.XAxis  = p.Plane.GetXAxis();
p.YAxis  = p.Plane.GetYAxis();
```

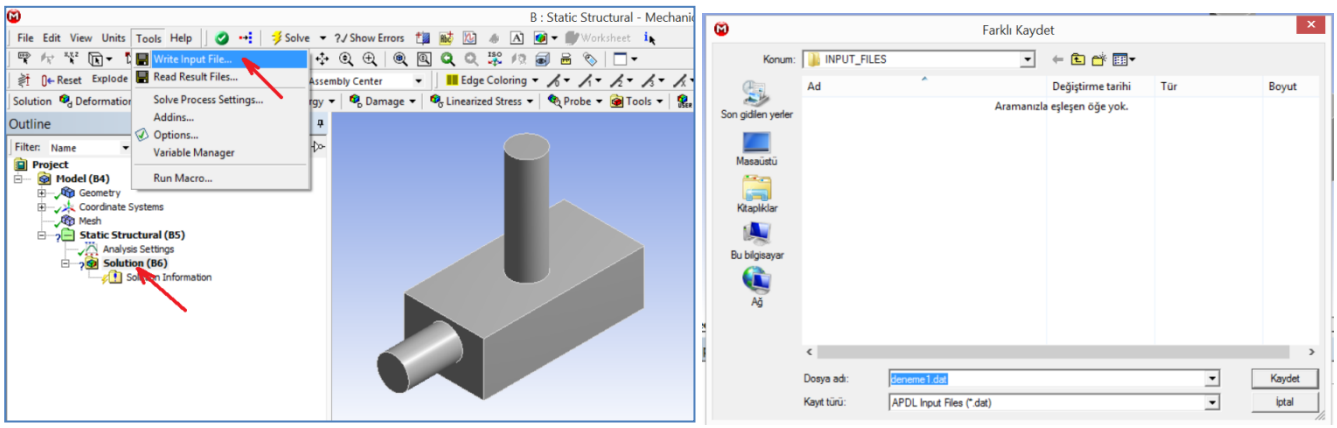
Kendi belirlediğimiz düzlem üzerinde sketch oluşturmak için ise şu kodlar kullanılır.

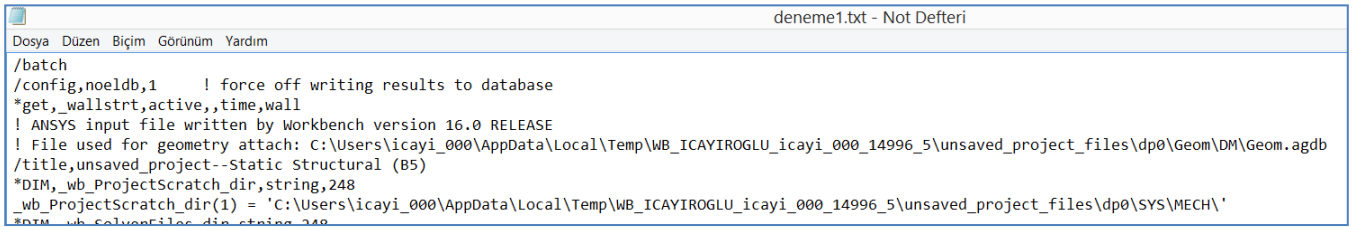
```
var YZPlane = agb.GetYZPlane(); //To get YZPlane object
agb.SetActivePlane (YZPlane); //To set Active plane
```

Örnek çalıştırılamadı. Uygun bir örnek çalıştırılınca ekle.

20. ÖRNEK: Ansys Mechanical'daki modelin APDL (eski ansys) scripting kodlarına dönüştürme.

Bunun için Ansys Structural Static içerisinde bir model oluşturalım. Model ağacından Solution başlığına tıklayıp Tools->Write Input File seçeneğini kullanarak dosyayı kaydedeceğimiz yeri belirleyelim. .dat uzantılı olarak kaydedelim. Not defteri açmaz ise uzantısını txt ye çevirip içeriğine bakabiliriz. Bu dosya Ansys-APDL tarafından okunabilir. Orada File->Read Input From seçeneğini kullanın.





...

