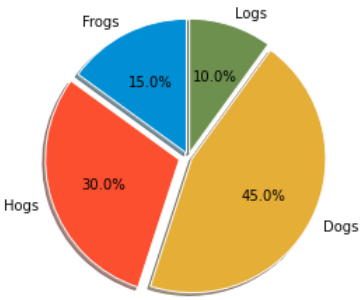
In [ ]:

Pie Chart- Pasta Grafiği:

Tanım:

Pie-Pasta grafiği birden fazla değişkenin bir bütün içerisindeki oranını belirtmek için kullanılan grafik türüdür. Her bir porsiyon bir değiişkeni temsil eder. Genellikle % değerleri göstermek için kullanılır.

In [20]:
```
import matplotlib.pyplot as plt

#plt.style.use("fivethirtyeight")

# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
colors =['#008fd5','#fc4f30','#e5ae37','#6d904f']

explode = (0.03, 0.07, 0.03, 0.03)  # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',colors=colors,
        shadow=True, startangle=90,wedgeprops={'edgecolor':'white'})
#Shadow makes 3d effect
#startangle starts from 90 degree
# autopct='%1.1f%% adds percentes in
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```

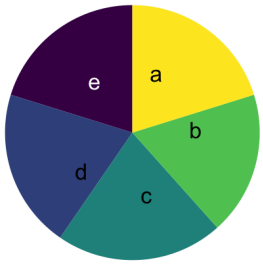


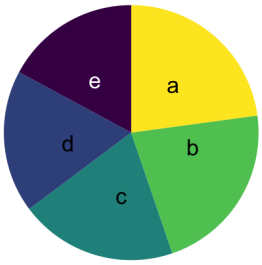
Ne için Kullanılır:

Pie-Pasta grafiği birden fazla değişkenin bir bütün içerisindeki oranını belirtmek için kullanılan grafik türüdür. Her bir porsiyon bir değiişkeni temsil eder. Genellikle % değerleri göstermek için kullanılır.

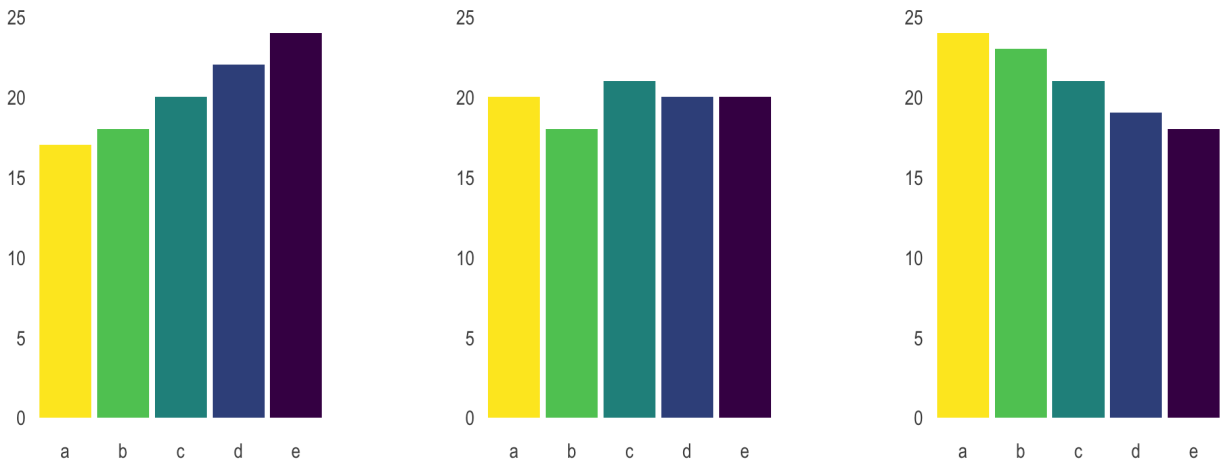
Çok bariz farklarda kullabilir veya bir porsiyondaki veriler yanda daha detaylı gösterilebilir , aksi durumda aşağıdaki hataya neden olabilir.

Bu görsel **Pasta grafiğidir** genel görselidir.





Bu görsel **Pasta grafiğindeki ayrıntıları daha net gösteren bar grafiği** genel görselidir.



Çeşitleri:

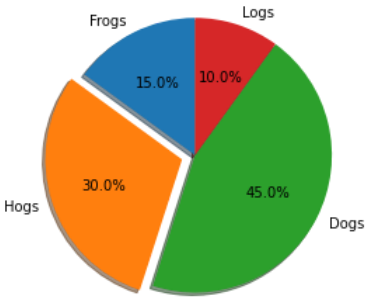
Pie Chart: Pasta grafiğidir ve yukarıda tanımlanmıştır.

Donut Chart: İçi boş pasta grafiğidir tamamen aynı özellikler içierir sadece görsel açıdan içersine bilgi eklenebilir aolması dolayısıyla pasta grafiğinden ayrılır.

Nested Pie Chert: Nested Pie grafiği birden fazla değişkenin bir bütün içerisindeki oranını belirtmenin ötesinde alt kategorileri de belirtmek için kullanılan grafik türüdür. Her bir porsiyon bir değiişkeni temsil eder. Genellikle % değerleri göstermek için kullanılır. Alt kategoriler ana kategorinin içinde gösterilerek grafik detaylandırılır.

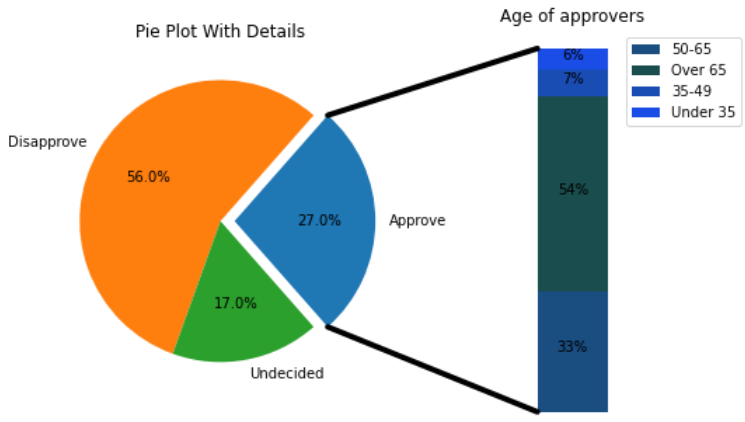
Pasta grafiğinin alt kategörileri daha ileride anlatılacağından burada detaya girilmeyecektir.

In [11]:
```
import matplotlib.pyplot as plt

# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0)  # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```

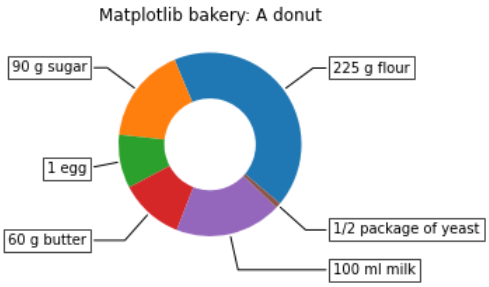


In [12]:

```
import matplotlib.pyplot as plt
from matplotlib.patches import ConnectionPatch
import numpy as np

# make figure and assign axis objects
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(9, 5))
fig.subplots_adjust(wspace=0)

# pie chart parameters
ratios = [.27, .56, .17]
labels = ['Approve', 'Disapprove', 'Undecided']
explode = [0.1, 0, 0]
# rotate so that first wedge is split by the x-axis
angle = -180 * ratios[0]
ax1.pie(ratios, autopct='%1.1f%%', startangle=angle,
        labels=labels, explode=explode)

# bar chart parameters

xpos = 0
bottom = 0
ratios = [.33, .54, .07, .06]
width = .2
colors = [[.1, .3, .5], [.1, .3, .3], [.1, .3, .7], [.1, .3, .9]]

for j in range(len(ratios)):
    height = ratios[j]
    ax2.bar(xpos, height, width, bottom=bottom, color=colors[j])
    ypos = bottom + ax2.patches[j].get_height() / 2
    bottom += height
    ax2.text(xpos, ypos, "%d%%" % (ax2.patches[j].get_height() * 100),
             ha='center')

ax1.set_title("Pie Plot With Details")
ax2.set_title('Age of approvers')
ax2.legend(('50-65', 'Over 65', '35-49', 'Under 35'))
ax2.axis('off')
ax2.set_xlim(- 2.5 * width, 2.5 * width)

# use ConnectionPatch to draw lines between the two plots
# get the wedge data
theta1, theta2 = ax1.patches[0].theta1, ax1.patches[0].theta2
center, r = ax1.patches[0].center, ax1.patches[0].r
bar_height = sum([item.get_height() for item in ax2.patches])

# draw top connecting line
x = r * np.cos(np.pi / 180 * theta2) + center[0]
y = r * np.sin(np.pi / 180 * theta2) + center[1]
con = ConnectionPatch(xyA=(-width / 2, bar_height), coordsA=ax2.transData,
                      xyB=(x, y), coordsB=ax1.transData)
con.set_color([0, 0, 0])
con.set_linewidth(4)
ax2.add_artist(con)

# draw bottom connecting line
x = r * np.cos(np.pi / 180 * theta1) + center[0]
y = r * np.sin(np.pi / 180 * theta1) + center[1]
con = ConnectionPatch(xyA=(-width / 2, 0), coordsA=ax2.transData,
                      xyB=(x, y), coordsB=ax1.transData)
con.set_color([0, 0, 0])
ax2.add_artist(con)
con.set_linewidth(4)

plt.show()
```



In [13]:
```
fig, ax = plt.subplots(figsize=(6, 3), subplot_kw=dict(aspect="equal"))

recipe = ["225 g flour",
          "90 g sugar",
          "1 egg",
          "60 g butter",
          "100 ml milk",
          "1/2 package of yeast"]

data = [225, 90, 50, 60, 100, 5]

wedges, texts = ax.pie(data, wedgeprops=dict(width=0.5), startangle=-40)

bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72)
kw = dict(arrowprops=dict(arrowstyle="-"),
          bbox=bbox_props, zorder=0, va="center")

for i, p in enumerate(wedges):
    ang = (p.theta2 - p.theta1)/2. + p.theta1
    y = np.sin(np.deg2rad(ang))
    x = np.cos(np.deg2rad(ang))
    horizontalalignment = {-1: "right", 1: "left"}[int(np.sign(x))]
    connectionstyle = "angle,angleA=0,angleB={}".format(ang)
    kw["arrowprops"].update({"connectionstyle": connectionstyle})
    ax.annotate(recipe[i], xy=(x, y), xytext=(1.35*np.sign(x), 1.4*y),
                horizontalalignment=horizontalalignment, **kw)

ax.set_title("Matplotlib bakery: A donut")

plt.show()
```



In [ ]:

In [14]:
```
#The effect of the donut shape is achieved by setting a width to the pie's wedges through the wedgeprops argument.

fig, ax = plt.subplots()

size = 0.3
vals = np.array([[60., 32.], [37., 40.], [29., 10.]])

cmap = plt.get_cmap("tab20c")
outer_colors = cmap(np.arange(3)*4)
inner_colors = cmap([1, 2, 5, 6, 9, 10])

ax.pie(vals.sum(axis=1), radius=1, colors=outer_colors,
       wedgeprops=dict(width=size, edgecolor='w'))

ax.pie(vals.flatten(), radius=1-size, colors=inner_colors,
       wedgeprops=dict(width=size, edgecolor='w'))

ax.set(aspect="equal", title='Nested Pie Plot')
plt.show()
```

In [ ]:

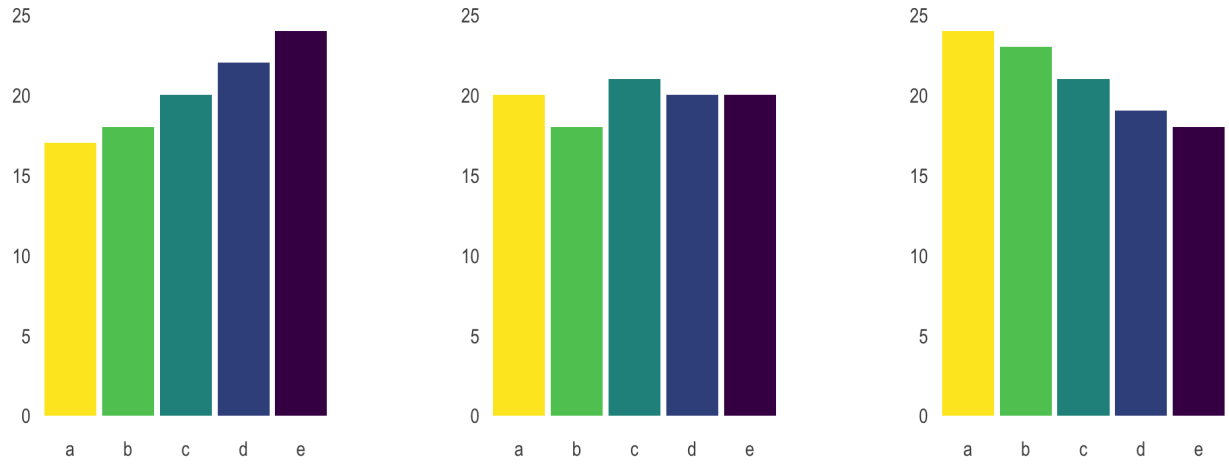
In [ ]:

Kullanım Hataları:

Pie Chart ile ilgili en önemli sorun insan gözünün açıları okumada oldukça beceriksiz olmasıdır. Değerleri biribirine yakın olan yüzdelerde aradaki farkı görmek oldukça sorunludur.

Bu görsel **Pasta grafiğidir** genel görselidir. Şekilde görüldüğü üzere grafikler arası fark neredeyse yok gibidir.

Bu görsel **Pasta grafiğindeki ayrıntıları daha net gösteren bar grafiği** genel görselidir. Yukarıdaki grafik Bar Chart olarak çizildiğinde arada bariz farklar olduğu görülmektedir.



Pie Chart **5'** den fazla değerin gösteriminde okunması zor bir grafiktir. Çok fazla parçaya bölünmüş bir pastanın kötü bir görünüm sergilemesi gibi....

In [ ]: