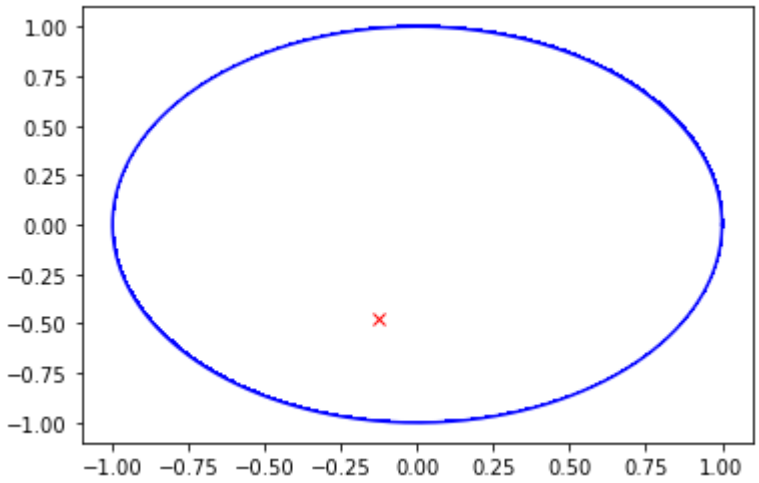
Bir çemberin içinde rastgele bir nokta seçmek istediğimizde başımıza en fazla ne gelebilir ki ?

Öncelikle basit kartezyen koordinat sisteminde rastgele bir nokta seçelim:

In [1]:

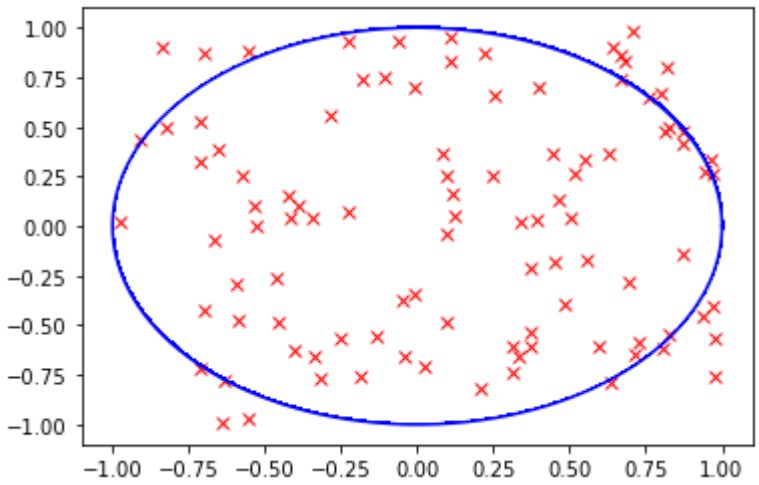
```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(353)
#random bir değer elde edeceğimiz zaman 0 ile 1 arasında bir değer elde ederiz. 2 ile çarpı
a=np.random.rand(2,1)*2-1
#Çemberin yarıçapını 1 olarak aldım.
t = np.linspace(0,2*np.pi,1000)
xx = 1*np.cos(t)
yy = 1*np.sin(t)
plt.plot(a[0],a[1],"rx",xx,yy,"b,-")
plt.show()
```



1 nokta aldık ve o nokta çemberin içinde çıktı. Peki ya 100 nokta alsaydık ne olurdu?

In [2]:

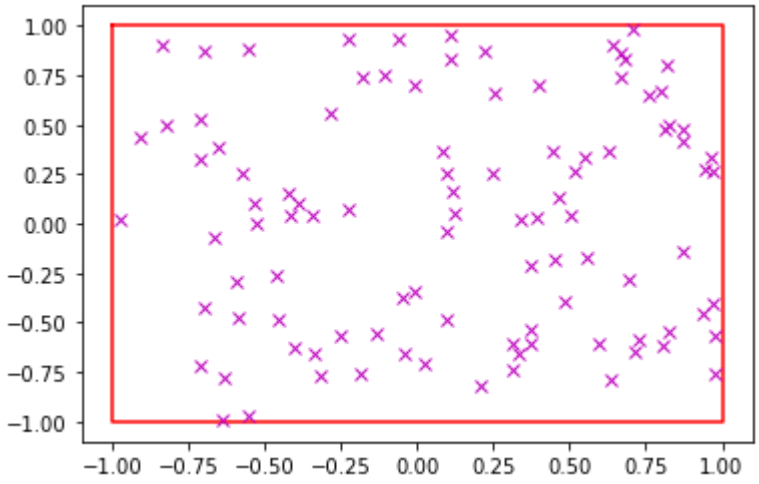
```
b=np.random.rand(100,2)*2-1
t = np.linspace(0,2*np.pi,1000)
xx = 1*np.cos(t)
yy = 1*np.sin(t)
plt.plot(b[:,0],b[:,1],"rx",xx,yy,"b,-")
plt.show()
```



Görüldüğü gibi bazı noktalar çemberin dışında, bunun sebebi kartezten koordinat sistemi kullanmamızdır.

In [3]:

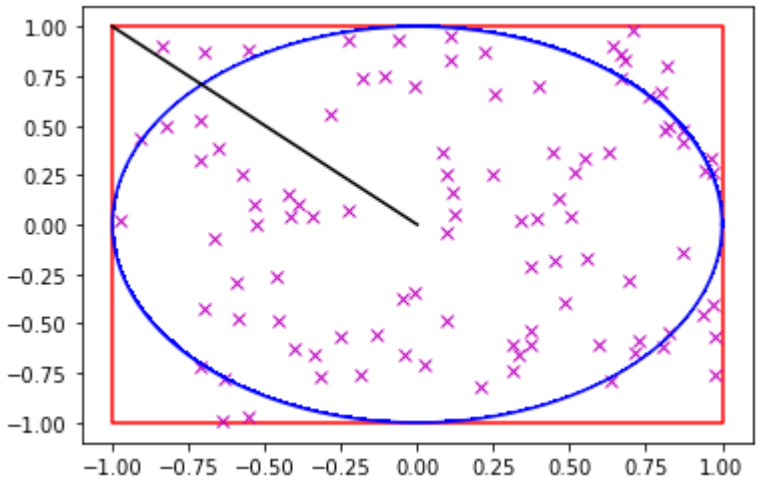
```
plt.plot([-1,1,1,-1,-1],[1,1,-1,-1,1],"-r",b[:,0],b[:,1],"mx")
plt.show()
```



Kartezyen koordinat sisteminde kare çizdiğimizde tüm noktaların o karenin içinde kaldığını görüyoruz, bunun nedeni kartezten koordinat sisteminde x ve y olarak iki bileşeni olan rastgele noktalar için en düşük -1 en fazla 1 değerini alabileceğini söyledik. Eğer bir nokta -1,1 konumundaysa orjinden -1,1 noktasına bir doğru çizdiğimizde karenin tam sınırında, çemberin ise dışında olduğunu göreceğiz. Bunun nedeni çemberin yarıçapına 1 dedik fakat orijinden -1,1 e doğru çizdiğimizde o doğrunun uzunluğu 1.414 olacak ve bu da çemberin yarıçapından daha büyük bir değer.

In [4]:

```
plt.plot([-1,1,1,-1,-1],[1,1,-1,-1,1],"-r",b[:,0],b[:,1],"mx",xx,yy,"b,-",[0,-1],[0,1],"k-"
plt.show()
```



Peki bu noktaların çember içerisinde olma ihtimali nedir? Bunun ihtimali bu diskin alanının karenin alanına oranı olacak.

$\frac{\pi r^2}{2r^2} = \frac{\pi}{4}$ bu da yaklaşık olarak %78.54'e tekabül ediyor.

Bu sorunu çözmenin bir yolu noktalar matrixinin içindeki noktaların orijinden olan konum vektörlerinin normunu hesaplamak ve normu 1'den büyük olanların matrixten atılmasını sağlamak. Fakat bu durumda istediğimiz kadar noktayı içerisine koyamayız, kodu değiştirmek zorunda kalırız.

Fakat kartezyan koordinatlar kullanmak yerinde kutupsal koordinatlar kullanırsak işimize gelebilir.

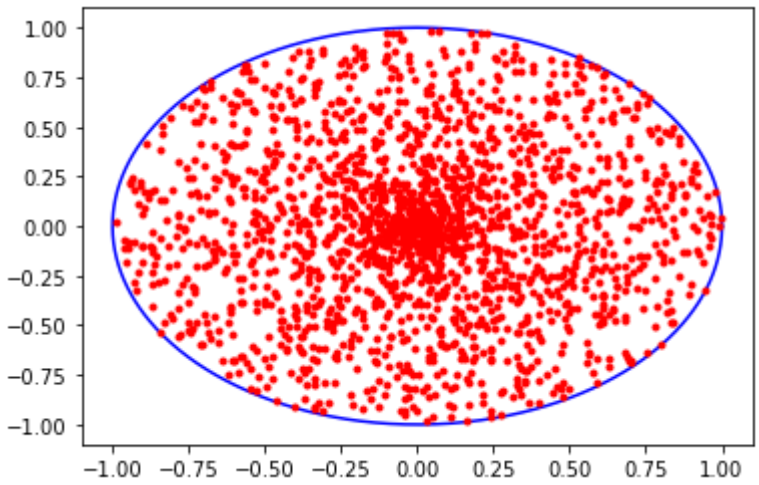
Kartezyen koordinat sisteminde (x,y) noktası aslında bir noktanın x ve y eksenine dik uzaklıklarıdır. İşaretleri ise koordinat sisteminde hangi bölgede olduğunu söyler.

Kutupsal koordinat sistemi ise aynı noktanın farklı özellikleriyle ifade edilmesidir. Kutupsal koordinat sistemi ise (r,ϕ) olarak ifade edilir. r orijine olan uzaklıktır. Yani $r = \sqrt{x^2 + y^2}$ 'dir. ϕ ise orijinden o noktaya çizdiğimiz doğrunun x ekseni ile yaptığı açıdır.

Kutupsal koordinat sistemini kartezyene dönüştürmek için $x = r * \cos(\phi)$, $y = r * \sin(\phi)$ kullanılır.

In [5]:

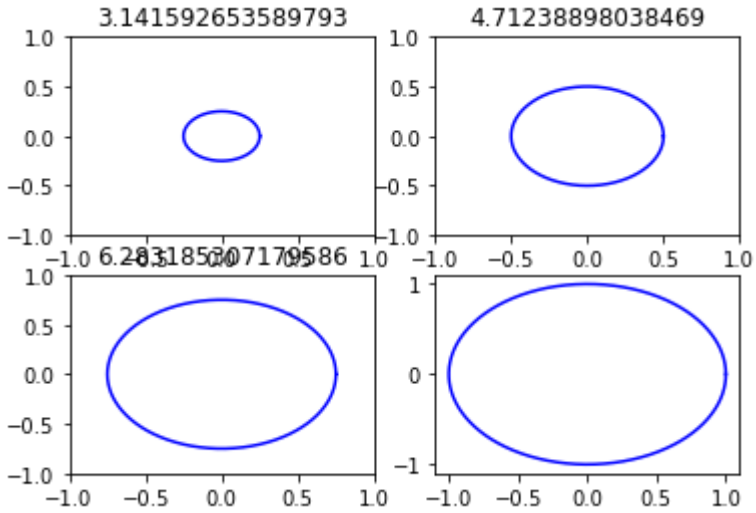
```
r=np.random.rand(2000)
phi=np.random.rand(2000)*np.pi*2
x=r*np.cos(phi)
y=r*np.sin(phi)
plt.plot(xx,yy,"b-",x,y,"r.")
plt.show()
```



Yukarıda da gördüğümüz gibi noktalar homojen olarak yayılmıyor, bu sefer de merkez nokta etrafında çok fazla nokta varken r değeri büyüdükçe yoğunluk azalıyor. Bunun sebebi ϕ'de yatıyor olamaz çünkü çemberin etrafında homojen bir dağılım mevcut o halde problem r'de yatıyor olmalı.