```
etcon,set             ! allow ANSYS to choose best KEYOP's for 180x elements
/com,********** Nodes for the whole assembly **********
nblock,3
(1i9,3e20.9e3)
        1   -2.000000000E+000   -2.000000000E+000    0.000000000E+000
        2   -2.439339828E+000   -9.393398282E-001    0.000000000E+000
        3   -3.500000000E+000   -5.000000000E-001    0.000000000E+000
        4   -4.560660172E+000   -9.393398282E-001    0.000000000E+000
```

...

```
_posttime=(_walldone-_wallasol)*3600
_totaltim=(_walldone-_wallstrt)*3600
/wb,file,end              ! done with WB generated input
```

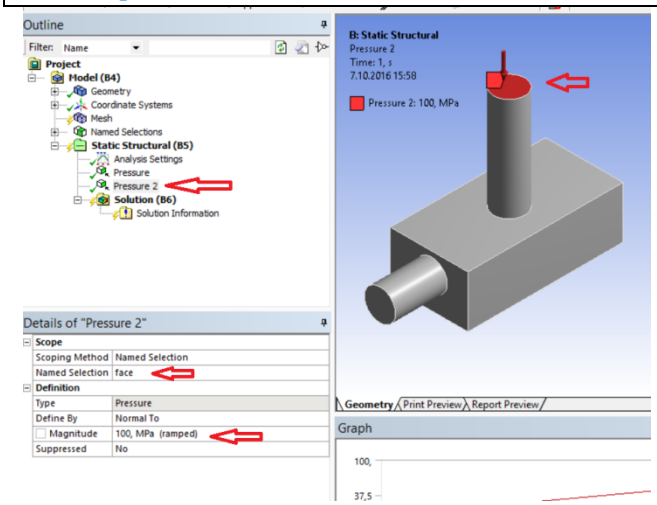
St 2965, Stn 37

21. ÖRNEK: Script kullanarak yüzeye basınç uygulama

Aşağıdaki kodların çalışabilmesi için öncesinde bir cisim oluşturun ve bunun bir yüzeyine "face" adini verin (Named Selection) oluşturun. Daha sonra Tools>Run Macro yolunu kullanarak bu kodları çalıştırın.

```
//Select the "Environment" node
var Env = DS.Tree.FirstActiveBranch.Environment;
DS.Script.SelectItems(""+Env.ID);

//Insert the Pressure Load
DS.Script.doInsertEnvironmentPressure();

//Change the Scoping Method to "Named Selection"
ListView.ActivateItem("Scoping Method");
ListView.ItemValue = "Named Selection";
```

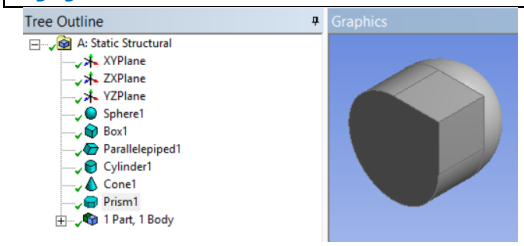
```
//Select the NS "face" : Dikkat burada "face" ismi verilmis yuzeyi bulabilmesi icin oncesinde olusturulmus olmasi gerekir.
ListView.ActivateItem("Named Selection");
ListView.ItemValue = "face";

//Provide the pressure magnitude
ListView.ActivateItem("Magnitude");
ListView.ItemValue = "100,0";

//Update the tree
DS.Script.fillTree();
```



22. ÖRNEK: Komutla primitive leri oluşturma

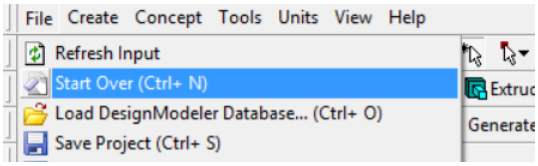
Aşağıdaki kodları deneme.js olarak kaydedin. DM içinde çalıştırdığımızda bu cisimler oluşacaktır. Tabii tüm cisimler 1 boyutlarında ve birleşmiş olarak gözükecektir.

