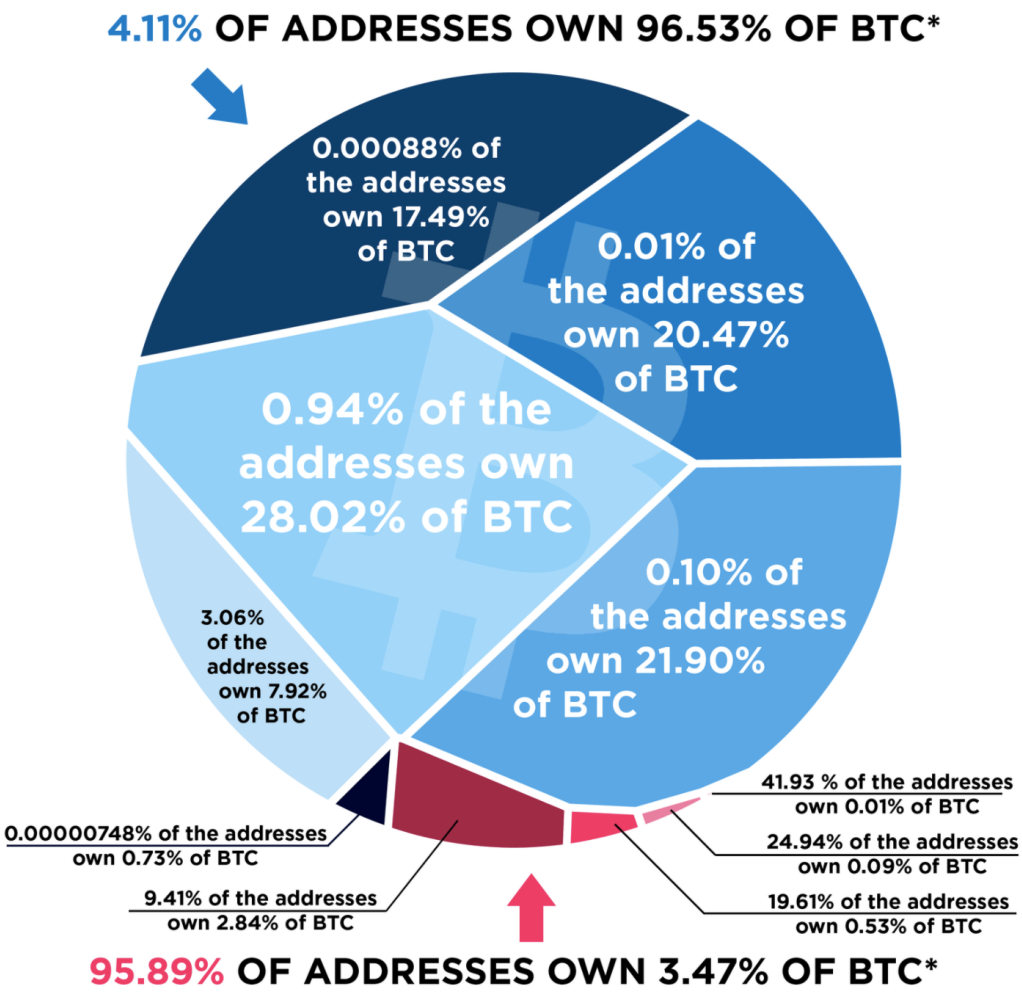
DATA ADVANCED PE

Bavo Knaeps & Jitse Wierdsma 2TINJ

Voorbeelden slechte data representatie

Voorbeeld 1 (cirkeldiagram)

Een eerste slecht voorbeeld van data representatie, is volgende “cirkeldiagram”.
Het is goed dat er geprobeerd werd een creatieve diagram te maken, maar de integriteit van de data is niet meer in tact gehouden.
Het doel van de diagram, is om de centralisatie van het bitcoin kapitaal aan te tonen.
De manier waarop de delen van de cirkel opgesplitst zijn is te hectisch.
Hierdoor is het onduidelijk wat er precies aan de hand is, aangezien het oppervlakte van elk deel moeilijk te vergelijken is met de anderen.
Daarnaast was een cirkeldiagram mogelijk zelfs niet de beste keuze voor deze data te representeren.
Een liggende staafdiagram, bijvoorbeeld, zou het makkelijker maken om op het eerste oog duidelijk de impact te zien van de centralisatie van het bitcoin kapitaal.