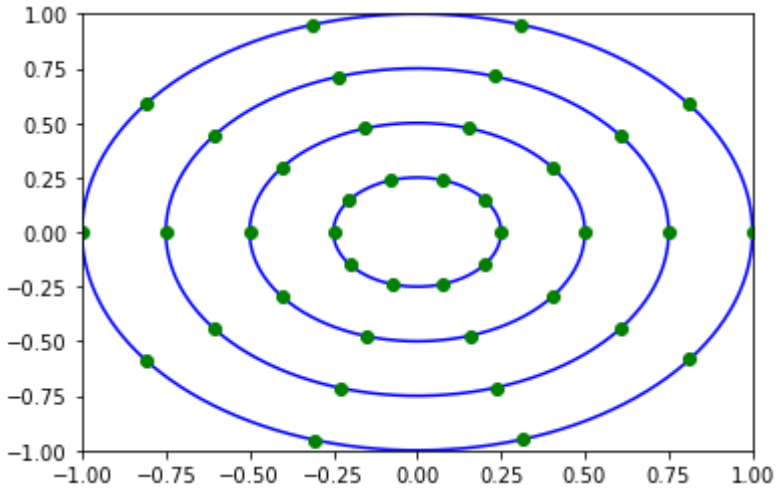
In [6]:

```
A=np.zeros(1000)
B=np.zeros(1000)
for i in range(1,5):
    t = np.linspace(0,2*np.pi,1000)
    xx11 = 0.25*i*np.cos(t)
    yy11 = 0.25*i*np.sin(t)
    A=xx11
    B=yy11
    plt.xlim([-1,1])
    plt.ylim([-1,1])
    plt.title(2*np.pi*0.25*i)
    plt.subplot(2,2,i)
    plt.plot(A,B,"b-")
```



In [7]:

```
A=np.zeros(1000)
B=np.zeros(1000)
for i in range(1,5):
    t = np.linspace(0,2*np.pi,1000)
    xx11 = 0.25*i*np.cos(t)
    yy11 = 0.25*i*np.sin(t)
    A=xx11
    B=yy11
    plt.xlim([-1,1])
    plt.ylim([-1,1])
    plt.plot(A,B,"b-",A[0:1000:100],B[0:1000:100],"og")
```



Yukarıdaki şekillerden de görüldüğü üzere r büyüdüğü zaman çevresi de büyüyor. Fakat r=0.1m , r=1m olduğu zaman da eşit ihtimalli olarak rastgele nokta alıyorlar yani r=0.1m için çevresi $2\pi r = 0.2\pi$. r=1m için ise çevresi $2\pi r = 2\pi$. Aralarında 10 kat kadar fark olmasına rağmen o çevrelerde nokta çıkma ihtimali aynı bu da küçük r değerlerinde yani merkez çevresinde yoğunlaşmış nokta görmemize sebep oluyor. Bu da homojenliği bozuyor.

Problemimize farklı bir matematiksel yaklaşım yapalım.

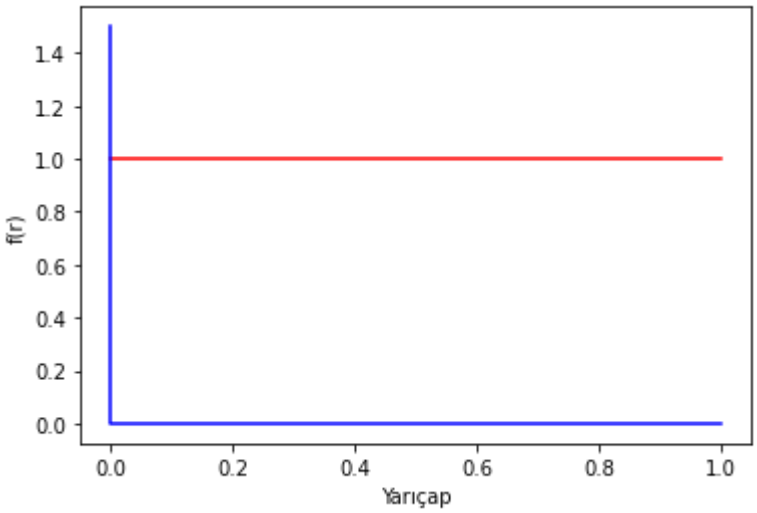
$Probability\ Density\ Function$ (PDF) - $Olasılık\ Yoğunluğu\ Fonksiyonu$

Sürekli olan rastgele değişkenimizin yoğunluğu. Bizim değişkenimiz yarıçap r 0 ile 1 değerleri arasında rastgele değerler alıyor ve her değerin gelme ihtimali eşit. Bu durumda bizim olasılık yoğunluğu fonksiyonumuz grafikte şöyle gözükür.

Grafiğin fonksiyonu $f_x(r) = 1$

In [8]:

```
x2=np.linspace(0,1,100)
y2=np.linspace(1,1,100)
plt.plot(x2,y2,"r-",[1,0,0],[0,0,1.5],"b-")
plt.xlabel("Yarıçap")
plt.ylabel("f(r)")
plt.show()
```



Bu grafik bize ayrıca gösteriyor ki her yarıçapın gelme oranı aynı 0.54'ün gelme ihtimali neyse 0.11'in gelme ihtimali de o.

Yukarıdaki doğrunun altında kalan alanın 1 olması gerekiyor çünkü olasılık en yüksek 1 olabiliyor ve bizim tüm r değerlerimiz 0 ile 1 arasında grafikte de tüm r ihtimallerini içine kattığımız için alan 1 olmalı.

