In [*]:
```
import gspread as gs
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from dateutil.parser import parse
sa = gs.service_account(filename="enhanced-victor-363219-68e4353f5f23.json")
sheet = sa.open("Untitled form (Responses)")
work_sheet = sheet.worksheet("Form Responses 1")
```

In [53]:
```
file1 = open('arrayCheckPoint.txt', 'r')
Lines = file1.readlines()
arrayCheck = []
for line in Lines:
    strList = line.strip().split(',')
    intList = [int(x) for x in strList if (x != '') & (x != 0)]

    arrayCheck.append(intList.sort())
len(arrayCheck)
```

Out[53]: 4187

In [54]:
```
# Using readlines()
file2 = open('coin.txt', 'r')
Lines = file2.readlines()
coin = []
for line in Lines:
    strList = line.strip().split(',')
    intList = [int(x) for x in strList if (x != '') & (x != 0)]
    coin.append(intList.sort())
len(coin)
```

Out[54]: 4187

In [55]:

```
df = pd.DataFrame(work_sheet.get_all_records())
df['arrive check point'] = arrayCheck
df['array get coin'] = coin
df['time elapse'] = pd.to_numeric(df['time elapse'])
df['a score'] = pd.to_numeric(df['a score'])
df['b score'] = pd.to_numeric(df['b score'])
df['total score'] = df['a score'] + df['b score']
df=df[df['time elapse'] <= 600]
df1 = df.dropna()
df1=df1.iloc[712:]
df1.head()
```

Out[55]:

	Timestamp	sessionID	a score	b score	time elapse	who caused death, 0 for left player, 1 for right, -1 none	death reason	level	true for pass level, false for died	arrive check point time array	coin get time array	ii v nc

In [29]:

```
def GetCoin(row):
    coins = row['array get coin']#.values[0]
    coinCnt=[]
    cnt=1
    for coin in coins:
        coinCnt.append(cnt)
        if (coin > 0):
            cnt+=1
    return coinCnt
df1['array coin'] = df1.apply (lambda row: GetCoin(row), axis=1)
df1.head()
```

In [40]:
```
df1
```

Out[40]:

	sessionID	a score	b score	time elapse	who caused death, 0 for left player, 1 for right, -1 none	death reason	level	true for pass level, false for died	arrive check point time array	c
	3017074808700032.0	3.0	3.0	25.0	1	Fall	Level0	FALSE	0	4
	3017074808700032.0	3.0	3.0	15.0	1	Fall	Level1	FALSE	0	
	3017074808700032.0	0.0	0.0	2.0	1	Touch different Color	Level1	FALSE	0	
	3017083046960000.0	0.0	1.0	22.0	1	Touch different Color	Level1	FALSE	0	
	3017083315590016.0	3.0	3.0	44.0	-1		Level0	TRUE	0	171
	...	...	...	...	...	...	...	...	...	
	38026520870023694	0.0	0.0	3.0	0	Touch different Color	Level1	FALSE	0	

	sessionID	a score	b score	time elapse	who caused death, 0 for left player, 1 for right, -1 none	death reason	level	true for pass level, false for died	arrive check point time array	c
	38028914761252638	0.0	0.0	20.0	-1		Level0.1	TRUE	101100000000	
	38028914761252638	0.0	0.0	227.0	1	Touch Enemy	Level0.1	FALSE	101100000000	
	38028914761252638	0.0	0.0	12.0	0	Touch different Color	Level0	FALSE	0	
	38028914761252638	0.0	0.0	22.0	1	Touch different Color	Level0	FALSE	0	

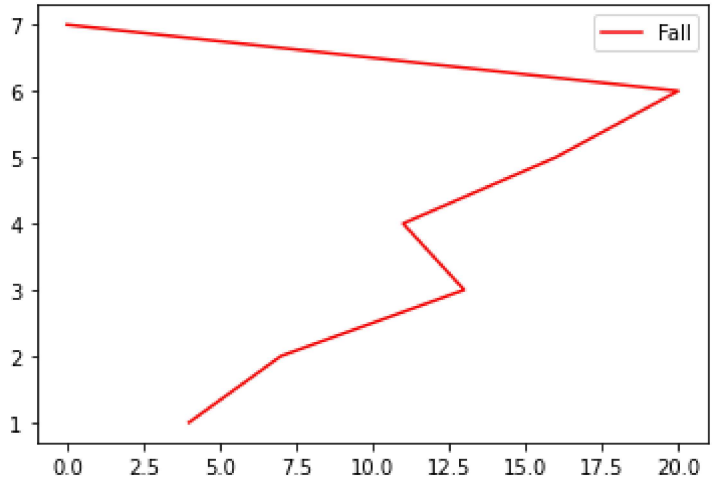
ns

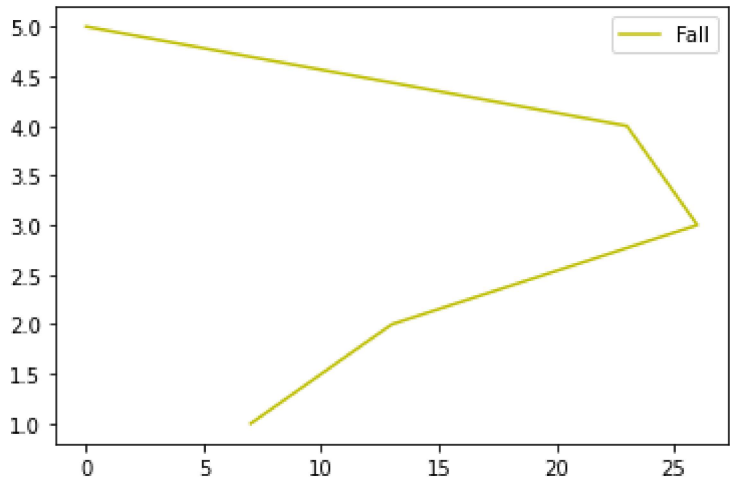
In [34]:

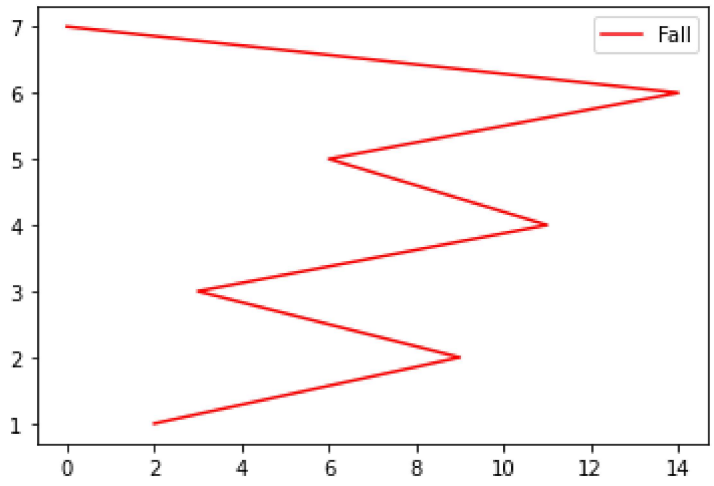
```
thisdict = {
  "Fall": "r",
  "Touch different Color": "g",
  "": "b",
    "Touch Enemy": "y"
}
```

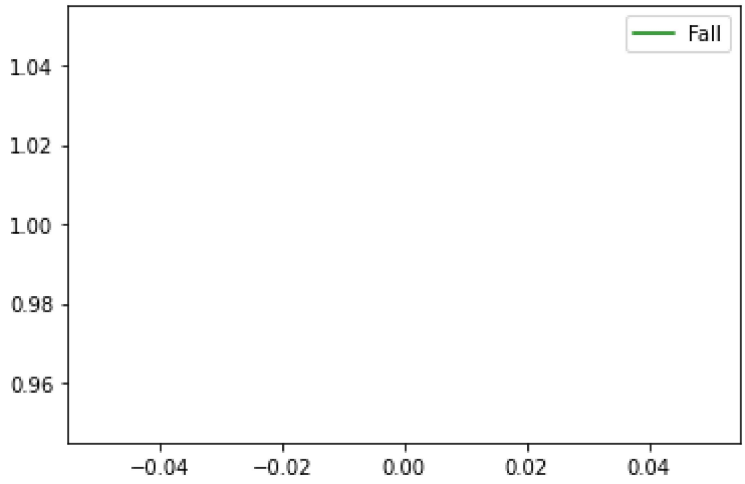
In [46]:

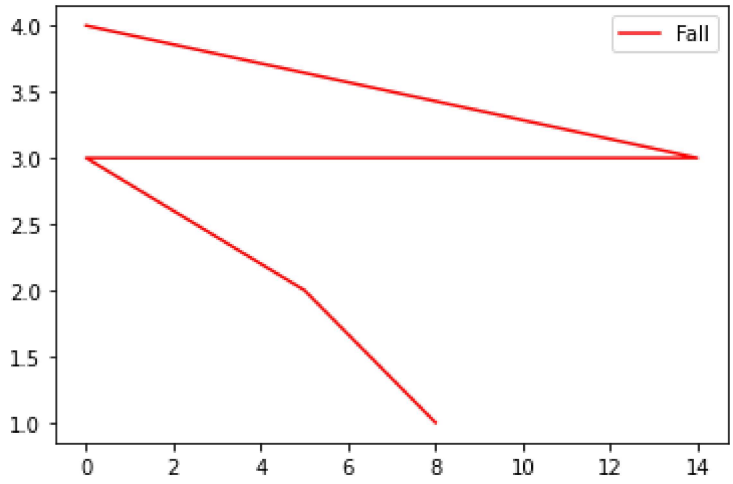
```
def plot(row):
    x = row['array get coin']
    y = row['array coin']
    death = row['death reason']
    plt.plot(x, y, thisdict[death])
for j in (['Level0', 'Level0.1', 'Level1', 'Level2', 'Level3']):
    level = df1[df1['level']==j]
    level.apply (lambda row: plot(row), axis=1)
    plt.legend(thisdict)
    plt.show()
```






Fall
7
6
5
4
3
2
1
0
2
4
6
8
10
12
14


Fall
1.04
1.02
1.00
0.98
0.96
−0.04
−0.02
0.00
0.02
0.04

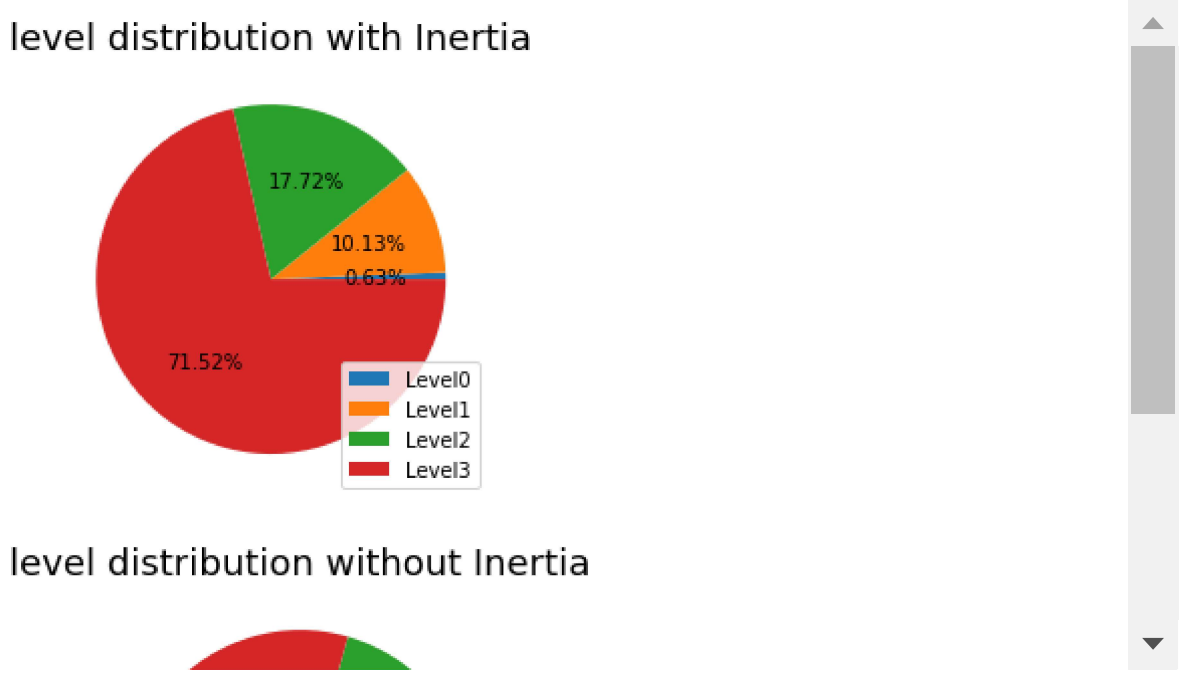


In [33]:
```
time = df1.groupby(['level', 'sessionID'])['time elapse'].apply(list)
score = df1.groupby(['level', 'sessionID'])['total score'].apply(list)
death = df1.groupby(['level', 'sessionID'])['death reason'].apply(list)
checktime=df1[(df1['arrive check point time array']!='') & (df1['arrive check
checktime['arrive check point time array']= checktime['arrive check point tim
checktime['arrive check point time array']
```

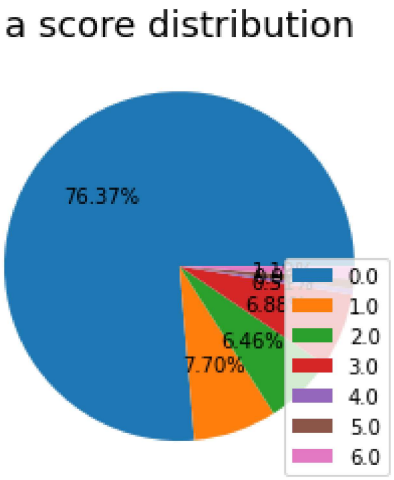
```
C:\Users\lucyg\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://p
andas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy)
  """
```

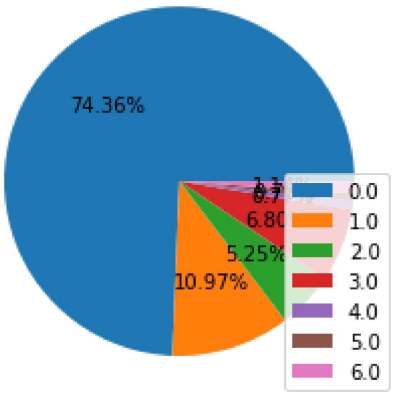
Out[33]:
```
744     142100000000
749      37000000000
751     141700000000
754     173100000000
762      46000000000
              ...
1929    102700000000
1933    132900000000
1935     58000000000
1936    213200000000
1938    112300000000
Name: arrive check point time array, Length: 426, dtype: object
```

In [3]:
```
truefalse=["TRUE", "FALSE"]
withinertia=['with Inertia', 'without Inertia']
for i in range(2):
        level=df1[df1['level']!='']
        level=level[level['inertia world for true, normal for false']==truefa
        df2 = level.groupby(['level']).size()
        plt.title("level distribution "+withinertia[i],fontsize=18)
        patches, text1, text2 = plt.pie(df2, autopct=lambda p: '{:.2f}%'.form
        plt.legend(patches, df2.index, loc="lower right")
        plt.show()
```



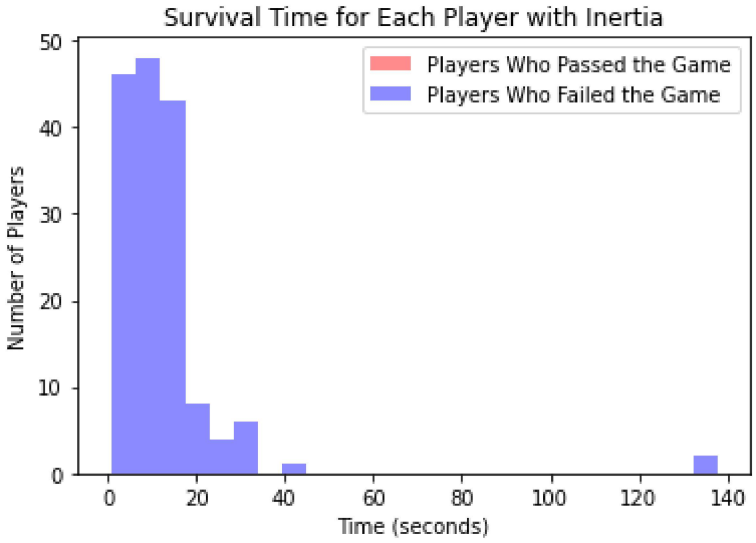
In [4]:
```
for i in (['a score', 'b score']):
    #level = df1[df1['level']==j]
    df2 = df1.groupby([i]).size()
    plt.title(i + " distribution",fontsize=18)
    patches, text1, text2 = plt.pie(df2, autopct=lambda p: '{:.2f}%'.format(p
    plt.legend(patches, df2.index, loc="lower right")
    plt.show()
```

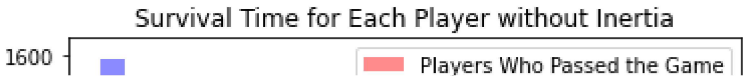




In [5]:

```
# plotting two histograms on the same axis
truefalse=["TRUE", "FALSE"]
withinertia=['with Inertia', 'without Inertia']
for i in range(2):
        passed=df1[df1['true for pass level, false for died'] == 'TRUE']
        passed=passed[passed['inertia world for true, normal for false']==tru
        notPassed=df1[df1['true for pass level, false for died'] == 'FALSE']
        notPassed=notPassed[notPassed['inertia world for true, normal for fal
        plt.hist(passed['time elapse'], bins=25, alpha=0.45, color='red')
        plt.hist(notPassed['time elapse'], bins=25, alpha=0.45, color='blue')

        plt.title("Survival Time for Each Player " + withinertia[i])

        plt.legend(['Players Who Passed the Game',
                    'Players Who Failed the Game'])
        plt.xlabel("Time (seconds)")
        plt.ylabel("Number of Players")
        plt.show()
```



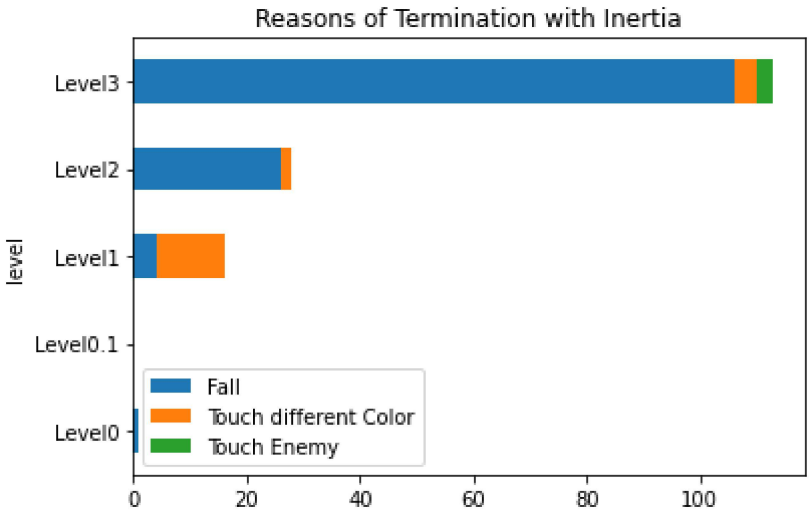


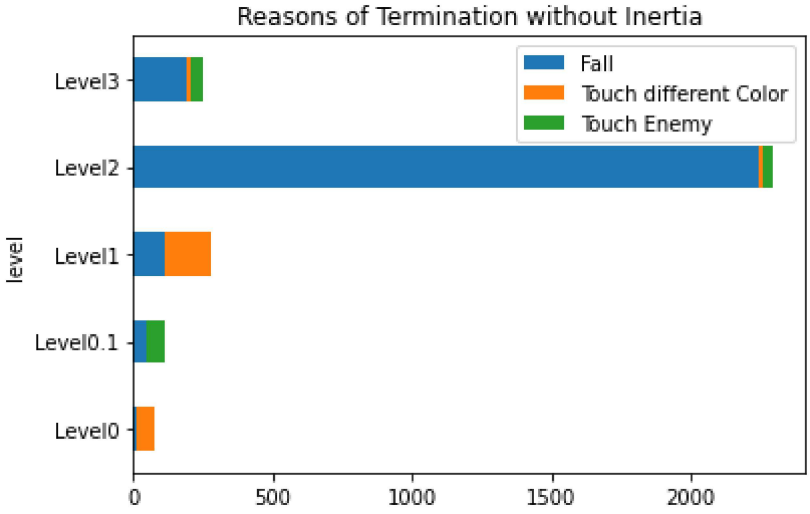
In [6]:

```
truefalse=["TRUE", "FALSE"]
withinertia=['with Inertia', 'without Inertia']
for i in range(2):
    inertia=df1[df1['inertia world for true, normal for false']==truefalse[i]
    data=[]
    for j in (['Level0', 'Level0.1', 'Level1', 'Level2', 'Level3']):
        level = inertia[inertia['level']==j]
        df3 = level[level['death reason'] != '']
        df4 = df3.groupby(['death reason']).size()
        dictionary=df4.to_dict()
        dictionary['level']=j
        data.append(dictionary)
    deathReason = pd.DataFrame(data)
    deathReason.plot(
        x = 'level',
        kind = 'barh',
        stacked = True,
        title = 'Reasons of Termination ' + withinertia[i], #stacked bar char
        mark_right = True)
    deathReason.set_index('level')
    display(deathReason)
```

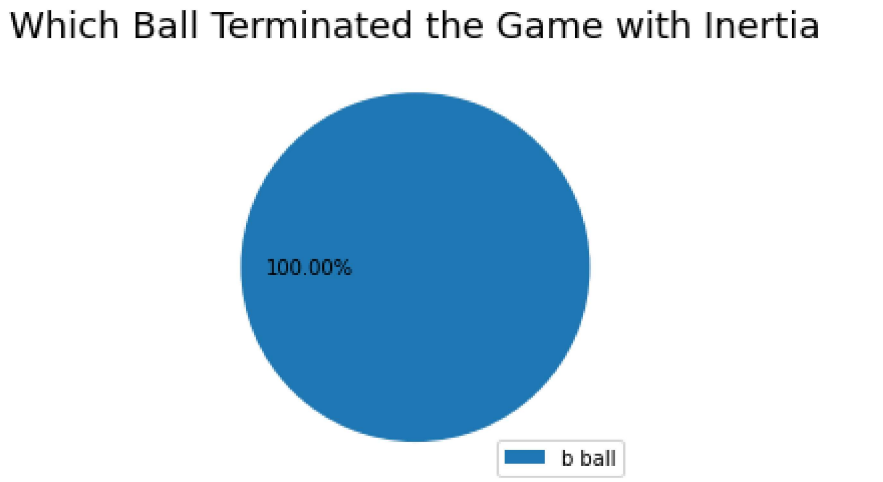
	Fall	level	Touch different Color	Touch Enemy
0	1.0	Level0	NaN	NaN
1	NaN	Level0.1	NaN	NaN
2	4.0	Level1	12.0	NaN
3	26.0	Level2	2.0	NaN
4	106.0	Level3	4.0	3.0

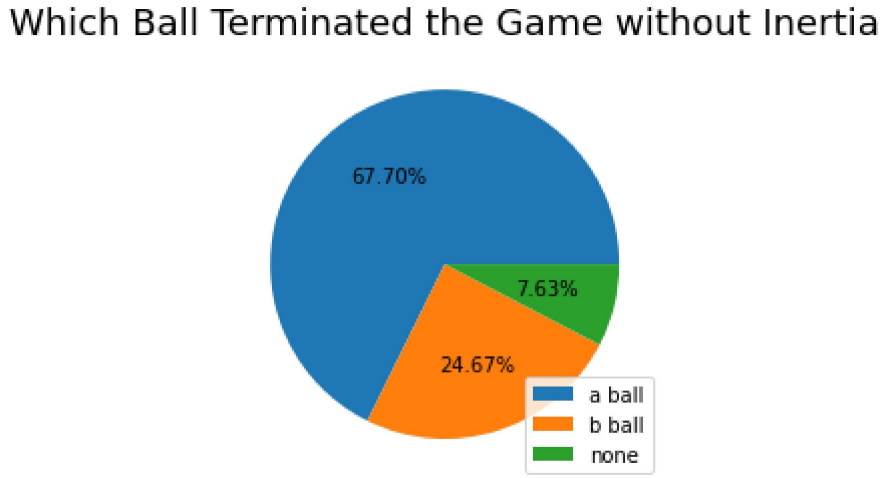
	Fall	Touch different Color	level	Touch Enemy
0	8	68.0	Level0	NaN
1	45	NaN	Level0.1	70.0
2	111	167.0	Level1	NaN
3	2244	14.0	Level2	39.0
4	189	17.0	Level3	42.0


Reasons of Termination with Inertia
Level3
Level2
Level1
Level0.1
Level0
level
Fall
Touch different Color
Touch Enemy
0
20
40
60
80
100

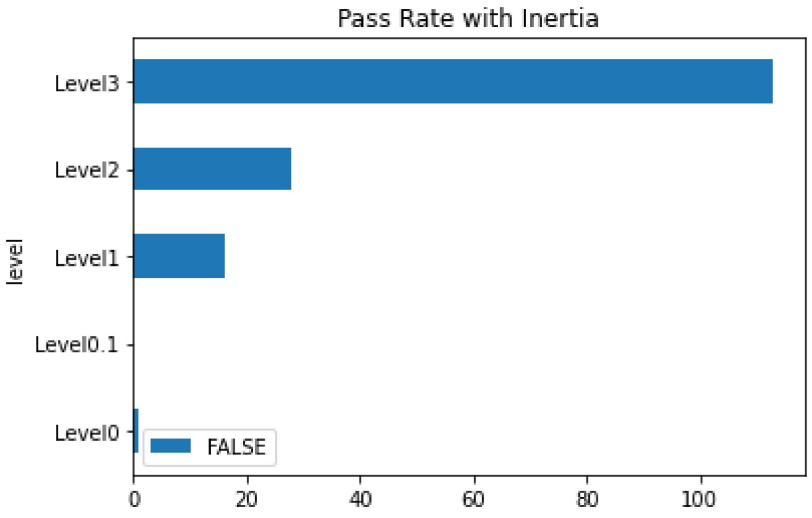

Reasons of Termination without Inertia
Level3
Level2
Level1
Level0.1
Level0
level
Fall
Touch different Color
Touch Enemy
0
500
1000
1500
2000

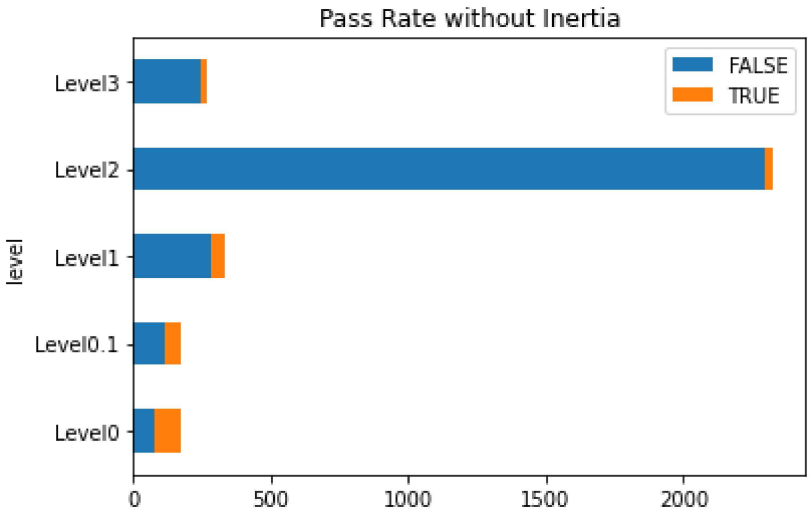
In [7]:
```
for i in range(2):
    whoCausedDeath = df1['who caused death, 0 for left player, 1 for right, -
    df6=df1
    df6["who caused death"] = whoCausedDeath
    df6=df6[df6['inertia world for true, normal for false']==truefalse[i]]
    df5 = df6.groupby(['who caused death']).size()
    plt.title("Which Ball Terminated the Game "+ withinertia[i],fontsize=18)
    patches, text1, text2 = plt.pie(df5, autopct=lambda p: '{:.2f}%'.format(p
    plt.legend(patches, df5.index, loc="lower right")
    plt.show()
```



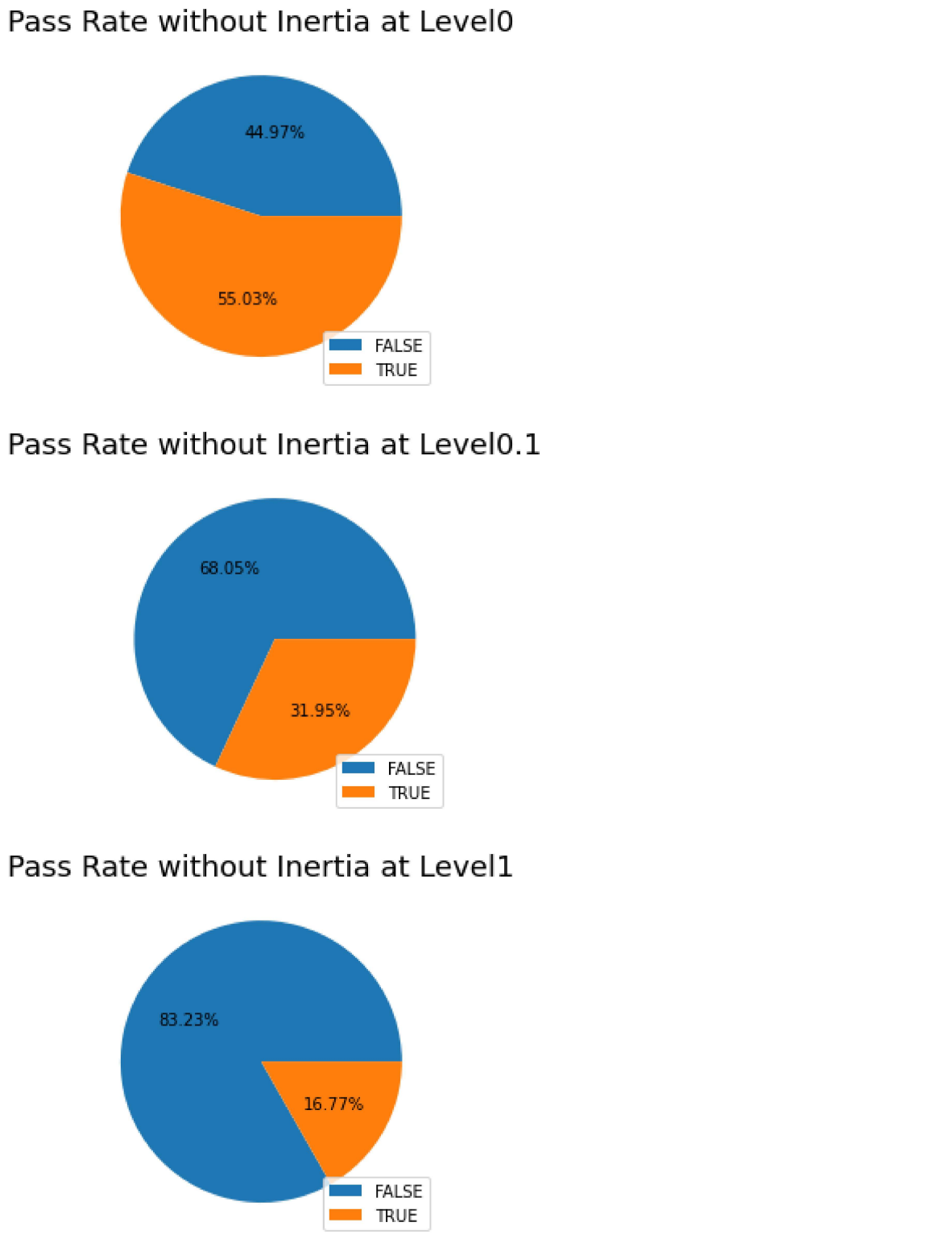


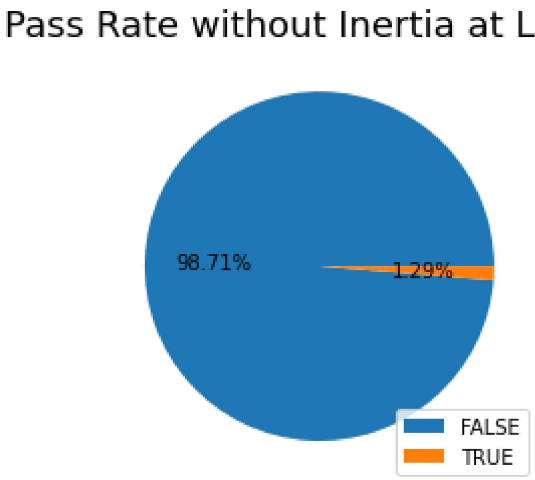
In [8]:
```
for i in range(2):
    inertia=df1[df1['inertia world for true, normal for false']==truefalse[i]
    data=[]
    for j in (['Level0', 'Level0.1', 'Level1', 'Level2', 'Level3']):
        level=inertia[inertia['level']==j]
        df2 = level.groupby(['true for pass level, false for died']).size()
        dictionary=df2.to_dict()
        dictionary['level']=j
        data.append(dictionary)

    passRate = pd.DataFrame(data)
    passRate.plot(
        x = 'level',
        kind = 'barh',
        stacked = True,
        title = 'Pass Rate ' + withinertia[i], #stacked bar chart
        mark_right = True)
```

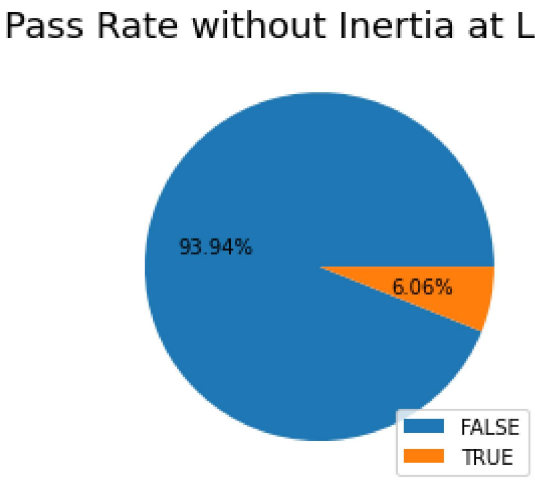



Pass Rate without Inertia
FALSE
TRUE
Level3
Level2
Level1
Level0.1
Level0
level
0
500
1000
1500
2000

In [9]:
```
inertia=df1[df1['inertia world for true, normal for false']=='FALSE']
for j in (['Level0', 'Level0.1', 'Level1', 'Level2', 'Level3']):
    level=inertia[inertia['level']==j]
    df2 = level.groupby(['true for pass level, false for died']).size()
    plt.title("Pass Rate without Inertia at " + j,fontsize=18)
    patches, text1, text2 = plt.pie(df2, autopct=lambda p: '{:.2f}%'.format(p
    plt.legend(patches, df2.index, loc="lower right")
    plt.show()
```




Pass Rate without Inertia at Level2
98.71%
1.29%
FALSE
TRUE


Pass Rate without Inertia at Level3
93.94%
6.06%
FALSE
TRUE

In [10]:
```
levelsPlayed = pd.DataFrame(df1.groupby('sessionID')['level'].apply(lambda x:
levelsPlayed.reset_index(inplace=True)
for i in ['Level0', 'Level0.1', 'Level1', 'Level2', 'Level3']:
    levelsPlayed[i]=levelsPlayed['level'].apply(lambda x: 1 if i in x else 0)
levelsCnt=pd.DataFrame(levelsPlayed.groupby(['Level0.1', 'Level0', 'Level1',
levelsCnt.reset_index(inplace=True)
levelsCnt = levelsCnt.rename(columns = {0:'count'})
levelsCnt=levelsCnt.sort_values(by='count', ascending=False)
levelsCnt
```

Out[10]:

	Level0.1	Level0	Level1	Level2	Level3	count
0	0	0	0	0	0	374
4	0	0	1	0	0	77
8	0	1	0	0	0	71
1	0	0	0	0	1	37
16	1	1	0	0	0	32
14	1	0	0	0	0	29
2	0	0	0	1	0	26
23	1	1	1	1	1	22
20	1	1	1	0	0	14
22	1	1	1	1	0	12
5	0	0	1	0	1	11
11	0	1	1	0	0	10
6	0	0	1	1	0	8
21	1	1	1	0	1	5
7	0	0	1	1	1	4
17	1	1	0	0	1	4
13	0	1	1	1	1	3
9	0	1	0	0	1	2
10	0	1	0	1	1	2
3	0	0	0	1	1	2
18	1	1	0	1	0	2
19	1	1	0	1	1	2
12	0	1	1	1	0	2
15	1	0	1	0	0	1

In [11]:
```
```

In [24]:

```
```

Out[24]:

```
int
```