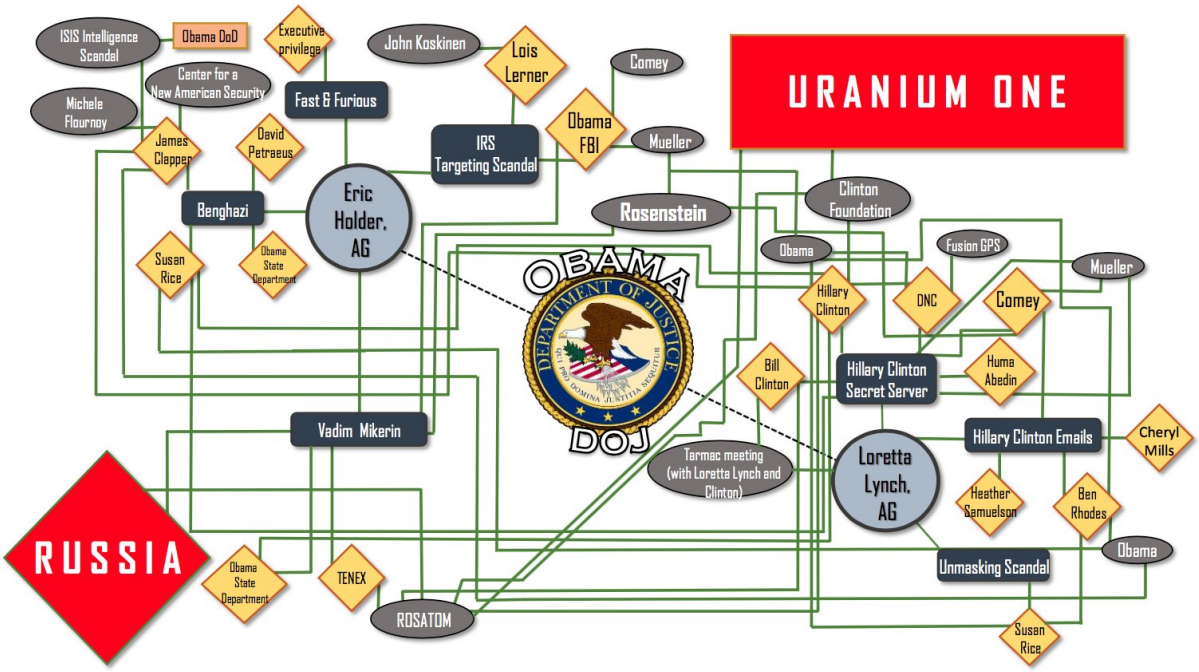


Voorbeeld 2 (flowchart)

Een tweede slecht voorbeeld, is deze flowchart. Het valt al snel op wat hier het probleem is.
Deze chart werd gebruikt in een hoorzitting in de “United States House Committee on the Judiciary”.
Het is niet zo makkelijk om te zien, maar als je goed kijkt, lijkt het alsof de cirkel van Obama verbonden is met een andere cirkel van Obama. Dit brengt ons tot de eerste grote fout aan deze flowchart: de lijnen zijn bijna onmogelijk om te volgen. Het is namelijk zeer moeilijk om te identificeren waar lijnen naartoe gaan als ze elkaar kruisen.

Daarnaast is het ook niet duidelijk waarom Uranium One en Russia er zo speciaal uit zien. Er trekken zelfs meer lijnen naar/uit andere (blijkend) minder significante stukken.

Op het eerste zicht kan je dus bijna niks halen uit deze flowchart.



Gebruikte pluralsight courses

Introduction to Data Visualization with Python
by YK Sugi
https://app.pluralsight.com/library/courses/data-visualization-with-python-introduction/table-of-contents (https://app.pluralsight.com/library/courses/data-visualization-with-python-introduction/table-of-contents)
Gebruikt voor informatie over Jupyter, Pandas en matplotlib, waarmee tal van grafieken gemaakt kunnen worden.
Een duidelijke cursus die de nodige uitleg geeft.

Importing Data from Microsoft Excel Files with Python
by Gabriel Cánepa
https://app.pluralsight.com/guides/importing-data-from-excel-with-python (https://app.pluralsight.com/guides/importing-data-from-excel-with-python)
Gebruikt voor informatie over openpyxl en hoe hiermee data gelezen kan worden uit Excel files, alsook hoe data of grafieken weggeschreven kunnen worden naar een Excel file.
Een korte cursus, met overzichtelijke code voorbeelden.

Coding

(functies en klassen staan op het einde van het document volledig uitgeschreven)

In [1]:
```
import openpyxl
import matplotlib
from Voetbal.Graphs import visual as visual
from Voetbal.Spelertjes import Spelertjes
```

Import openpxl om data in te lezen en data weg te schrijven naar xlsx file.
Import methodes uit andere python files om later te gebruiken.

In [2]:
```
spelertjes = Spelertjes()
graphs = visual()
```

Maak objecten aan met de geïmporteerde objecten.

In [3]:
```
fileName = 'voetbal.xlsx'
sheetName = 'gegevens'
saveFileName = 'correct.xlsx'

spelertjes.readFile(fileName, sheetName)
spelertjes.writeFile(fileName, sheetName, saveFileName)

fileName = saveFileName
sheetName = "grafiek"
```

Lees de excel file "voetbal.xlsx" in en ga naar tab 'gegevens'. Maak deze file inorde (dubbele kolom verwijderen) en vul de data. Save daarna de file met de naam 'correct.xlsx'.
De filename gaat daarna op de saveFileName gezet worden (zodat we het correcte bestand gebruiken met de juiste data op de juiste plaats). We veranderen ook de sheetname voordat we de methodes uitvoeren.
De grafieken zijn nu ook tezien in de excel file (in een iets minder goede vorm).

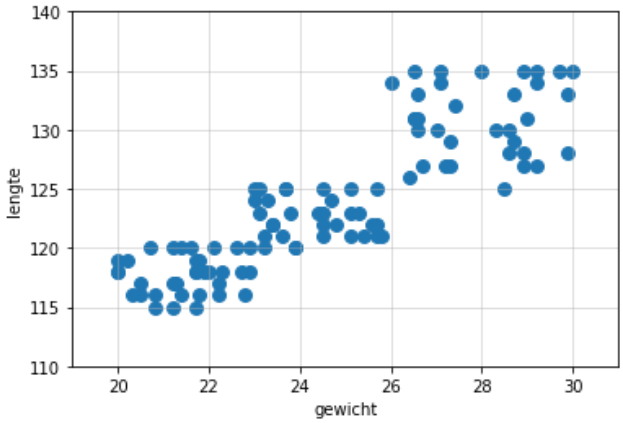
In [4]:
```
def writeFile(self, fileName, sheetName, saveFileName):
    wb = load_workbook(fileName)
    ws = wb[sheetName]

    font = Font(name='Calibri',size=12,bold=True)
    header = ["naam", "positie", "aantal gemaakte goalen", "geboortecategori
e", "inzet", "gewicht", "lengte", "geboortedatum"]

    # fill the headers with the right values
    for i in range(len(header)):
        cellref = ws.cell(1, i + 1)
        cellref.value = header[i]
        cellref.font = font

        # clear last column
        cellref = ws.cell(1, i + 2)
        cellref.value = None

        # fill data on tab 'gegevens' with new generated values
        for i in range(len(self.spelertjesValues)):
            for j in range(len(self.spelertjesValues[i])):
                cellref = ws.cell(row = i + 2, column = j + 1)
                cellref.value = self.spelertjesValues[i][j]
                # clear last column
                cellref = ws.cell(row = i + 2, column = j + 2)
                cellref.value = None

        wb.save(saveFileName)

    # code used in other class to fill spelertjes (commented so it doesn't d
o anything in the notebook)
    # for row in ws.rows:
        #     args = [cell.value for cell in row]
        #     birthDate = self.generateBirth()
        #     spelertje = Spelertje(args[0], args[2], args[3], self.returnCa
t(birthDate),self.effort[self.returnCat(birthDate)], args[6],args[7], birthD
ate)
        #     self.spelertjesValues.append(spelertje.returnArray())
        #     self.addSpeler(spelertje)
```

Hoe data genereren?

Om de date te vullen gebruiken we een methode generateBirth. Hier gaat er een random datum gegenereerd worden tussen 01-01-2011 en, 31-12-2011. We gebruiken dan deze datum om de categorie toe te wijzen aan het spelertje. Dit wordt gedaan met de methode returnCat. De volledige Code kan je op het einde terugvinden.

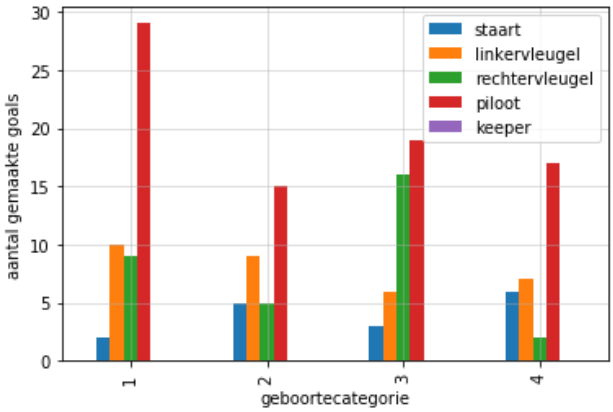
Teken de scatter chart.

In [5]:
```
graphs.drawScatterChart(fileName, sheetName, fileName)
```



Teken de staafdiagram.

In [6]:
```
graphs.drawBarChart(fileName,sheetName,fileName)
```



Bespreking staafdiagram

Per geboortecategorie, wordt er nog een opdeling gemaakt volgens positie (staart, linkervleugel, ...).
Aan de hand van de beschikbare data, wordt vervolgens voor elke positie in elke geboortecategorie het aantal goals getekend met een staaf. Via de y-as is dan een schatting van het aantal goals zienbaar.
Uit deze gegevens blijkt dat piloten bijna altijd de meeste goals scoren, op een gedeelde tweede plaats de linker- en rechtervleugel, op de derde plaats de staart en tot slot als laatste de keeper, die in deze dataset geen goals gescoord heeft.

Bereken gemiddelde en modus van aantal goals

In [7]:
```
graphs.averageAndModus(fileName)
```

```
Modus:
---------------

staart :0
linkervleugel :1
rechtervleugel :1
piloot :4
keeper :0

Average Goals:
---------------

staart : 0.8
linkervleugel : 1.6
rechtervleugel : 1.6
piloot : 4.0
keeper : 0.0
```

Bespreking modus en gemiddelde

Aan de hand van deze gegevens kunnen we zien dat het meeste voorkomende aantal gemaakte goals bij de keepers en bij de staart 0 is, bij de linker en de rechtervleugel 1 is en bij de piloot 4.
Het gemmiddelde van het aantal gemaakte goals per positie is voor de staart 0.8, voor de linker-en rechtervleugel 1.6, voor de keepers 0 en voor de piloot 4. Zo kunnen we vaststellen dat de piloot het belangrijkste is bij het aantal gemaakte goals.

Bereken kwartiel 1 en de standaardafwijking van kolom G (het gewicht)

In [8]:
```
graphs.calculateQuartileAndStd(fileName)
```

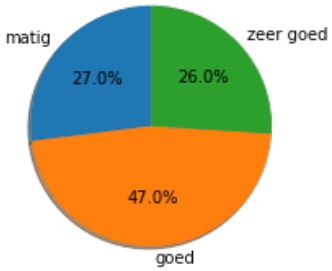
```
Kwartiel 1: 22.075000000000003
Standaard afwijking: 2.8691211197856394
```

Lijkt er een verband te zijn tussen positie op het veld en het aantal goals scoren?

Blijkend uit de staafdiagram en het gemiddelde aantal goals, is er wel degelijk een verband tussen de positie op het veld en het aantal goals dat gescoord wordt. Zo is het op eerste zicht duidelijk op de staafdiagram dat een speler met positie 'piloot' bijna altijd het meeste goals scoort, terwijl een speler met positie 'keeper' bijna nooit een goal scoort (of zelfs helemaal niet in deze dataset).

Teken de cirkeldiagram.

In [9]:
```
graphs.drawPieChart(fileName)
```

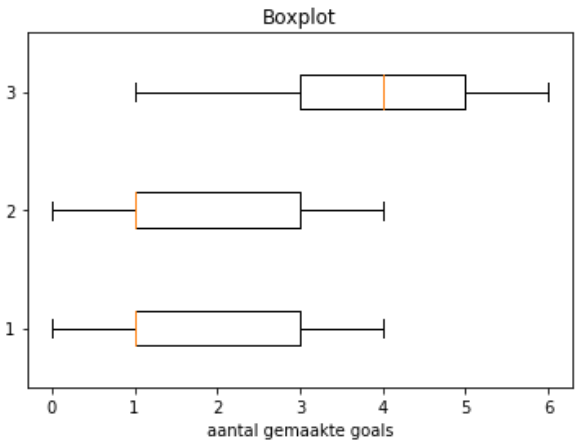


Bespreking cirkeldiagram

De gegevens gebruikt door de cirkeldiagram worden steeds willekeurig gegenereerd. Hierdoor is het moeilijk om een duidelijke beschrijving te geven. Wat wel op valt: De categorie 'goed' zit meestal rond de 50%, en de categoriën 'matig' en 'zeer goed' schommelen fel rond de 25%. Dit komt doordat categorie 2 en 3 een 'goede' inzet toegewezen krijgen, cat. 1 'zeer goed' en cat. 2 'matig'. Dus door het genereren van de geboortedatums is er 50% kans dat deze valt in categorie 2 of 3, en 50% kans dat ze valt in categorie 1 of 4.

Teken boxplot

In [10]:
```
graphs.drawBoxPlot(fileName)
```



Bespreking boxplot

Plot nummer 3 verwijst naar de piloten. We zien een minimum van 1 goal, een eerste kwartiel (Q1) van 3 goals, een mediaan van 4 goals, een derde kwartiel (Q3) van 5 goals, en een maximum van 6 goals. Plot nummers 2 en 1 verwijzen naar de rechter- en linkervleugel respectievelijk. Opmerkelijk is dat de mediaan bij beide samenvalt met Q1. Dit wilt zeggen dat de middelste waarde van de eerste helft van de data gelijk is aan de mediaan, wat 'het midden' is van de volledige dataset. Voor de rest zien we hier een minimum van 0 goals en een maximum van 4 goals, alsook de Q1 en mediaan van 1 goal, en een Q3 van 3 goals.

De piloten scoren dus meestal meer als linker- of rechtervleugel spelers.

Welk soort gegeven (van de 4 besproken in de cursus) is 'aantal gemaakte goalen', 'inzet' en 'gewicht'

'aantal gemaakte goalen' is een kwantitatief gegeven, meer specifiek een discreet gegeven.
'inzet' is een kwalitatief gegeven, meer specifiek een ordinaal gegeven.
'gewicht' is een kwantitatief gegeven, meer specifiek een continu gegeven.

Alle gebruikte code

In [11]:
```
from openpyxl import load_workbook
from openpyxl.chart.series import Series
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from Voetbal.Spelertjes import Spelertjes
from openpyxl.chart import (ScatterChart, Reference, Series, marker, BarChar
t)

class visual:
    def drawScatterChart(self, fileName, sheetName, saveFileName = None):
        if saveFileName is None:
            saveFileName = fileName

        wb = load_workbook(fileName)
        ws = wb['gegevens']

        chart = ScatterChart()
        chart.title = "Scatter Chart"
        chart.style = 13
        chart.x_axis.scaling.min = 19
        chart.x_axis.scaling.max = 31
        chart.y_axis.scaling.min = 110
        chart.y_axis.scaling.max = 140
        chart.x_axis.title = 'gewicht'
        chart.y_axis.title = 'lengte'
        chart.legend = None

        xvalues = Reference(ws, min_col=6, min_row=2, max_row=101)
        values = Reference(ws, min_col=7, min_row=2, max_row=101)
        #fill x and y, skip first
        x=[]
        y=[]
        iterrows = iter(ws.rows)
        next(iterrows)
        for row in iterrows:
            x.append(row[5].value)
            y.append(row[6].value)

        series = Series(values, xvalues)
        series.graphicalProperties.line.noFill = True
        series.marker = marker.Marker('circle', 5.2)
        chart.series.append(series)

        # Style the lines
        s1 = chart.series[0]
        s1.marker.symbol = "circle"
        s1.marker.graphicalProperties.solidFill = "4076A9"  # Marker filling
        s1.marker.graphicalProperties.line.solidFill = "4076A9"  # Marker ou
tline
        s1.graphicalProperties.line.noFill = True

        ws = wb[sheetName]
        ws.add_chart(chart, "L7")
        wb.save(saveFileName)

        area = np.pi * 20
        plt.scatter(x, y, s=area, alpha=1)
        plt.xlabel("gewicht")
        plt.ylabel("lengte")
        plt.grid(True,alpha=0.5)
        plt.axis([19,31,110,140])
        plt.show()
```

In [12]:

```
    def drawBarChart(self, fileName, sheetName, saveFileName = None):
        spelertjes = Spelertjes()
        if saveFileName is None:
            saveFileName = fileName
        #read all the data using openpyxl and write data to grafiek tab
        wb = load_workbook(fileName)
        ws = wb['gegevens']


        goals = {"staart":{1:0,2:0,3:0,4:0}, "linkervleugel":{1:0,2:0,3:0,4:
0},"rechtervleugel":{1:0,2:0,3:0,4:0},
                 "piloot":{1:0,2:0,3:0,4:0},"keeper":{1:0,2:0,3:0,4:0}}

        positions = ["staart","linkervleuger","rechtervleuger","piloot","kee
per"]

        iterrows = iter(ws.rows)
        next(iterrows)

        for row in iterrows:
            position = goals[row[1].value]
            position[row[3].value] += row[2].value
            goals[row[1].value] = position
        ws = wb['grafiek']

        for i in range(2,6):
            cellref = ws.cell(1, i)
            cellref.value = i - 1

        for i in range(2,7):
            cellref = ws.cell(i, 1)
            cellref.value = positions[i-2]
        row = 2

        for i in goals.values():
            column = 2
            for j in i.values():
                cellref = ws.cell(row, column)
                cellref.value = j
                column += 1
            row += 1

        chart1 = BarChart()
        chart1.type = "col"
        chart1.style = 10
        chart1.title = "goals per position per birth cat"
        chart1.y_axis.title = 'goals'
        chart1.x_axis.title = 'position'

        data = Reference(ws, min_col=2, min_row=1, max_row=6, max_col=5)
        cats = Reference(ws, min_col=1, min_row=1 , max_row=6)
        chart1.add_data(data,titles_from_data=True)
        chart1.set_categories(cats)
        chart1.shape = 4
        ws.add_chart(chart1, "C24")
        wb.save(saveFileName)

        pd.DataFrame(goals).plot(kind='bar')
        plt.xlabel("geboortecategorie")
        plt.ylabel("aantal gemaakte goals")
        plt.grid(True, alpha=0.5)
        plt.show()
```

In [13]:

```
    def averageAndModus(self, fileName,):
        # read all the data using openpyxl and write data to grafiek tab
        wb = load_workbook(fileName)
        ws = wb['gegevens']

        goals = {"staart": 0, "linkervleugel": 0,
                 "rechtervleugel": 0,
                 "piloot": 0, "keeper": 0}

        # count amount of players in cat | is always 20 but just in case it
changes...
        goalsCounter = {"staart": 0, "linkervleugel": 0,
                 "rechtervleugel": 0,
                 "piloot": 0, "keeper": 0}

        averageGoals = {"staart": 0, "linkervleugel": 0,
                 "rechtervleugel": 0,
                 "piloot": 0, "keeper": 0}

        modus = {"staart":{0:0,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0}, "linkervleu
gel":{0:0,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0},
                 "rechtervleugel":{0:0,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0},
                 "piloot":{0:0,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0},"keeper":
                     {0:0,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0}}

        iterrows = iter(ws.rows)
        next(iterrows)
        for row in iterrows:
            temp = modus[row[1].value]
            temp[row[2].value] += 1
            modus[row[1].value] = temp
            goalsCounter[row[1].value] += 1
            goals[row[1].value] += row[2].value


        for i in goals:
            averageGoals[i] = goals[i]/goalsCounter[i]
        #still need to write this to excel file
        print("Modus: \n---------------\n")
        for pos in modus:
            print(str(pos)+ " :"+ str(max(modus[pos], key=modus[pos].get)))
        print()
        print("Average Goals: \n---------------\n")
        for goal in averageGoals:
            print(str(goal) + " : " + str(averageGoals[goal]))

    def calculateQuartileAndStd(self,fileName):
        # read all the data using openpyxl and write data to grafiek tab
        wb = load_workbook(fileName)
        ws = wb['gegevens']

        data = []
        #skip first row
        iterrows = iter(ws.rows)
        next(iterrows)
        for row in iterrows:
            data.append(row[5].value)
        #calculate std and quartile 1
        print("Kwartiel 1 : " + str(np.percentile(data,25)))
        print("standaard afwijking : " + str(np.std(data)))
```

In [14]:

```
    def drawBoxPlot(self,fileName):
        wb = load_workbook(fileName)
        ws = wb['gegevens']

        data = {"linkervleugel":[],"rechtervleugel":[],
                "piloot":[]}
        iterrows = iter(ws.rows)
        next(iterrows)
        for row in iterrows:
            pos = row[1].value
            if pos == "piloot" or pos == "linkervleugel" or pos == "rechterv
leugel":
                data[row[1].value].append(row[2].value)

        plt.boxplot(data.values(), 0, 'rs', 0)
        plt.title("Boxplot")
        plt.xlabel("aantal gemaakte goals")
        plt.show()

    def drawPieChart(self, fileName):
        wb = load_workbook(fileName)
        ws = wb['gegevens']

        data = {"matig": 0, "goed": 0, "zeer goed": 0}
        iterrows = iter(ws.rows)
        next(iterrows)

        total = 0

        for row in iterrows:
            inzet = row[4].value

            # Check for possible incorrect values
            if inzet == "matig" \
                    or inzet == "goed" \
                    or inzet == "zeer goed":
                data[inzet] += 1
                total += 1

        fig, ax1 = plt.subplots(figsize=(6, 3), subplot_kw=dict(aspect="equa
l"))

        ax1.pie(data.values(), explode=(0, 0, 0), labels=data.keys(), autopc
t='%1.1f%%',
                shadow=True, startangle=90)
        ax1.axis('equal')

        plt.show()
```

In [16]:
```
import random
import datetime
from openpyxl import load_workbook
from openpyxl.styles import Font


class Spelertje:
    def __init__(self, name, position, goals, birthCat, effort, weight, leng
th,birthDate):
        self.name = name
        self.position= position
        self.goals = goals
        self.birthCat = birthCat
        self.effort = effort
        self.weight = weight
        self.length = length
        self.birthDate= birthDate

    def returnArray(self):
        return [self.name, self.position, self.goals, self.birthCat, self.ef
fort,
                self.weight, self.length,self.birthDate]

    def __str__(self):
        return str(self.name) + " - " + str(self.position) + " - " + str(sel
f.goals) + " - " + \
    str(self.birthCat) + " - " + str(self.effort) + " - " + str(self.weight)
+ " - " + \
    str(self.length + "-" + str(self.birthDate))

class Spelertjes:
    def __init__(self):
        self.spelertjes = []
        self.effort = {1:"zeer goed", 2 : "goed", 3:"goed",4:"matig"}
        self.spelertjesValues = []

    def addSpeler(self, spelertje):
        self.spelertjes.append(spelertje)

    def returnCat(self, date):
        day =  date.timetuple().tm_yday
        if day <= 90:
            return 1
        if day <= 181:
            return 2
        if day <= 273:
            return 3
        if day <= 365:
            return 4

    def generateBirth(self):
        startdate = datetime.date(2011, 1, 1)
        date = startdate + datetime.timedelta(random.randint(1,365))
        return date
```

In [17]:

```
    def readFile(self, fileName, sheetName):
        wb = load_workbook(fileName)
        ws = wb[sheetName]

        for row in ws.rows:
            args = [cell.value for cell in row]
            birthDate = self.generateBirth()
            spelertje = Spelertje(args[0], args[2], args[3], self.returnCat(
birthDate),self.effort[self.returnCat(birthDate)], args[6],args[7], birthDat
e)
            self.spelertjesValues.append(spelertje.returnArray())
            self.addSpeler(spelertje)
        #remove first value
        self.spelertjes.remove(self.spelertjes[0])
        self.spelertjesValues.remove(self.spelertjesValues[0])

    def writeFile(self, fileName, sheetName, saveFileName):
        wb = load_workbook(fileName)
        ws = wb[sheetName]

        font = Font(name='Calibri',size=12,bold=True)
        header = ["naam", "positie", "aantal gemaakte goalen", "geboortecate
gorie", "inzet", "gewicht", "lengte","geboortedatum"]

        #fill the headers with the right values
        for i in range(len(header)):
            cellref = ws.cell(1,i+1)
            cellref.value = header[i]
            cellref.font = font
            #clear last column
            cellref = ws.cell(1,i+2)
            cellref.value = None

        #fill gegevens with new generated values
        for i in range(len(self.spelertjesValues)):
            for j in range(len(self.spelertjesValues[i])):
                cellref = ws.cell(row=i+2, column=j+1)
                cellref.value = self.spelertjesValues[i][j]
                #clear last column
                cellref = ws.cell(row=i + 2, column=j + 2)
                cellref.value = None

        wb.save(saveFileName)
```