```
ag.gui.CreatePrimitive(1); //Kure olusturur
ag.gui.CreatePrimitive(2); //Kup olusturur
ag.gui.CreatePrimitive(3); //Parallelepiped1
ag.gui.CreatePrimitive(4); //Cylinder1
ag.gui.CreatePrimitive(5); //Cone1
ag.gui.CreatePrimitive(6); //Prism1
```



23. ÖRNEK: Primitive komutlarını kullanarak Ölçülü küp çizme

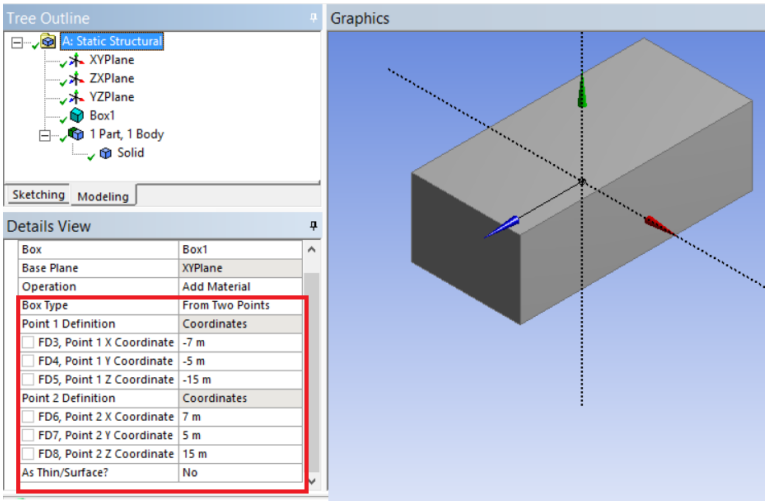
Primitive komutunlarını kullanarak bir küp çizdirelim fakat küpün ölçüleri olsun. Öncelikle önceki uygulamaları sıfırlamak için **Start Over (Ctrl+N)** komutunu kullanalım.




```
ag.gui.CreatePrimitive(2); //Kup olusturur
```

```
ag.listview.ActivateItem("Operation");
ag.listview.ItemValue="Add Material"; //yada  ag.listview.ItemValue="Add Frozen"; komutunu kullan

ag.listview.ActivateItem("Box Type");
ag.listview.ItemValue="From Two Points";

ag.listview.ActivateItem("Point 1 Definition");
ag.listview.ItemValue="Coordinates";

ag.listview.ActivateItem("FD3, Point 1 X Coordinate");
ag.listview.ItemValue= -7,5;

ag.listview.ActivateItem("FD4, Point 1 Y Coordinate");
ag.listview.ItemValue= -5;

ag.listview.ActivateItem("FD5, Point 1 Z Coordinate");
ag.listview.ItemValue= -15;


ag.listview.ActivateItem("Point 2 Definition");
ag.listview.ItemValue="Coordinates";

ag.listview.ActivateItem("FD6, Point 2 X Coordinate");
ag.listview.ItemValue= 7,5;

ag.listview.ActivateItem("FD7, Point 2 Y Coordinate");
ag.listview.ItemValue= 5;

ag.listview.ActivateItem("FD8, Point 2 Z Coordinate");
ag.listview.ItemValue= 15;

agb.Regen(); //Agac yapisina yeni bir eleman eklendiginde bu komut kullanilmali.
```

Dikkat edilirse komutlada kullanılan ifadeler, Properties penceresindeki menülerde yazan ifadelerdir.



Sketch Functions

- **ConstructionPoint(x, y)** : Returns a new Point in the sketch.
- **Line(x1, y1, x2, y2)** : Returns a new Line in the sketch.
- **Circle(xc, yc, radius)** : Returns a new Circle in the sketch.
- **ArcCtrEdge(xc, yc, xbegin, ybegin, xend, yend)** : Returns a new Arc in the sketch.
- **EllipticalArc(xc, yc, xmax, ymax, xmin, ymin, xbegin, ybegin, xend, yend)** : Returns a new Elliptical Arc in the sketch.
- **EllipticalArc(xc, yc, xmax, ymax, xmin, ymin, xbegin, ybegin, xend, yend)** :
- **SplineBegin()** : Returns an empty spline in the sketch. Then use the edge functions SplineXY and SplineFitPtEnd to finish defining it as a fit point spline. Functions SplineXYW, SplineKnot and SplineCtrlPtEnd are used to define a spline via control points, weights and knots.
- **Fillet(edge1, selx1, sely1, edge2, selx2, sely2, radius, trim)** : Returns the fillet (Arc or Circle) between the two edges near their selected ends.
- **Chamfer(edge1, selx1, sely1, edge2, selx2, sely2, offset, trim)** : Returns the chamfer (Line) between the two edges near their selected ends.
- **SplitEdge(edge, splitx, splity)** : Returns the new edge created by splitting the existing edge at the supplied location. The original edge is trimmed to that location.