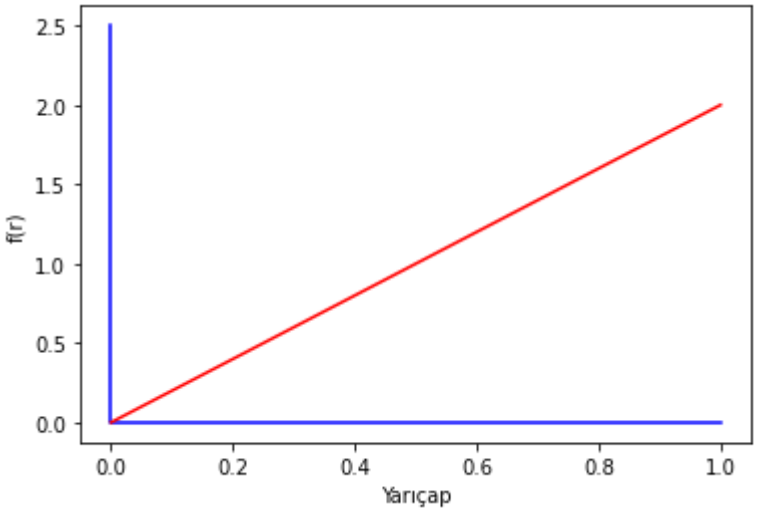
Buna bir başka örnek de Bir parçacığın uzayda bulunma ihtimali eğer biz tüm uzayda arama yaparsak yani tüm uzay için $-\infty$'den ∞'ye kadar integral alırsak. Uzayda bir parçaçık bulmamız kesindir. Olasılığı 1'dir.

Problemimize geri dönersek homojenliği yakalamak için ne yapmamız gerekiyor ? $S=2\pi r$ olduğundan r doğrusal olarak artarken noktaların sayısı da artması gerekiyor.

Grafiğimizin şu şekilde gözükmesi gerekiyor.

In [9]:

```
x3=np.linspace(0,1,100)
y3=2*x3
plt.plot([1,0,0],[0,0,2.5],"b-",x3,y3,"r-")
plt.xlabel("Yarıçap")
plt.ylabel("f(r)")
plt.show()
```



Asıl soru bu doğrunun eğimini ben nasıl buldum? Demiştim ki eğer tüm ihtimalleri kapsıyorsa doğrunun altında kalan alan 1 olacak.

Doğrunun denkleminin y=mx olarak verildiğini de biliyoruz. O zaman $f_x(r) = mr$ olması gerekiyor ve doğrunun altında kalan alan 1'e eşit olmalı.

$\int_0^1 mrdr$=1 olmalı buradan $m = 2$ olması gerektiğini buluruz.

Bu grafik bize homojen bir dağılım sağlamak için r değerinin arttıkça olasılık fonksiyonunun artması gerektiğini yani daha büyük r değerleri için daha fazla nokta olması gerektiğini söylüyor.

Aslında bizim istediğimiz homojen bir dağılım değil bu grafiğin en önemli olarak bize söylediği şey o bize gereken doğrusal bir dağılım.

$Cumulative\ Distribuiton\ Funtion\ (CDF)$ ise sürekli olasılık dağılımlarından türetilen özel bir fonksiyondur. Türkçeye cumulative birikim olarak çevirilebilir bu bize bu fonksiyonun nasıl oluştuğuna dair bir ipucu verebilir. CDF'i oluşturması için var olan olasılıkların toplanarak 1'e ulaşması ve $F_X(x)$'i oluşturur.

Bunun için kullanılan formul $F_X(x)$=$\int_{-\infty}^{x} f_x(x)dx$ 'dir. Yalnız şunu unutmamak gerekir ki bu formul yalnızca sürekli olasılık dağılımı olan yerlerde geçerlidir.

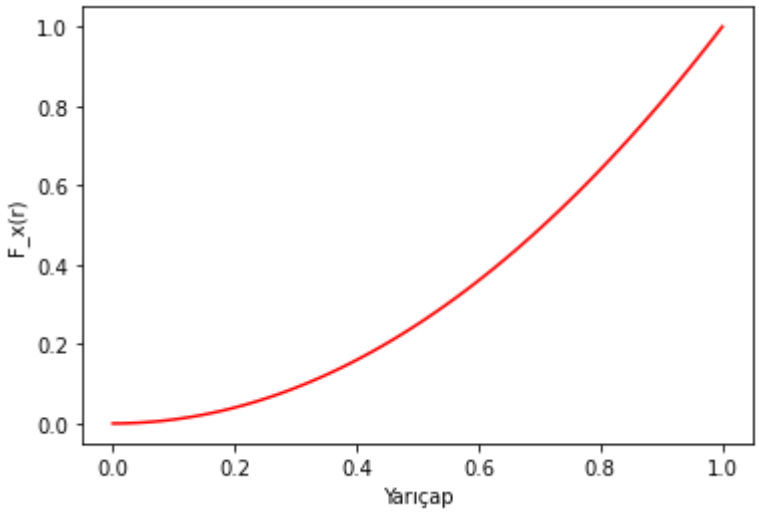
(Formulde integralin üst sınırında yazan x için 0.5 yazarsak 0.5'e kadar olan olasılıkları toplar CDF'ler giderek 1'e yakınlaşan fonksiyonlardır. Bu örnekte işimize yarayan tüm olasılıkların toplanması.)

Bu formule göre bizim CDF'imiz $F_X(r)$=$\int_{-\infty}^{0} 0dr$ + $\int_0^r 2rdr$=r^2

CDF'in bize söylediği en önemli şey $P(X < r) = r$'dır. P burada olasılıktır yani r=0.5m için bakarsak $P(X < 0.5) = 0.5$'dir X burada rastgele değerlerdir.

In [10]:

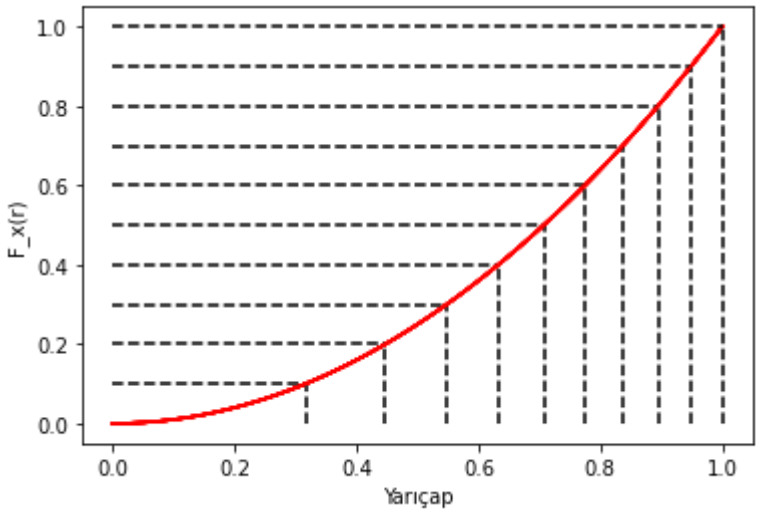
```
x4=np.linspace(0,1,100)
y4=x4**2
plt.plot(x4,y4,"r-")
plt.xlabel("Yarıçap")
plt.ylabel("F_x(r)")
plt.show()
```



CDF'in bize söylediği en önemli şey $F_X(r)$=$P(X < r) = r$'dır. P burada olasılıktır yani r=0.5m için bakarsak $P(X < 0.5) = 0.5$'dir X burada rastgele değerlerdir.

In [11]:

```
x4=np.linspace(0,1,100)
y4=x4**2
AA=np.zeros(10)
BB=np.zeros(10)
for i in range(1,11):
    plt.xlabel("Yarıçap")
    plt.ylabel("F_x(r)")
    AA[i-1]=(0.1*i)
    BB[i-1]=(0.1*i)**0.5
    plt.plot(x4,y4,"r-",[0,BB[i-1]],[AA[i-1],AA[i-1]],"k--",[BB[i-1],BB[i-1]],[0,AA[i-1]],"
```

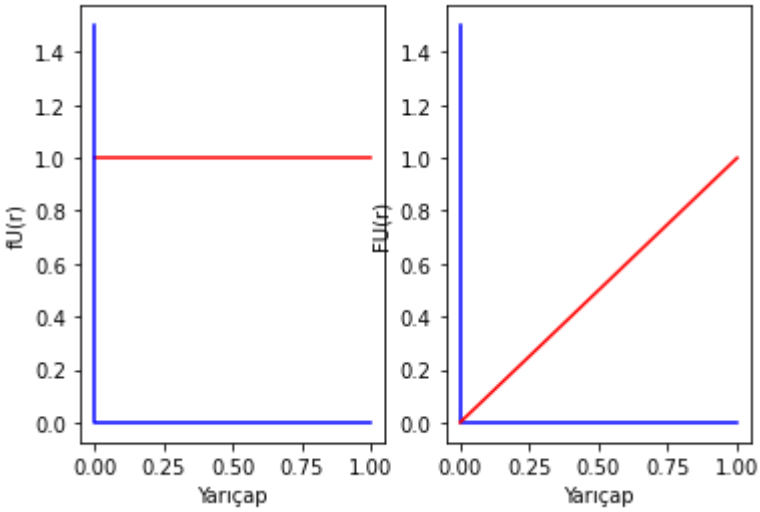


Rastgele değerlerimiz olan X değerleri homojen olarak dağılmadığı için X'lerin olasılıkları y ekseninde gösterilmiştir çünkü olasılıkları eşit ve homojen olarak dağılmıştır ki tam olarak amacımız da buydu. Buradaki y değerlerimiz giriş değerleri'nin olasılıkları gibi düşündüğümüzde yarıçap değerinin arttıkça daha yoğun bir şekilde yarıçap değer noktaları elde ettiğimizi görüyoruz bu aslında bizim tam olarak istediğimiz şey.

Sonraki aşamaya geçmeden önce toplu olarak bir kere homojen dağılım PDF ve CDF'leri görelim. U 0 ile 1 arasında homojen rastgele değişken olsun.

In [12]:

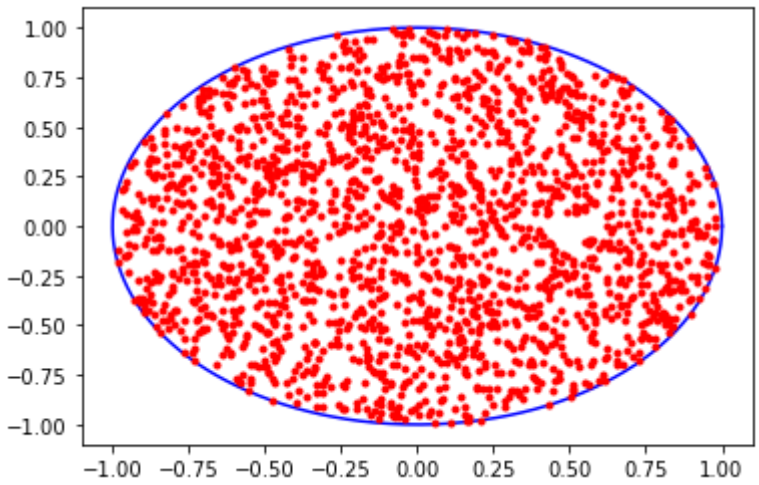
```
x2=np.linspace(0,1,100)
y2=np.linspace(1,1,100)
plt.subplot(1,2,1)
plt.plot(x2,y2,"r-",[1,0,0],[0,0,1.5],"b-")
plt.xlabel("Yarıçap")
plt.ylabel("fU(r)")

x3=np.linspace(0,1,100)
y3=x3
plt.subplot(1,2,2)
plt.plot([1,0,0],[0,0,1.5],"b-",x3,y3,"r-")
plt.xlabel("Yarıçap")
plt.ylabel("FU(r)")
plt.show()
```



$F_U(r) = P(X < r) = r$ olduğunu söylemiştik. Bir taktik ile homojen dağılımı alıp değiştirebilen bir şeye ihtiyacımız var. Böyle bir şey istememizin sebebi bizim rastgele yarıçap üreten kodumuz da bir homojen dağılımdır. Burada yapacağımız taktik $Inverse\ Transform\ Sampling$'dir. $F_X(r) = P(U < r)$= $P(U < F_X(r))$ ve eğer içerisinin tersini alırsak. $P(F_X(U)^{-1} < r)$ bunun sonunca $X = F_X(U)^{-1}$ elde ederiz. Yani bize gerekli olan X (rastgele homojen olmayan değerlerimiz) r^2'nin tersi olan $\sqrt{r}$ 'dir. $F_X(r)^{-1} = \sqrt{r}$

In [13]:

```
r=np.sqrt(np.random.rand(2000)) #tek yaptığım değişiklik r değerlerini almak yerine r^0.5 d
phi=np.random.rand(2000)*np.pi*2
x=r*np.cos(phi)
y=r*np.sin(phi)
plt.plot(xx,yy,"b-",x,y,"r.")
plt.show()
```

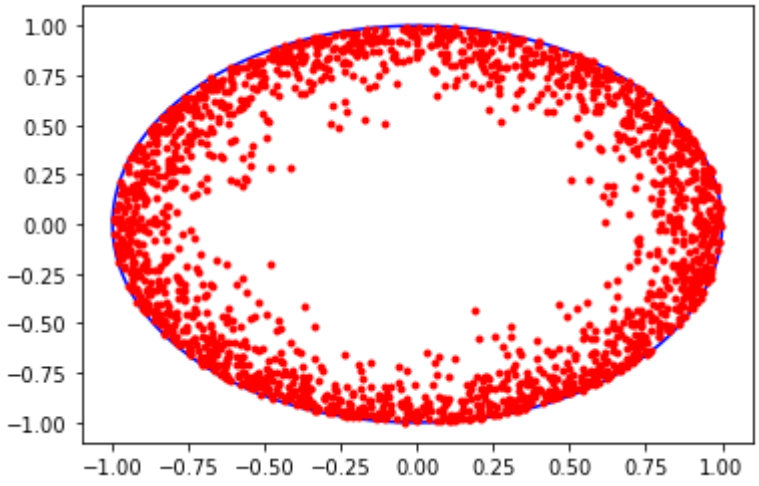


Yukarıda gördüğümüz üzere çember içerisinde homojen dağılımlı rastgele noktalar elde edebildik.

Eğer noktaların yoğunluğunu yarıçap arttıkça daha da fazla olmasını istiyorsak tek yapmamız karekökünü almak yerine küp kökünü veya daha yüksek kuvvetlerde kökünü almaktır. Hatta merkezde hiç nokta olmamasını bile sağlayabiliriz.

In [14]:

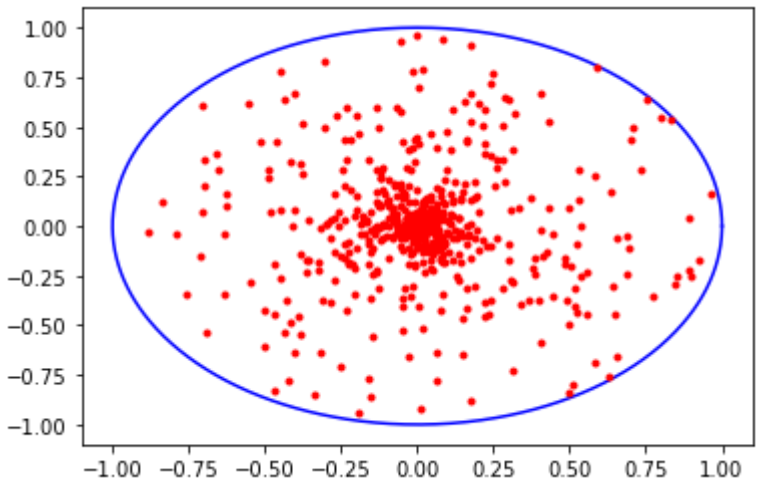
```
r=(np.random.rand(2000))**0.1
phi=np.random.rand(2000)*np.pi*2
x=r*np.cos(phi)
y=r*np.sin(phi)
plt.plot(xx,yy,"b-",x,y,"r.")
plt.show()
```



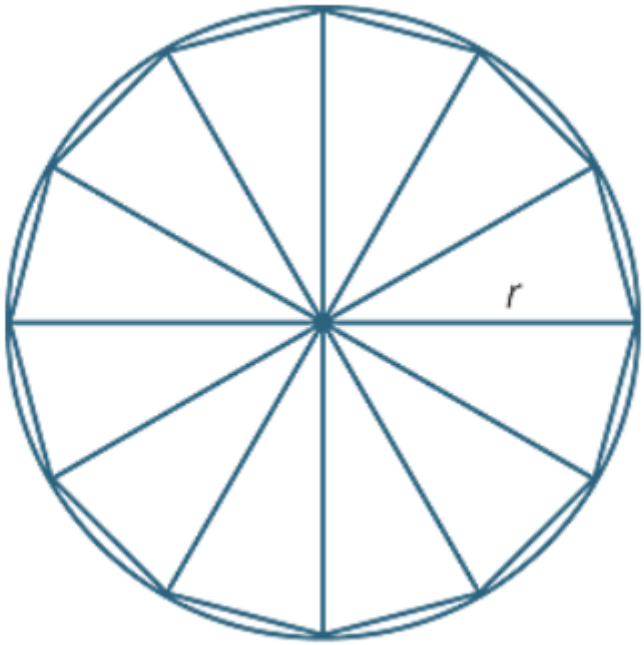
Aynı şekilde noktaların merkezde daha fazla toplanmasını istersek bu sefer daha yüksek sayılarda üssünü almamız gerekecek

In [15]:

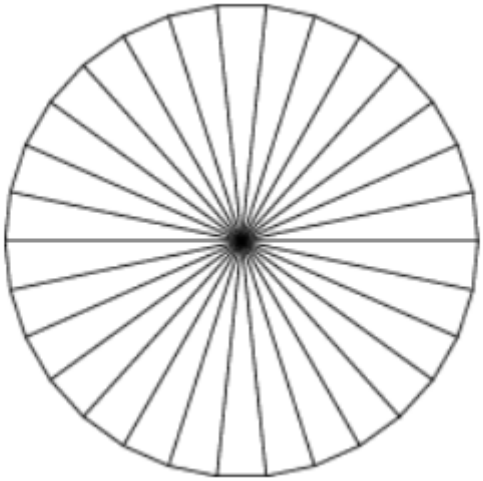
```
r=(np.random.rand(2000))**10
phi=np.random.rand(2000)*np.pi*2
x=r*np.cos(phi)
y=r*np.sin(phi)
plt.plot(xx,yy,"b-",x,y,"r.")
plt.show()
```



Şuanda aslında amacımıza çoktan ulaştık rastgele yarıçap değişkenimizin kökünü aldığımızda istediğimiz homojen dağılmış noktaları elde ettik peki bunu yapmanın başka bir yolu var mı ?



Bir diski üçgenlerden oluşuyor gibi düşünebiliriz eğer bu üçgenlerin sayısını arttırırsak şekil o kadar diske benzemeye başlayacak

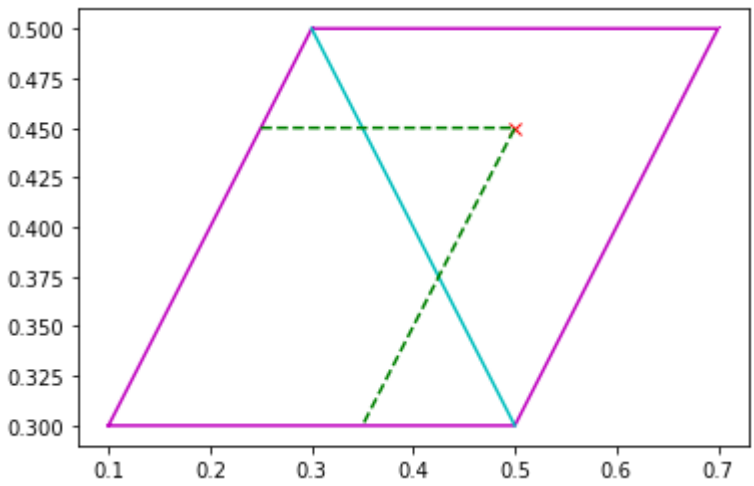


Üçgen sayısı arttıkça şekil bu şekilde gözükecektir.

Peki bu neden önemli ? Yukarıdaki şekilde orijin noktası siyah büyük bir nokta ile gösterilmiş bunun nedeni eğer biz üçgenleri orijin noktasında döndürerek elde edersek diski her üçgene karşı gelen bir karşı üçgen olacaktır. Eğer ki ben bu üçgenler içerisinde rastgele bir nokta seçersem çember içerisinde rastgele bir nokta seçmiş olurum. Ve bunu kartezyen koordinat sisteminde yaparsam bazı noktalar yine çemberin dışında olacaktır. Bunu engellemek için üçgenlerin tabanlarından çember dışına doğru üçgenleri yansıtırsak çember dışında kalan nokta o üçgenin içerisinde kalır ve o noktayı da üçgenimizin tabanını esas alarak yansıtırsak o nokta çemberin içinde olur.

In [16]:

```
x = [0.1,0.5,0.7,0.3,0.1]
y = [0.3,0.3,0.5,0.5,0.3]
x1=[0.5,0.3]
y1=[0.3,0.5]
x2=[0.5]
y2=[0.45]
x3=[0.25,0.5,0.5,0.35]
y3=[0.45,0.45,0.45,0.3]
plt.plot(x,y,"m,-",x1,y1,"c-",x2,y2,"rx",x3,y3,"g--")
plt.show()
```



Yukarıdaki örnekte eşkenar dörtgenin skalaları değiştirilerek çizilmiştir çember içerisindeki üçgeni temsil etmemektedir. Yukarıdaki örnekte 2 tane üçgen görmekteyiz aslında bu üçgenlerden solda olanı çemberin içinde olan üçgen sağdaki üçgen ise çemberin dışında olan hayali üçgen gibi düşünürsek rastgele bir nokta

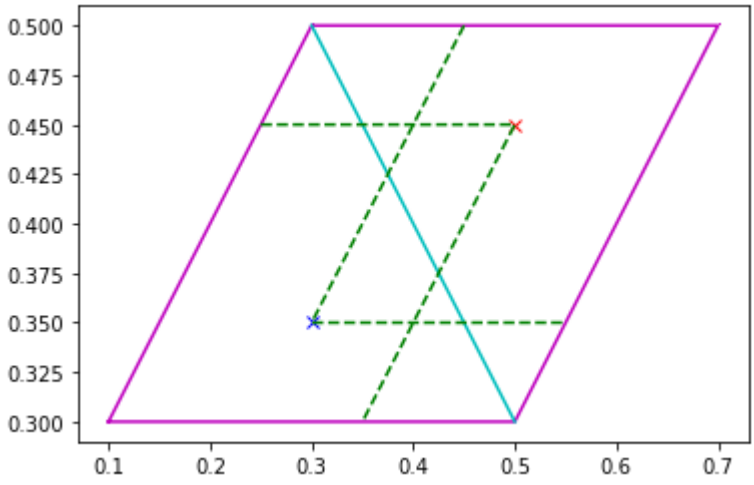
aldığımızda içteki üçgenin dışında fakat kartezyen koordinat sistemine dahil bir yerde nokta çıktığını simüle ettiğimizi düşünelim. Eyvah nokta dışarıda çıktı şimdi ne yapacağız ? tabiki turkuaz renkli olan doğruye göre yansıtıcaz. Doğrunun eğimine bakarsak $-0,2 \div 0,2 = -1$'dir. $y = -x$ doğrusuna göre yansıtmamız gerekiyor.

In [17]:

```
döndürmat=np.array([[-1,0],[0,-1]])#pi kadar döndürmek için döndürme matrixi
nokta=np.array([x2,y2])
nok=np.dot(döndürmat,nokta)
nok[0]=nok[0]+0.8 #burada 0.8 eklememin sebebi x=0 olduğunda y=0.8 olacak doğru üzerinde ya
```

In [18]:

```
x = [0.1,0.5,0.7,0.3,0.1]
y = [0.3,0.3,0.5,0.5,0.3]
x1=[0.5,0.3]
y1=[0.3,0.5]
x2=[0.5]
y2=[0.45]
x3=[0.25,0.5,0.5,0.35]
y3=[0.45,0.45,0.45,0.3]
x4=[0.3]
y4=[0.35]
x5=[0.45,0.3,0.55]
y5=[0.5,0.35,0.35]
plt.plot(x,y,"m,-",x1,y1,"c-",x2,y2,"rx",x3,y3,"g--",x4,y4,"bx",x5,y5,"g--")
plt.show()
```



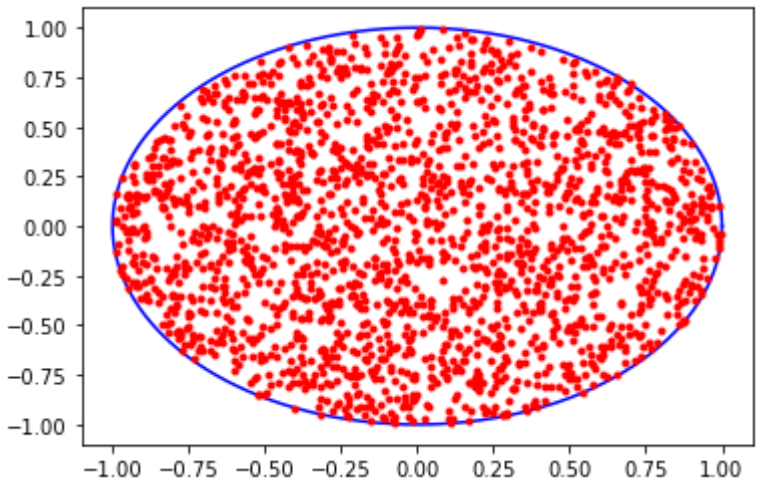
Mavi x ile gösterilen yer yansıyan noktamız. Ve artık o nokta içeride.

Fakat biz bu üçgenlerden sonsuz tane olduğunu düşünürsek bu üçgenin kalınlığı hakkında endişelenmemize gerek kalmaz o zaman bir doğru gibi bile düşünebiliriz. Yani yukarıdaki grafikte bu eşkenardörtgeni masa üzerinde düşünürsek üstten bakıyormuş gibiyiz şimdi de bu masaya masanın hizasından bakalım.

Turkuaz doğrunun uzunluğu 0'a indirmiş gibi düşünebiliriz. Ve mavi x'in koordinatları eşkenar dörtgene göre toplamı $x1$ ve $x2$ olsun kırmızı x'in konumu $x1+x2$ olacak.

In [19]:

```
theta=np.random.rand(2000)*2*np.pi
r=np.random.rand(2000)+np.random.rand(2000)
for i in range(2000):
    if r[i]>= 1:
        r[i]=2-r[i] #2'den çıkarma sebebim r max yarıçarpı 1'dir.1+1=2'dir.
xb=r*np.cos(theta)
yb=r*np.sin(theta)
plt.plot(xx,yy,"b-",xb,yb,"r.")
plt.show()
```



TAA DAA

2 Farklı yol ile çember içerisinde rastgele noktalar seçmiş olduk.

References:

https://www.youtube.com/watch?v=4y_nmpv-9II&list=PLnQX-jgAF5pTkwtUuVpqS5tuWmJ-6ZM-Z&index=7&ab_channel=nubDotDev (https://www.youtube.com/watch?v=4y_nmpv-9II&list=PLnQX-jgAF5pTkwtUuVpqS5tuWmJ-6ZM-Z&index=7&ab_channel=nubDotDev)

https://www.youtube.com/watch?v=glSmBTNmjzY&ab_channel=BUdersBo%C4%9Fazi%C3%A7iliden%C3%96zelDers (https://www.youtube.com/watch?v=glSmBTNmjzY&ab_channel=BUdersBo%C4%9Fazi%C3%A7iliden%C3%96zelDers)

https://www.w3schools.com/python/matplotlib_subplots.asp (https://www.w3schools.com/python/matplotlib_subplots.asp)

https://www.youtube.com/watch?v=W_EfvJkL9tc&list=PLcNWqzWzYG2ufnZbJYffUkgYkhh3kS44L&index=53&ab_channel=BUdersBo%C4%9Faz (https://www.youtube.com/watch?v=W_EfvJkL9tc&list=PLcNWqzWzYG2ufnZbJYffUkgYkhh3kS44L&index=53&ab_channel=BUdersBo%C4%9Faz

https://www.youtube.com/watch?v=tZJpvA1sp0Y&ab_channel=BUdersBo%C4%9Fazi%C3%A7iliden%C3%96zelDers (https://www.youtube.com/watch?v=tZJpvA1sp0Y&ab_channel=BUdersBo%C4%9Fazi%C3%A7iliden%C3%96zelDers)

https://en.wikipedia.org/wiki/Cumulative_distribution_function (https://en.wikipedia.org/wiki/Cumulative_distribution_function)

http://www.amsi.org.au/teacher_modules/the_circle.html (http://www.amsi.org.au/teacher_modules/the_circle.html)

https://math.stackexchange.com/questions/809451/how-is-a-circle-just-a-bunch-of-triangles/809457 (https://math.stackexchange.com/questions/809451/how-is-a-circle-just-a-bunch-of-triangles/809457)

Cafer Kocamaz

21824665

In [ ]: