😎 Matplotlib-Venn

Un diagrama de Venn (también llamado diagrama primario, diagrama de conjuntos o diagrama lógico) es un diagrama que muestra todas las posibles relaciones lógicas entre una colección finita de conjuntos diferentes.

Cada conjunto está representado por un círculo. El tamaño del círculo a veces representa la importancia del grupo, pero no siempre. Los grupos suelen superponerse: el tamaño de la superposición representa la intersección entre ambos grupos.

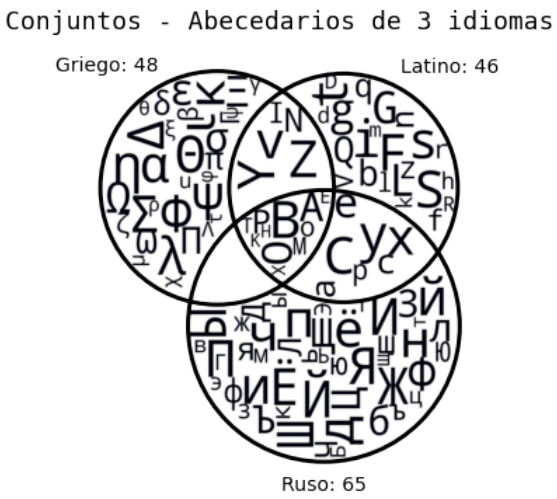
Aquí hay un ejemplo que muestra el número de palabras compartidas en las letras de 3 cantantes franceses: Nekfeu , Booba y Georges Brassens (https://www.data-to-viz.com/story/venn.png)

Un diagrama de Venn hace un buen trabajo para estudiar la intersección entre 2 o 3 conjuntos pero se vuelve muy difícil leer con más grupos .

Aquí hay un ejemplo de un diagrama de 6 vías publicado en Nature que muestra la relación entre el genoma del plátano y el genoma de otras cinco especies: https://www.nature.com/articles/nature11241/figures/4 (https://www.nature.com/articles/nature11241/figures/4)

Para aprender más sobre Diagramas de Venn ver: https://en.wikipedia.org/wiki/Venn_diagram#Edwards.27_Venn_diagrams (https://en.wikipedia.org/wiki/Venn_diagram#Edwards.27_Venn_diagrams)

Diagrama de Venn - Wikipedia, la enciclopedia libre: https://es.wikipedia.org/wiki/Diagrama_de_Venn (https://es.wikipedia.org/wiki/Diagrama_de_Venn)



Propósitos y beneficios

- **Organizar información visualmente para ver la relación entre los conjuntos de elementos, como semejanzas y diferencias.**
- **Comparar dos o más opciones y ver claramente lo que tienen en común y lo que puede distinguirlos.**
- **Resolver problemas matemáticos complejos.**
- **Comparar conjuntos de datos, encontrar correlaciones y predecir probabilidades de determinados acontecimientos.**
- **Razonar la lógica detrás de declaraciones o ecuaciones.**

Usos en diferentes campos:

- **Matemática: La teoría de conjuntos es una rama completa de la matemática.**
- **Estadística y probabilidad: se utilizan para predecir la probabilidad de determinados acontecimientos. Esto se relaciona con el campo del análisis predictivo.**
- **Lógica: se utilizan para determinar la validez de conclusiones y argumentos específicos.**
- **Lingüística: se utilizan para estudiar las diferencias y similitudes entre idiomas.**
- **Negocios: se utilizan para comparar y contrastar productos, servicios, procesos , etc. Son una herramienta de comunicación efectiva para ilustrar esa comparación.**
- **Psicología**
- **Ingeniería**
- **Planes de marketing**
- **Estudios de mercado**
- **Estudios demográficos**

La biblioteca matplotlib-venn ha sido creada por Konstantin Tretyakov (https://github.com/konstantint/matplotlib-venn) Lo primero que tenemos que hacer es instalar la librería:

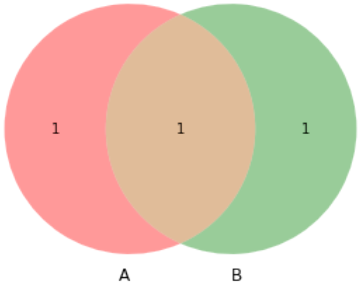
- **pip install matplotlib-venn** o **conda install matplotlib-venn** (si estás trabajando en jupyter notebook)

este comando también instalará las dependencias requeridas.

In [1]:
```
from matplotlib import pyplot as plt
# importamos  de matplotlib el módulo pyplot. La misma instrucción podría expresarse así:
# import matplotlib.pyplot as plt

from matplotlib_venn import venn2
# importamos de matplotlib_venn el módulo venn2 que tiene implementado clases, funciones, métodos, etc,
# para generar diagramas de Venn.
```

In [2]:
```
venn2((1, 1, 0))
plt.show()
```

In [3]:
```
venn2((1, 1, 1)) # generamos un diagrama  base  de  2  conjuntos.
plt.show()
```

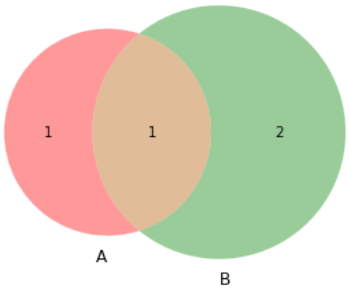


El argumento (1,1,1) indica el tamaño relativo de los tres subconjuntos en este orden:
Ab (izquierda), aB (derecha), AB (intersección).

- **Ab = contenido en el grupo A, pero no B**
- **aB = Contenido en el grupo B, pero no A**
- **AB = contenido en los grupos A y B**

Así, la tupla (1, 2, 1) dibujaría el conjunto B del doble de tamaño respecto de A:

In [4]:
```
venn2((1, 2, 1))
plt.show()
```



Y la tupla (2, 1, 1) qué dibujaría?. Construir.

Para identificar a cada uno de los subconjuntos (3 en diagramas de 2 conjuntos) el módulo utiliza una nomenclatura que consiste en colocar un 1 para indicar que la sección está incluida en el conjunto y un 0 para indicar que está excluida.
De esta manera, siguiendo el orden «ABC»:

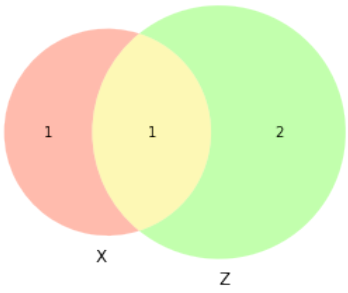
- **el subconjunto 10 (Ab) es el de la izquierda (el que pertenece a A pero no a B)**
- **el 01 (aB) , el de la derecha (el que pertenece a B pero no a A)**
- **y el 11 (AB), el del medio (la intersección)**

Teniendo esto en cuenta, como primer argumento , también podemos pasarle como parámetro un **diccionario** indicando el tamaño de cada uno de los subconjuntos:

El parámetro **set_labels** permite cambiar los nombres de los conjuntos mostrados en el diagrama. Si no queremos nombres especificamos en set_labels = **None**

set_colors determina el color de los conjuntos. Nótese que por defecto tiene un **alpha** (transparencia) de 0.4:

In [5]:
```
venn2({"10": 1, "01": 2, "11": 1}, set_labels=("X", "Z"), set_colors=("#FF5733", "#68FF33"))
plt.show()
```



La función venn2() retorna una instancia de una clase llamada VennDiagram.
Los métodos de ésta clase que nos interesan son:

- **get_label_by_id()**
- **get_patch_by_id()**

Ambas toman un subconjunto y retornan instancias de:

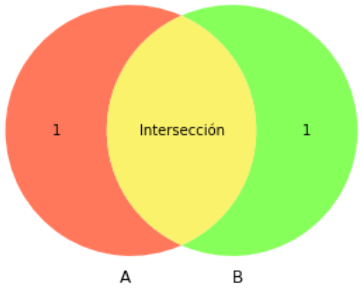
- **matplotlib.text.Text**
- **matplotlib.patches.PathPatch**

A partir de ellas podemos modificar individualmente el aspecto de cada subconjunto.

Más información aquí (https://monstott.github.io/visualizing_set_diagrams_with_python)

Con **get_label_by_id** el siguiente código agrega la palabra “Intersección” al subconjunto 11 (AB) usando la función **set_text** y establece el color de fuente con la función **set_color**:

In [6]:
```
dv = venn2((1, 1, 1),set_colors=("#FF5733", "#68FF33"), alpha=0.8)
dv.get_label_by_id("11").set_text("Intersección")
dv.get_label_by_id("11").set_color("#000000") #color de la fuente
plt.show()

# Obsérvese que,  previamente, asignamos a la variable “dv”, el diagrama y es para escribir un código más
# claro y ordenado.
```



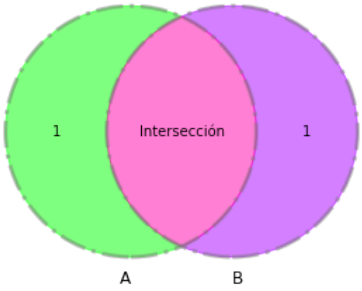
Por ejemplo, con **get_patch_by_id** t la función **set_color** cambiamos el color del conjunto B:

Estableciendo el identificador y método podemos personalizar los colores de cada parte de los conjuntos.

Para agregar líneas con color al diagrama, se debe importar el módulo **venn2_circles**:

In [7]:

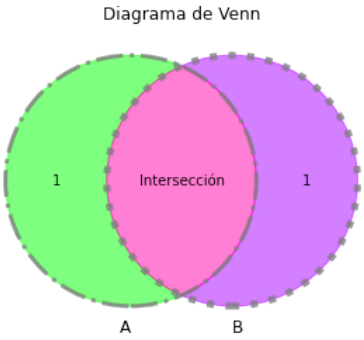
```
from matplotlib_venn import venn2_circles

dv = venn2((1, 1, 1),set_colors=("#FF5733", "#68FF33"), alpha=0.5)
dv.get_label_by_id("11").set_text("Intersección")
dv.get_label_by_id("11").set_color("#000000") #color de la fuente
dv.get_patch_by_id("01").set_color("#AA00FF") # Establece el color del conjunto B.
dv.get_patch_by_id("10").set_color("#00FF00") # Establece el color del conjunto A.
dv.get_patch_by_id("11").set_color("#FF00AA") # Establece el color de la intersección.

venn2_circles(subsets=(1, 1, 1), color="gray", alpha=0.5, linestyle="-.", linewidth=3)
plt.show()
```



Si configuramos el círculo del diagrama de Venn en la variable 'c', puedo llamar a cada círculo individual c[0] y c[1] y establecer estilo y ancho de línea:

In [8]:

```
dv = venn2((1, 1, 1),set_colors=("#FF5733", "#68FF33"), alpha=0.5)
dv.get_label_by_id("11").set_text("Intersección")
dv.get_label_by_id("11").set_color("#000000")
dv.get_patch_by_id("01").set_color("#AA00FF")
dv.get_patch_by_id("10").set_color("#00FF00")
dv.get_patch_by_id("11").set_color("#FF00AA")

c = venn2_circles(subsets=(1, 1, 1), color="grey", alpha=0.8)

c[0].set_lw(3.0)
c[0].set_linestyle("-.")
c[1].set_lw(5.0)
c[1].set_ls('dotted')
plt.title ("Diagrama de Venn");
plt.show()
```

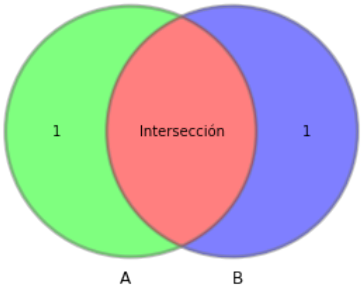


Ahora bien, lo que nos interesa es poder representar los valores. Estos pueden surgir de listas, tuplas, conjuntos , etc y como resultado de operaciones entre conjuntos.

Creamos dos conjuntos y los representamos....

In [9]:

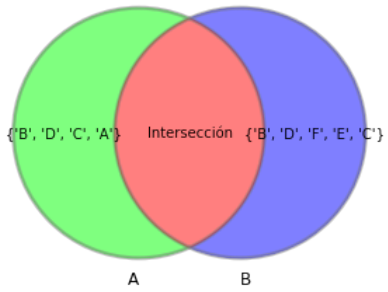
```
set1 = set(['A','B','C','D'])
set2 = set(['B','C','D', 'E', 'F'])

dv=venn2((1,1,1),set_colors=("#FF5733","#68FF33"),alpha=0.5)
dv.get_label_by_id("11").set_text("Intersección")
dv.get_label_by_id("11").set_color("#000000")
dv.get_patch_by_id("01").set_color("#0000FF")
dv.get_patch_by_id("10").set_color("#00FF00")
dv.get_patch_by_id("11").set_color("#FF0000")

c=venn2_circles(subsets=(1,1,1),color="gray",alpha=0.5,linestyle="-",linewidth=3)

plt.show()
```



Pero **NO** se visualizan cambios….Para visualizar los valores hay que pasar los **sets** a la propiedad **set_text()** de **get_label_by_id**:

In [10]:

```
set1 = set(['A','B','C','D'])
set2 = set(['B','C','D','E','F'])

dv=venn2((1,1,1),set_colors=("#FF5733","#68FF33"),alpha=0.5)
dv.get_label_by_id("11").set_text("Intersección")
dv.get_label_by_id("11").set_color("#000000")
dv.get_patch_by_id("01").set_color("#0000FF")
dv.get_patch_by_id("10").set_color("#00FF00")
dv.get_patch_by_id("11").set_color("#FF0000")

dv.get_label_by_id('10').set_text(set1)
dv.get_label_by_id('01').set_text(set2)

c=venn2_circles(subsets=(1,1,1),color="gray",alpha=0.5,linestyle="-",linewidth=3)

plt.show()
```



Este diagrama **NO** es lo que queríamos… **Realizamos operaciones entre conjuntos, eliminamos la palabra "Intersección" y realizamos algunas operaciones para mejorar el gráfico**:

In [11]:

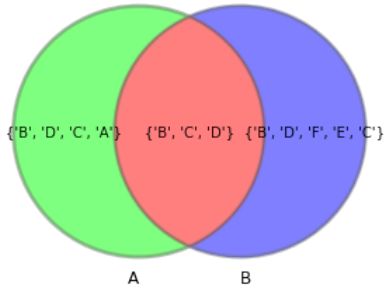
```
set1 = set(['A','B','C','D'])
set2 = set(['B','C','D','E','F'])
inter=set(set1.intersection(set2))
inter
```

Out[11]:

```
{'B', 'C', 'D'}
```

In [12]:

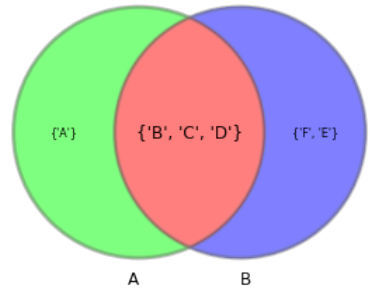
```
dv=venn2((1,1,1),set_colors=("#FF5733","#68FF33"),alpha=0.5)
dv.get_label_by_id("11").set_color("#000000")
dv.get_patch_by_id("01").set_color("#0000FF")
dv.get_patch_by_id("10").set_color("#00FF00")
dv.get_patch_by_id("11").set_color("#FF0000")

'''
En las siguientes líneas pasamos como parámetros al método set_text(), los resultados obtenidos de las
operaciones de creación de conjuntos e intersección.
'''

dv.get_label_by_id('10').set_text(set1)
dv.get_label_by_id('01').set_text(set2)
dv.get_label_by_id('11').set_text(inter)

c=venn2_circles(subsets=(1,1,1),color="gray",alpha=0.5,linestyle="-",linewidth=3)

plt.show()
```



Podemos destacar la intersección cambiándole el tamaño de fuente con **set_fontsize**:

In [13]:

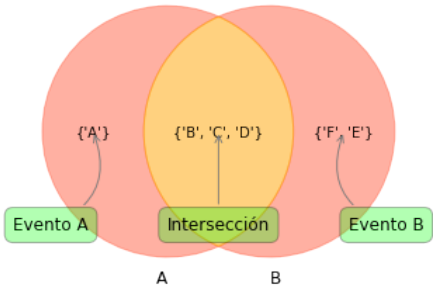
```
set1=set(['A','B','C','D'])
set2=set(['B','C','D','E','F'])
inter=set(set1.intersection(set2))
A=set1-inter
B=set2-inter

dv=venn2((1,1,1),set_colors=("#FF5733","#68FF33"),alpha=0.5)
dv.get_label_by_id("11").set_color("#000000")
dv.get_patch_by_id("01").set_color("#0000FF")
dv.get_patch_by_id("10").set_color("#00FF00")
dv.get_patch_by_id("11").set_color("#FF0000")
dv.get_label_by_id('10').set_text(A)
dv.get_label_by_id('10').set_fontsize(8)
dv.get_label_by_id('01').set_text(B)
dv.get_label_by_id('01').set_fontsize(8)
dv.get_label_by_id('11').set_text(inter)
dv.get_label_by_id('11').set_fontsize(12)

c=venn2_circles(subsets=(1,1,1),color="gray",alpha=0.5,linestyle="-",linewidth=3)
plt.show()
```



Utilizando la función **annotate()** de matplotlib agregamos anotaciones:

In [14]:

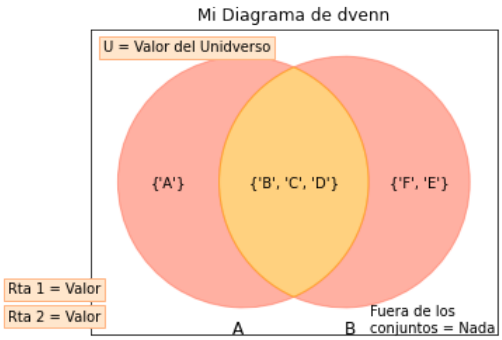
```
set1=set(['A','B','C','D'])
set2=set(['B','C','D','E','F'])
inter=set(set1.intersection(set2))
A=set1-inter
B=set2-inter

dv = venn2(subsets = {'10': 1, '01': 1, '11': 1}, set_labels = ('A', 'B'))
dv.get_patch_by_id('10').set_alpha(0.5)
dv.get_patch_by_id('10').set_color('tomato')
dv.get_patch_by_id('01').set_alpha(0.5)
dv.get_patch_by_id('01').set_color('tomato')
dv.get_patch_by_id('11').set_alpha(0.5)
dv.get_patch_by_id('11').set_color('orange')
dv.get_label_by_id('10').set_text(A)
dv.get_label_by_id('01').set_text(B)
dv.get_label_by_id('11').set_text(inter)

plt.annotate('Evento A', xy = dv.get_label_by_id('10').get_position(), xytext = (-30,-70), size = 'large',
             ha = 'center', textcoords = 'offset points', bbox = dict(boxstyle = 'round, pad = 0.5',
                                                                      fc = 'lime', alpha = 0.3),
             arrowprops = dict(arrowstyle = '->', connectionstyle = 'arc3, rad = 0.5', color = 'gray'))

plt.annotate('Evento B', xy = dv.get_label_by_id('01').get_position(), xytext = (30,-70), size = 'large',
             ha = 'center', textcoords = 'offset points', bbox = dict(boxstyle = 'round, pad = 0.5',
                                                                      fc = 'lime', alpha = 0.3),
             arrowprops = dict(arrowstyle = '->', connectionstyle = 'arc3, rad = -0.5',color = 'gray'))

plt.annotate('Intersección', xy = dv.get_label_by_id('11').get_position(), xytext = (0,-70), size = 'large',
             ha = 'center', textcoords = 'offset points', bbox = dict(boxstyle = 'round, pad = 0.5',
                                                                      fc = 'lime', alpha = 0.3),
             arrowprops = dict(arrowstyle = '->', connectionstyle = 'arc3,rad = 0',color = 'gray'))

plt.show()
```



Establecer el conjunto Universal. Con la función **plt.text()** agregamos anotaciones y con **plt.axis()** formamos el recuadro:

In [15]:

```
set1=set(['A','B','C','D'])
set2=set(['B','C','D','E','F'])
inter=set(set1.intersection(set2))
A=set1-inter
B=set2-inter

dv = venn2(subsets = {'10': 1, '01': 1, '11': 1}, set_labels = ('A', 'B'))
dv.get_patch_by_id('10').set_alpha(0.5)
dv.get_patch_by_id('10').set_color('tomato')
dv.get_patch_by_id('01').set_alpha(0.5)
dv.get_patch_by_id('01').set_color('tomato')
dv.get_patch_by_id('11').set_alpha(0.5)
dv.get_patch_by_id('11').set_color('orange')
dv.get_label_by_id('10').set_text(A)
dv.get_label_by_id('01').set_text(B)
dv.get_label_by_id('11').set_text(inter)

plt.text(-1.05, -0.42,
         s="Rta 1 = " + str('Valor'),size=10,ha="left",va="bottom",
         bbox=dict(boxstyle="square",ec=(1.0, 0.7, 0.5),fc=(1.0, 0.9, 0.8),))

plt.text(-1.05, -0.52,
         s="Rta 2 = " + str('Valor'),size=10,ha="left",va="bottom",
         bbox=dict(boxstyle="square",ec=(1.0, 0.7, 0.5),fc=(1.0, 0.9, 0.8),))

plt.text(-0.70, 0.52,
         s="U = " + str('Valor del Unidverso'),
         size=10,ha="left",va="top",bbox=dict(boxstyle="square",  # tipo de cuadro
                   ec=(1.0, 0.7, 0.5),
                   fc=(1.0, 0.9, 0.8),))
plt.text(0.28, -0.55,
         s="Fuera de los\nconjuntos = " + str('Nada'),
         size=10)

plt.axis('on')
plt.title("Mi Diagrama de dvenn")
plt.show()
```



Símbolos

Símbolo	Descripción	Valor para este Problema
$n(A)$	El número de elementos en el conjunto A	125
$n(A \cap B)$	El número de elementos en la intersección de los conjuntos A y B (todos los elementos que están en la superposición de ambos conjuntos)	52
$n(A \cup B)$	El número de elementos en la unión de los conjuntos A y B (todos los elementos que están en uno o en ambos conjuntos)	330
$n(A')$	El número de elementos en el complemento de A (el número de elementos fuera del conjunto) A)	375
$n((A \cup B)')$	El número de elementos en el complemento de $A \cup B$ (el número de elementos fuera del conjunto A y B)	170
$n((A \cap B)')$	El número de elementos en el complemento de $A \cap B$ (todo lo que esta fuera de la intersección de A y B)	448
$n(A \cap B')$	El número de elementos en la intersección de los complementos de A y B (el número de elementos en A pero no en B)	73

Ejercicio.

1. En una escuela de 500 estudiantes, hay:

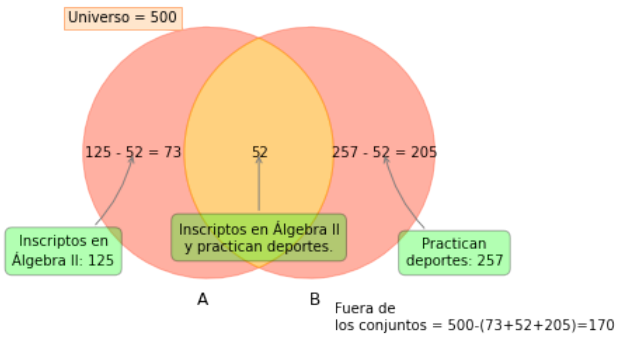
- 125 estudiantes inscriptos en Álgebra II,
- 275 estudiantes que practican deportes y
- 52 estudiantes que están inscriptos en Álgebra II y practican deportes.

Crear un diagrama de Venn para ilustrar esta información.

Solución: Primero, se establece que el conjunto representa los estudiantes inscriptos en Álgebra II y el conjunto que representa los estudiantes que practican deportes. Generalmente hablando, es más fácil empezar en el centro o en la "intersección" del diagrama de Venn. Una vez que se ubica el 52 en la intersección, podemos restarlo al número total de estudiantes que practican deporte y al número total de estudiantes inscriptos en Álgebra II para determinar cuántos solo hacen una actividad o la otra. Finalmente, podemos restar este total a 500 para encontrar cuántos están completamente fuera de los círculos.

In [17]:

```
from matplotlib_venn import *
from matplotlib import pyplot as plt

U=500

v = venn2(subsets = {'10': 1, '01': 1, '11': 1}, set_labels = ('A', 'B'))
v.get_patch_by_id('10').set_alpha(0.5)
v.get_patch_by_id('10').set_color('tomato')
v.get_patch_by_id('01').set_alpha(0.5)
v.get_patch_by_id('01').set_color('tomato')
v.get_patch_by_id('11').set_alpha(0.5)
v.get_patch_by_id('11').set_color('orange')
v.get_label_by_id('10').set_text('125 - 52 = 73')
v.get_label_by_id('01').set_text('257 - 52 = 205')
v.get_label_by_id('11').set_text('52')

plt.text(-0.70, 0.52,
         s="Universo = " + str(f'{U}'),
         size=10,ha="left",va="top",bbox=dict(boxstyle="square",  # tipo de cuadro
                   ec=(1.0, 0.7, 0.5),
                   fc=(1.0, 0.9, 0.8),))

plt.annotate('Inscriptos en\nÁlgebra II: 125', xy = v.get_label_by_id('10').get_position(), xytext = (-50,-80),
             size = 'medium', ha = 'center', textcoords = 'offset points', bbox = dict(boxstyle = 'round, pad = 0.5',
             fc = 'lime', alpha = 0.3), arrowprops = dict(arrowstyle = '->', connectionstyle = 'arc3, rad = 0.2',
                                                          color = 'gray'))

plt.annotate('Practican\ndeportes: 257', xy = v.get_label_by_id('01').get_position(), xytext = (50,-80),
             size = 'medium', ha = 'center', textcoords = 'offset points', bbox = dict(boxstyle = 'round, pad = 0.5',
            fc = 'lime', alpha = 0.3),arrowprops = dict(arrowstyle = '->', connectionstyle = 'arc3, rad = -0.2',
                                                        color = 'gray'))

plt.annotate('Inscriptos en Álgebra II\ny practican deportes.', xy = v.get_label_by_id('11').get_position(),
             xytext = (0,-70), size = 'medium', ha = 'center', textcoords = 'offset points',
             bbox = dict(boxstyle = 'round, pad = 0.5', fc = 'lime', alpha = 0.3),
            arrowprops = dict(arrowstyle = '->', connectionstyle = 'arc3,rad = 0',color = 'gray'))

# Valor de los que quedan afuera
plt.text(0.28, -0.65,
         s="Fuera de\nlos conjuntos = " + str('500-(73+52+205)=170'),
         size=10)
# plt.axis('on')
plt.show()
```

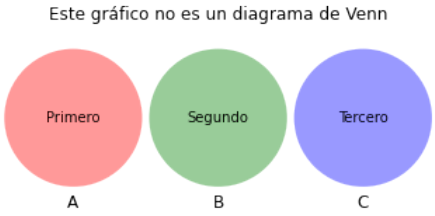


2. En una escuela de 600 alumnos, 100 no estudian ningún idioma extranjero, 450 estudian francés y 50 estudian francés e inglés. ¿Cuántos estudian solo inglés?

Y si necesitamos representar más de 2 conjuntos? Tenemos venn3

In [18]:

```
from matplotlib_venn import venn3

dv = venn3(subsets=(1,1,0,1,0,0,0))
dv.get_label_by_id('100').set_text('Primero')
dv.get_label_by_id('010').set_text('Segundo')
dv.get_label_by_id('001').set_text('Tercero')

plt.title("Este gráfico no es un diagrama de Venn")
plt.show()
```

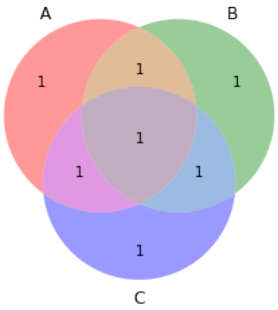


La función venn3() es esencialmente similar a venn2(). El primer argumento será ahora una tupla de 7 elementos que se corresponden con los siete subconjuntos en el siguiente orden:

- **Abc**
- **aBc**
- **ABc**
- **abC**
- **AbC**
- **aBC**
- **ABC**

In [19]:

```
dv=venn3((1,1,1,1,1,1,1))
plt.show()
```



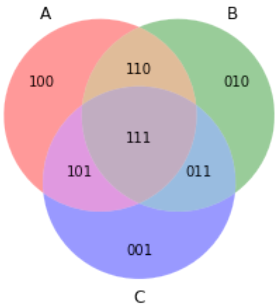
Para identificar a cada uno de los subconjuntos se emplean ahora tres dígitos según el orden A, B, C.

La imagen que vamos a generar a continuación muestra cada uno de ellos con su respectivo identificador.

Podemos establecer las etiquetas utilizando un ciclo for:

In [20]:

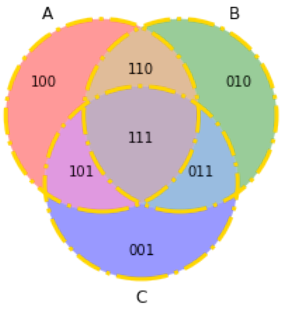
```
dv=venn3((1,1,1,1,1,1,1))

for subset in("111","110","101","100","011","010","001"):
    dv.get_label_by_id(subset).set_text(subset)
plt.show()
```



Para venn3 también aplican los atributos **alpha, set_labels y set_colors**, como en venn2. Delineamos los círculos:

In [21]:

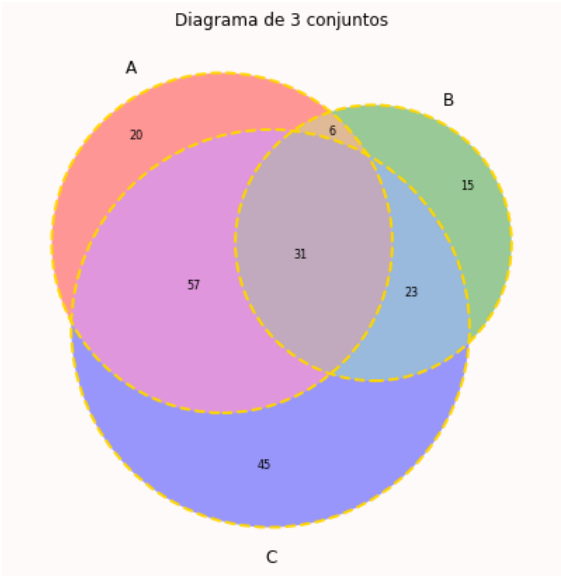
```
from matplotlib_venn import venn3, venn3_circles

dv=venn3((1,1,1,1,1,1,1))
for subset in("111","110","101","100","011","010","001"):
    dv.get_label_by_id(subset).set_text(subset)

venn3_circles(subsets=(1,1,1,1,1,1,1),color="gold",alpha=1,linestyle="-.",linewidth=3)
plt.show()
```



Distribuimos los valores, definimos el tamaño de la ventana, color de fondo del gráfico y agregamos un título:

In [22]:

```
plt.figure(figsize=(7,7),facecolor='snow')

dv=venn3(subsets=(20,15,6,45,57,23,31),set_labels=('A','B','C'))

for subset in ("111","110","101","100","011","010","001"):
    dv.get_label_by_id(subset).set_fontsize(8)

venn3_circles(subsets=(20,15,6,45,57,23,31),color="gold",alpha=1,linestyle='dashed',linewidth=2)

plt.title("Diagrama de 3 conjuntos")
plt.show()
```

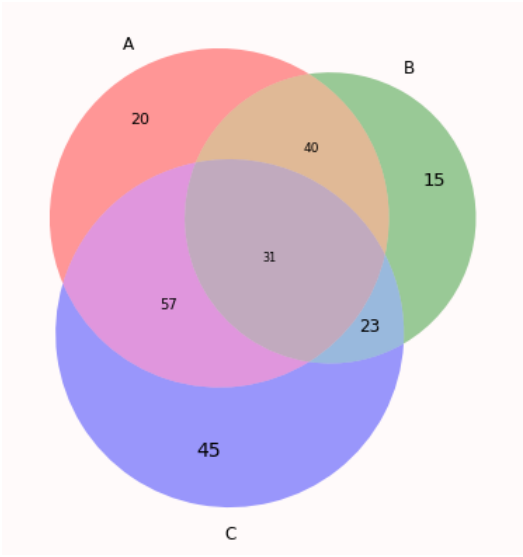


Nota: para este ejemplo se usaron distintos valores en subsets para la mejor comprensión del diagrama.
Agregando la instrucción: plt.savefig("nombre.extensión"), se puede generar un archivo pdf, png, o jpg de la imagen.

Podemos manejar el tamaño de la fuente de la siguiente manera:

In [23]:

```
plt.figure(figsize=(7,7), facecolor='snow')
dv = venn3(subsets=(20, 15, 40, 45, 57, 23, 31), set_labels = ('A', 'B', 'C'))

cont=8
for subset in ("111", "110", "101", "100", "011", "010", "001"):
    dv.get_label_by_id(subset).set_fontsize(cont)
    cont = cont+1

plt.show()
```



Si necesitamos representar un área desconocida, podemos hacerlo de la siguiente manera:

In [24]:

```
plt.figure(figsize=(7,7))
dv = venn3(subsets=(3, 1, 1, 1, 1, 1, 1), set_labels = ('A', 'B', 'C'))

for subset in ("111", "110", "101", "100", "011", "010", "001"):
    dv.get_label_by_id(subset).set_text(subset)

dv.get_patch_by_id('100').set_alpha(1.0)
dv.get_patch_by_id('100').set_color('white')
dv.get_label_by_id('100').set_text('Desconocido')
dv.get_label_by_id('A').set_text('A')
c = venn3_circles(subsets=(3, 1, 1, 1, 1, 1, 1), color="gray", alpha=1, linestyle='dashed',
                  linewidth=3)

c[0].set_lw(3.0)
c[0].set_ls('dotted')

plt.title("Diagrama con área desconocida")
plt.show()
```

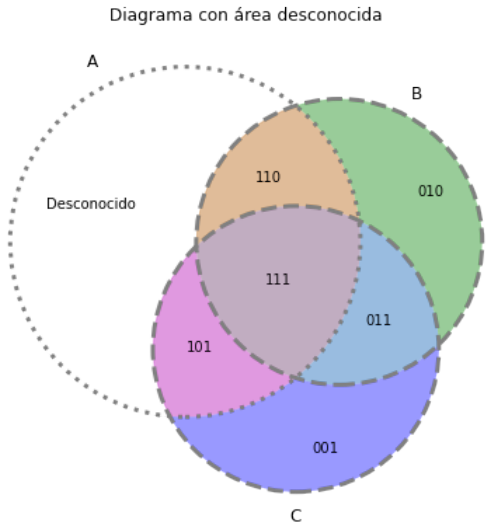
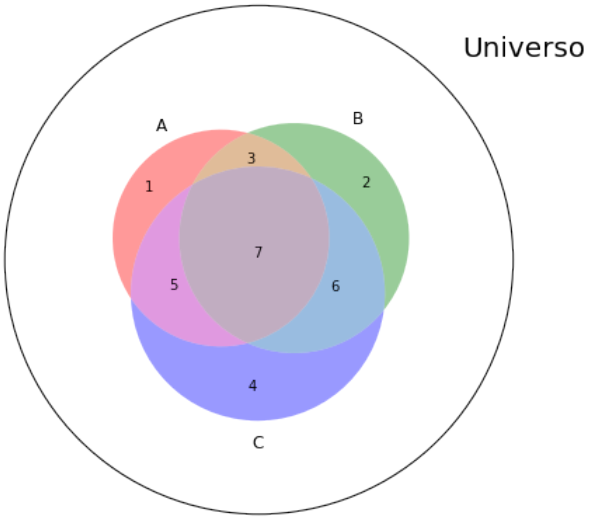


Diagrama incrustado en un círculo:

In [25]:

```
plt.figure(figsize=(7,7))
dv = venn3(subsets=(1,2,3,4,5,6,7), set_labels = ('A', 'B', 'C'))

from matplotlib.patches import Circle

plt.gca().add_patch(Circle([0,0], 1, fill=False))

plt.xlim(-1.05,1.05)
plt.ylim(-1.05,1.05)
plt.text(0.8, 0.8, 'Universo', fontsize=20)
plt.show()
```



Ejercicios.

1. Crea un diagrama de Venn para representar la información siguiente y responder las preguntas:
En una encuesta a 15 estudiantes secundarios, se descubrió que:

- 80 estudiantes tienen laptops.
- 110 estudiantes tienen celulares.
- 125 estudiantes tienen iPod
- 62 estudiantes tienen una laptop y un celular.
- 58 estudiantes tienen una laptop y un iPod.
- 98 estudiantes tienen un celular y un iPod.
- 50 estudiantes tienen los tres objetos.

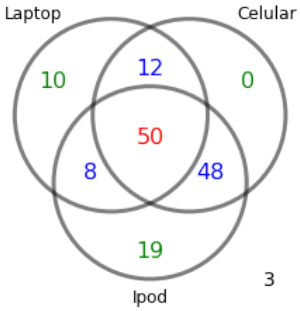
Responde:
a. ¿Cuántos estudiantes tienen solo un celular?
b. ¿Cuántos estudiantes no tienen ninguno de los objetos mencionados?
c. ¿Cuántos estudiantes tienen un iPod y una laptop, pero no un celular?

Solución: Primero, usaremos la información dada para construir el diagrama de Venn, como se muestra a continuación:

Podemos comenzar con escribir 50 en el centro, donde los estudiantes tienen los 3 objetos. Luego, podemos encontrar los valores en azul al restar 50 de cada uno de los valores "superpuestos". Por ejemplo, hay 62 estudiantes con una laptop y un celular y 50 de ellos también tienen un iPod. Para encontrar el número de los que tienen laptop y celular pero no iPod, resta 62 – 50 = 12 . Una vez que encuentres los valores azules, podemos encontrar los valores verdes al restar los valores azules y rojos en cada subconjunto del total en el subconjunto. Por ejemplo, el número de estudiantes con un celular pero no otro objeto tecnológico es 110 - (50 + 12 + 48) = 0. Finalmente, podemos sumar todos los valores en los círculos y restar esto de 150, el número de estudiantes encuestados, para determinar que 3 estudiantes no tienen ninguno de estos objetos.

In [26]:

```
# dibujamos los diagramas
diagram = venn3((1, 1, 1, 1, 1, 1, 1), set_labels=(
    "Laptop", "Celular", "Ipod"), set_colors=("#FFFFFF", "#FFFFFF", "#FFFFFF"))

# establecemos el tamaño de la fuente
for subset in ("111", "110", "101", "100", "011", "010", "001"):
    diagram.get_label_by_id(subset).set_fontsize(16)

c = venn3_circles(subsets=(1, 1, 1, 1, 1, 1, 1), color="#000000", alpha=0.5, linewidth=3)

# transferimos los resultados de las operaciones
diagram.get_label_by_id('100').set_text('10')
diagram.get_label_by_id('100').set_color('green')
diagram.get_label_by_id('010').set_text('0')
diagram.get_label_by_id('010').set_color('green')
diagram.get_label_by_id('001').set_text('19')
diagram.get_label_by_id('001').set_color('green')
diagram.get_label_by_id('110').set_text('12')
diagram.get_label_by_id('110').set_color('blue')
diagram.get_label_by_id('011').set_text('48')
diagram.get_label_by_id('011').set_color('blue')
diagram.get_label_by_id('101').set_text('8')
diagram.get_label_by_id('101').set_color('blue')
diagram.get_label_by_id('111').set_text('50')
diagram.get_label_by_id('111').set_color('red')

plt.text(0.50, -0.65, s='3',size=14)
plt.show()
```



Ahora que el diagrama de Venn está completo, podemos utilizarlo para responder las preguntas.
a. Hay 0 estudiantes que solo tienen un celular.
b. Hay 3 estudiantes con ningún objeto tecnológico mencionado.
c. Hay 8 estudiantes con un iPod y una laptop pero no un celular.

2. En una encuesta de 80 dueños de casa, se descubrió que:

- 30 tenían al menos un perro.
- 42 tenían al menos un gato.
- 21 tenían al menos una mascota "otra" (pez, tortuga, reptil, hámster, etc.).
- 20 Tenían perro(s) y gato (s).
- 10 tenían gato(s) y mascota(s) otra.
- 8 tenían perro(s) y mascota(s) otra.
- 5 tenían los tres tipos de mascotas.

Haz un diagrama de Venn para ilustrar los resultados de la encuesta y responde:

a. ¿Cuántos tenían perro(s) y gato(s) pero no mascota(s) "otra"?
b. ¿Cuántos solo tenían perro(s)?
c. ¿Cuántos no tenían mascotas?
d. ¿Cuántos dueños de mascota(s) otra también tenían perro(s) o gato(s), pero no ambos?

3. En una encuesta realizada en la ciudad de Buenos Aires, acerca de los medios de transporte más utilizados entre colectivos, subte o moto, se obtuvieron los siguientes resultados: de los 3200 encuestados, 1950 utilizan el subte, 400 se desplazan en moto, 1500 van en colectivo, 800 se desplazan en colectivo y subte, además ninguno de los que se transporta en moto utiliza colectivo o subte.

a. El número de personas que solo utiliza el subte es….
b. Las persona que solo utilizan máximo 2 medios de transporte son…

3. Se encuesta a 150 familias consultando por el nivel educacional actual de sus hijos. Los resultados obtenidos son:

- 10 familias tienen hijos en Enseñanza Básica, Enseñanza Media y Universitaria.
- 16 familias tienen hijos en Enseñanza Básica y Universitaria
- 30 familias tienen hijos en Enseñanza Media y Enseñanza Básica.
- 22 familias tienen hijos en Enseñanza Media y Universitaria.
- 72 familias tienen hijos en Enseñanza Media.
- 71 familias tienen hijos en Enseñanza Básica.
- 38 familias tienen hijos en Enseñanza Universitaria.

Con la información anterior, deducir:

a. El número de familias que solo tienen hijos universitarios.
b. El número de familias que tienen hijos solo en dos niveles.
c. El número de familias que tienen hijos que no estudian.

4. Una encuesta sobre 500 estudiantes inscriptos en una o más asignaturas de Matemática, Física y Química durante un semestre, reveló los siguientes números de estudiantes en los cursos indicados: Matemática 329, Física 186, Química 295, Matemática y Física 83, Matemática y Química 217, Física y Química 63. Cuántos alumnos estarán inscriptos en:

a) Los tres cursos
b) Matemática pero no Química
c) Física pero no matemática
d) Química pero no Física
e) Matemática o Química, pero no Física
f) Matemática y Química, pero no Física
g) Matemática pero no Física ni Química

Más información (https://datascienceplus.com/how-to-visualize-complex-sets-intersections-with-python/)

Ejemplos de aplicaciones de Diagramas de Venn

Tablas de contingencia, diagramas de Venn y probabilidad (https://es.khanacademy.org/math/ap-statistics/probability-ap/probability-addition-rule/e/two-way-tables-venn-diagrams-probability)

Tablas de contingencia de frecuencias y diagramas de Venn (https://es.khanacademy.org/math/cc-eighth-grade-math/cc-8th-data/two-way-tables/v/two-way-frequency-tables-and-venn-diagrams)

Introducción a la estadística empresarial - Diagrama de Venn (https://openstax.org/books/introducci%C3%B3n-estad%C3%ADstica-empresarial/pages/3-5-diagramas-de-venn#:~:text=Un%20diagrama%20de%20Venn%20es,c%C3%ADrculos%20u%20%C3%B3valos%20representan%20eventos.)

Resolución de problemas con Diagramas de Venn (http://www.joseluislorente.es/3eso/probabilidad/5_resolucin_de_problemas_por_diagramas_de_venn.html)