PDSP 2026, Lecture 02, 6 August 2025

Set up the table as a list

- Each entry is a row
- Each row is a tuple of columns
- `(City,HTeam,ATeam,HGoals,AGoals,HPens,APens,HXG,AXG)`
- `HGoals`, `AGoals`, `HPens`, `APens` are integers (`Int`)
- `HXG`, `AXG` are decimal fractions (`Float`)
- `City`, `HTeam`, `ATeam` are text (`String`)
- Assign a value to a variable: `variable = value`

In [1]:
```
matchlist = [
  ("Mexico City","MEX","RSA",2,0,0,0,1.84,0.52),
  ("Zapopan","KOR","CZE",2,1,0,0,1.45,1.12),
  ("Toronto","CAN","BIH",1,1,0,0,1.35,0.98),
  ("Inglewood","USA","PAR",4,1,0,0,2.76,0.88),
  ("Foxborough","QAT","SUI",1,1,0,0,0.78,1.54),
  ("East Rutherford","BRA","MAR",1,1,0,0,1.62,1.1),
  ("Philadelphia","HAI","SCO",0,1,0,0,1.25,0.65),
  ("Seattle","AUS","TUR",2,0,0,0,1.48,0.72),
  ("Atlanta","GER","CUW",7,1,0,0,4.82,0.44),
  ("Santa Clara","NED","JPN",2,2,0,0,1.95,1.82),
  ("Houston","CIV","ECU",1,0,0,0,1.1,0.95),
  ("Kansas City","SWE","TUN",5,1,0,0,3.12,0.78),
  ("Miami Gardens","ESP","CPV",0,0,0,0,2.15,0.35),
  ("Arlington","BEL","EGY",1,1,0,0,1.58,1.12),
  ("Guadalupe","KSA","URU",1,1,0,0,0.85,1.76),
  ("Vancouver","IRN","NZL",2,2,0,0,1.88,1.2),
  ("East Rutherford","FRA","SEN",3,1,0,0,2.44,1.05),
  ("Foxborough","IRQ","NOR",1,4,0,0,0.68,2.85),
  ("Inglewood","ARG","ALG",3,0,0,0,2.65,0.42),
  ("Santa Clara","AUT","JOR",3,1,0,0,1.98,0.88),
  ("Atlanta","POR","COD",1,1,0,0,1.78,1.15),
  ("Miami Gardens","ENG","CRO",4,2,0,0,2.92,1.64),
  ("Houston","GHA","PAN",1,0,0,0,1.3,0.75),
  ("Seattle","UZB","COL",1,3,0,0,0.95,2.1),
  ("Atlanta","MEX","KOR",1,0,0,0,1.62,0.58),
  ("Inglewood","CZE","RSA",1,1,0,0,1.35,1.15),
  ("Vancouver","CAN","QAT",6,0,0,0,3.82,0.22),
  ("Mexico City","SUI","BIH",4,1,0,0,2.45,0.88),
  ("East Rutherford","BRA","HAI",3,0,0,0,1.16,0.9),
  ("Arlington","SCO","MAR",0,1,0,0,0.97,0.54),
  ("Toronto","USA","AUS",2,0,0,0,1.21,0.32),
  ("Guadalupe","TUR","PAR",0,1,0,0,0.4,0.3),
  ("Foxborough","GER","CIV",2,1,0,0,1.85,0.72),
  ("Kansas City","ECU","CUW",0,0,0,0,1.92,0.08),
  ("Philadelphia","NED","SWE",5,1,0,0,3.45,0.88),
  ("Seattle","TUN","JPN",0,4,0,0,0.35,3.12),
  ("Houston","BEL","IRN",0,0,0,0,1.45,0.62),
  ("Miami Gardens","NZL","EGY",1,3,0,0,0.55,2.15),
  ("Santa Clara","ESP","KSA",4,0,0,0,3.28,0.25),
  ("East Rutherford","URU","CPV",2,2,0,0,1.65,1.1),
```

```
("Arlington","FRA","IRQ",3,0,0,0,2.21,0.45),
("Toronto","NOR","SEN",3,2,0,0,1.95,1.48),
("Guadalupe","ARG","AUT",2,0,0,0,1.78,0.55),
("Foxborough","JOR","ALG",1,2,0,0,0.88,1.92),
("Kansas City","POR","UZB",5,0,0,0,3.54,0.42),
("Philadelphia","COL","COD",1,0,0,0,1.62,0.58),
("Seattle","ENG","GHA",0,0,0,0,1.12,0.89),
("Mexico City","PAN","CRO",0,1,0,0,0.42,1.48),
("Mexico City","CZE","MEX",0,3,0,0,1.1,1.27),
("Zapopan","RSA","KOR",1,0,0,0,1.26,1.39),
("Vancouver","SUI","CAN",2,1,0,0,0.71,1.35),
("Toronto","BIH","QAT",3,1,0,0,1.48,0.95),
("East Rutherford","SCO","BRA",0,3,0,0,0.45,2.18),
("Foxborough","MAR","HAI",4,2,0,0,2.35,1.12),
("Inglewood","TUR","USA",3,2,0,0,1.62,1.45),
("Santa Clara","PAR","AUS",0,0,0,0,0.55,0.48),
("Atlanta","ECU","GER",2,1,0,0,1.28,1.55),
("Houston","CUW","CIV",0,2,0,0,0.32,1.68),
("Kansas City","TUN","NED",1,3,0,0,0.62,2.15),
("Seattle","JPN","SWE",1,1,0,0,1.12,0.95),
("Arlington","NZL","BEL",1,5,0,0,0.48,3.25),
("Guadalupe","EGY","IRN",1,1,0,0,0.85,0.92),
("Miami Gardens","URU","ESP",0,1,0,0,0.78,1.35),
("Philadelphia","CPV","KSA",0,0,0,0,0.42,0.38),
("Foxborough","NOR","FRA",1,4,0,0,1.3,0.96),
("East Rutherford","SEN","IRQ",5,0,0,0,3.01,0.14),
("Santa Clara","JOR","ARG",1,3,0,0,0.76,2.13),
("Inglewood","ALG","AUT",3,3,0,0,1.62,1.44),
("Miami Gardens","COL","POR",0,0,0,0,1.62,0.73),
("Atlanta","COD","UZB",3,1,0,0,2.35,0.2),
("East Rutherford","PAN","ENG",0,2,0,0,0.69,1.39),
("Philadelphia","CRO","GHA",2,1,0,0,0.42,0.74),
("Inglewood","RSA","CAN",0,1,0,0,0.13,1.32),
("Houston","BRA","JPN",2,1,0,0,1.69,0.23),
("Foxborough","GER","PAR",1,1,3,4,0.72,0.42),
("Guadalupe","NED","MAR",1,1,2,3,0.23,1.4),
("Arlington","CIV","NOR",1,2,0,0,1.15,2.02),
("East Rutherford","FRA","SWE",3,0,0,0,3.17,0.67),
("Mexico City","MEX","ECU",2,0,0,0,1.02,0.73),
("Atlanta","ENG","COD",2,1,0,0,2.04,0.76),
("Seattle","BEL","SEN",3,2,0,0,1.74,3.58),
("Santa Clara","USA","BIH",2,0,0,0,0.92,0.25),
("Inglewood","ESP","AUT",3,0,0,0,2.84,0.32),
("Toronto","POR","CRO",2,1,0,0,2.18,1.34),
("Vancouver","SUI","ALG",2,0,0,0,2.52,0.73),
("Arlington","AUS","EGY",1,1,2,4,0.87,1.36),
("Miami Gardens","ARG","CPV",3,2,0,0,2.16,0.46),
("Kansas City","COL","GHA",1,0,0,0,2.19,0.26),
("Philadelphia","PAR","FRA",0,1,0,0,0.3,1.8),
("Houston","CAN","MAR",0,3,0,0,0.8,1.2),
("Arlington","BRA","NOR",1,2,0,0,2.61,1.05),
("Mexico City","MEX","ENG",2,3,0,0,1.88,1.61),
("Miami Gardens","POR","ESP",0,1,0,0,0.63,1.69),
("Santa Clara","USA","BEL",1,4,0,0,0.67,2.15),
("Kansas City","ARG","EGY",3,2,0,0,2.12,1.54),
("Seattle","SUI","COL",0,0,4,3,1.15,1.46),
("Foxborough","FRA","MAR",2,0,0,0,3.1,0.13),
("Inglewood","ESP","BEL",2,1,0,0,2.08,0.37),
("Miami Gardens","NOR","ENG",1,2,0,0,0.77,0.96),
("Kansas City","ARG","SUI",3,1,0,0,2.0,0.53),
```

```
  ("Arlington","FRA","ESP",0,2,0,0,0.3,1.63),
  ("Atlanta","ENG","ARG",1,2,0,0,0.54,1.80),
  ("Miami Gardens","FRA","ENG",4,6,0,0,2.88,2.88),
  ("East Rutherford","ESP","ARG",1,0,0,0,0.52,0.09)
]
```

- Can evaluate the value of a variable directly in a Jupyter notebook cell

In [2]:
```
matchlist
```

Out[2]:
```
[('Mexico City', 'MEX', 'RSA', 2, 0, 0, 0, 1.84, 0.52),
 ('Zapopan', 'KOR', 'CZE', 2, 1, 0, 0, 1.45, 1.12),
 ('Toronto', 'CAN', 'BIH', 1, 1, 0, 0, 1.35, 0.98),
 ('Inglewood', 'USA', 'PAR', 4, 1, 0, 0, 2.76, 0.88),
 ('Foxborough', 'QAT', 'SUI', 1, 1, 0, 0, 0.78, 1.54),
 ('East Rutherford', 'BRA', 'MAR', 1, 1, 0, 0, 1.62, 1.1),
 ('Philadelphia', 'HAI', 'SCO', 0, 1, 0, 0, 1.25, 0.65),
 ('Seattle', 'AUS', 'TUR', 2, 0, 0, 0, 1.48, 0.72),
 ('Atlanta', 'GER', 'CUW', 7, 1, 0, 0, 4.82, 0.44),
 ('Santa Clara', 'NED', 'JPN', 2, 2, 0, 0, 1.95, 1.82),
 ('Houston', 'CIV', 'ECU', 1, 0, 0, 0, 1.1, 0.95),
 ('Kansas City', 'SWE', 'TUN', 5, 1, 0, 0, 3.12, 0.78),
 ('Miami Gardens', 'ESP', 'CPV', 0, 0, 0, 0, 2.15, 0.35),
 ('Arlington', 'BEL', 'EGY', 1, 1, 0, 0, 1.58, 1.12),
 ('Guadalupe', 'KSA', 'URU', 1, 1, 0, 0, 0.85, 1.76),
 ('Vancouver', 'IRN', 'NZL', 2, 2, 0, 0, 1.88, 1.2),
 ('East Rutherford', 'FRA', 'SEN', 3, 1, 0, 0, 2.44, 1.05),
 ('Foxborough', 'IRQ', 'NOR', 1, 4, 0, 0, 0.68, 2.85),
 ('Inglewood', 'ARG', 'ALG', 3, 0, 0, 0, 2.65, 0.42),
 ('Santa Clara', 'AUT', 'JOR', 3, 1, 0, 0, 1.98, 0.88),
 ('Atlanta', 'POR', 'COD', 1, 1, 0, 0, 1.78, 1.15),
 ('Miami Gardens', 'ENG', 'CRO', 4, 2, 0, 0, 2.92, 1.64),
 ('Houston', 'GHA', 'PAN', 1, 0, 0, 0, 1.3, 0.75),
 ('Seattle', 'UZB', 'COL', 1, 3, 0, 0, 0.95, 2.1),
 ('Atlanta', 'MEX', 'KOR', 1, 0, 0, 0, 1.62, 0.58),
 ('Inglewood', 'CZE', 'RSA', 1, 1, 0, 0, 1.35, 1.15),
 ('Vancouver', 'CAN', 'QAT', 6, 0, 0, 0, 3.82, 0.22),
 ('Mexico City', 'SUI', 'BIH', 4, 1, 0, 0, 2.45, 0.88),
 ('East Rutherford', 'BRA', 'HAI', 3, 0, 0, 0, 1.16, 0.9),
 ('Arlington', 'SCO', 'MAR', 0, 1, 0, 0, 0.97, 0.54),
 ('Toronto', 'USA', 'AUS', 2, 0, 0, 0, 1.21, 0.32),
 ('Guadalupe', 'TUR', 'PAR', 0, 1, 0, 0, 0.4, 0.3),
 ('Foxborough', 'GER', 'CIV', 2, 1, 0, 0, 1.85, 0.72),
 ('Kansas City', 'ECU', 'CUW', 0, 0, 0, 0, 1.92, 0.08),
 ('Philadelphia', 'NED', 'SWE', 5, 1, 0, 0, 3.45, 0.88),
 ('Seattle', 'TUN', 'JPN', 0, 4, 0, 0, 0.35, 3.12),
 ('Houston', 'BEL', 'IRN', 0, 0, 0, 0, 1.45, 0.62),
 ('Miami Gardens', 'NZL', 'EGY', 1, 3, 0, 0, 0.55, 2.15),
 ('Santa Clara', 'ESP', 'KSA', 4, 0, 0, 0, 3.28, 0.25),
 ('East Rutherford', 'URU', 'CPV', 2, 2, 0, 0, 1.65, 1.1),
 ('Arlington', 'FRA', 'IRQ', 3, 0, 0, 0, 2.21, 0.45),
 ('Toronto', 'NOR', 'SEN', 3, 2, 0, 0, 1.95, 1.48),
 ('Guadalupe', 'ARG', 'AUT', 2, 0, 0, 0, 1.78, 0.55),
 ('Foxborough', 'JOR', 'ALG', 1, 2, 0, 0, 0.88, 1.92),
 ('Kansas City', 'POR', 'UZB', 5, 0, 0, 0, 3.54, 0.42),
 ('Philadelphia', 'COL', 'COD', 1, 0, 0, 0, 1.62, 0.58),
 ('Seattle', 'ENG', 'GHA', 0, 0, 0, 0, 1.12, 0.89),
 ('Mexico City', 'PAN', 'CRO', 0, 1, 0, 0, 0.42, 1.48),
 ('Mexico City', 'CZE', 'MEX', 0, 3, 0, 0, 1.1, 1.27),
 ('Zapopan', 'RSA', 'KOR', 1, 0, 0, 0, 1.26, 1.39),
 ('Vancouver', 'SUI', 'CAN', 2, 1, 0, 0, 0.71, 1.35),
 ('Toronto', 'BIH', 'QAT', 3, 1, 0, 0, 1.48, 0.95),
 ('East Rutherford', 'SCO', 'BRA', 0, 3, 0, 0, 0.45, 2.18),
 ('Foxborough', 'MAR', 'HAI', 4, 2, 0, 0, 2.35, 1.12),
 ('Inglewood', 'TUR', 'USA', 3, 2, 0, 0, 1.62, 1.45),
 ('Santa Clara', 'PAR', 'AUS', 0, 0, 0, 0, 0.55, 0.48),
 ('Atlanta', 'ECU', 'GER', 2, 1, 0, 0, 1.28, 1.55),
 ('Houston', 'CUW', 'CIV', 0, 2, 0, 0, 0.32, 1.68),
 ('Kansas City', 'TUN', 'NED', 1, 3, 0, 0, 0.62, 2.15),
 ('Seattle', 'JPN', 'SWE', 1, 1, 0, 0, 1.12, 0.95),
```

```
('Arlington', 'NZL', 'BEL', 1, 5, 0, 0, 0.48, 3.25),
('Guadalupe', 'EGY', 'IRN', 1, 1, 0, 0, 0.85, 0.92),
('Miami Gardens', 'URU', 'ESP', 0, 1, 0, 0, 0.78, 1.35),
('Philadelphia', 'CPV', 'KSA', 0, 0, 0, 0, 0.42, 0.38),
('Foxborough', 'NOR', 'FRA', 1, 4, 0, 0, 1.3, 0.96),
('East Rutherford', 'SEN', 'IRQ', 5, 0, 0, 0, 3.01, 0.14),
('Santa Clara', 'JOR', 'ARG', 1, 3, 0, 0, 0.76, 2.13),
('Inglewood', 'ALG', 'AUT', 3, 3, 0, 0, 1.62, 1.44),
('Miami Gardens', 'COL', 'POR', 0, 0, 0, 0, 1.62, 0.73),
('Atlanta', 'COD', 'UZB', 3, 1, 0, 0, 2.35, 0.2),
('East Rutherford', 'PAN', 'ENG', 0, 2, 0, 0, 0.69, 1.39),
('Philadelphia', 'CRO', 'GHA', 2, 1, 0, 0, 0.42, 0.74),
('Inglewood', 'RSA', 'CAN', 0, 1, 0, 0, 0.13, 1.32),
('Houston', 'BRA', 'JPN', 2, 1, 0, 0, 1.69, 0.23),
('Foxborough', 'GER', 'PAR', 1, 1, 3, 4, 0.72, 0.42),
('Guadalupe', 'NED', 'MAR', 1, 1, 2, 3, 0.23, 1.4),
('Arlington', 'CIV', 'NOR', 1, 2, 0, 0, 1.15, 2.02),
('East Rutherford', 'FRA', 'SWE', 3, 0, 0, 0, 3.17, 0.67),
('Mexico City', 'MEX', 'ECU', 2, 0, 0, 0, 1.02, 0.73),
('Atlanta', 'ENG', 'COD', 2, 1, 0, 0, 2.04, 0.76),
('Seattle', 'BEL', 'SEN', 3, 2, 0, 0, 1.74, 3.58),
('Santa Clara', 'USA', 'BIH', 2, 0, 0, 0, 0.92, 0.25),
('Inglewood', 'ESP', 'AUT', 3, 0, 0, 0, 2.84, 0.32),
('Toronto', 'POR', 'CRO', 2, 1, 0, 0, 2.18, 1.34),
('Vancouver', 'SUI', 'ALG', 2, 0, 0, 0, 2.52, 0.73),
('Arlington', 'AUS', 'EGY', 1, 1, 2, 4, 0.87, 1.36),
('Miami Gardens', 'ARG', 'CPV', 3, 2, 0, 0, 2.16, 0.46),
('Kansas City', 'COL', 'GHA', 1, 0, 0, 0, 2.19, 0.26),
('Philadelphia', 'PAR', 'FRA', 0, 1, 0, 0, 0.3, 1.8),
('Houston', 'CAN', 'MAR', 0, 3, 0, 0, 0.8, 1.2),
('Arlington', 'BRA', 'NOR', 1, 2, 0, 0, 2.61, 1.05),
('Mexico City', 'MEX', 'ENG', 2, 3, 0, 0, 1.88, 1.61),
('Miami Gardens', 'POR', 'ESP', 0, 1, 0, 0, 0.63, 1.69),
('Santa Clara', 'USA', 'BEL', 1, 4, 0, 0, 0.67, 2.15),
('Kansas City', 'ARG', 'EGY', 3, 2, 0, 0, 2.12, 1.54),
('Seattle', 'SUI', 'COL', 0, 0, 4, 3, 1.15, 1.46),
('Foxborough', 'FRA', 'MAR', 2, 0, 0, 0, 3.1, 0.13),
('Inglewood', 'ESP', 'BEL', 2, 1, 0, 0, 2.08, 0.37),
('Miami Gardens', 'NOR', 'ENG', 1, 2, 0, 0, 0.77, 0.96),
('Kansas City', 'ARG', 'SUI', 3, 1, 0, 0, 2.0, 0.53),
('Arlington', 'FRA', 'ESP', 0, 2, 0, 0, 0.3, 1.63),
('Atlanta', 'ENG', 'ARG', 1, 2, 0, 0, 0.54, 1.8),
('Miami Gardens', 'FRA', 'ENG', 4, 6, 0, 0, 2.88, 2.88),
('East Rutherford', 'ESP', 'ARG', 1, 0, 0, 0, 0.52, 0.09)]
```

- Can only check value of a name if it has bene assigned earlier

In [3]:
```
n
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 1
----> 1 n

NameError: name 'n' is not defined
```

- Any expression can be evaluated in a Jupyter notebook cell

In [4]:
```
7+3
```

Out[4]: 10

In [5]:
```
x = 9
y = x + 5
```

In [6]:
```
y
```

Out[6]: 14

Questions

- How many matches were played?
- What was the maximum number of goals scored in a match?
- What was the average number of goals scored in a match?
- How many matches had above average number of goals?
- How many cities were venues?
- Which team played as Home Team at maximum number of venues?
- Is being Home Team an advantage?

How many matches were played?

- Python's built-in function `len()` directly tells the length of a list

In [7]:
```
len(matchlist)
```

Out[7]: 104

Count the number of entries explicitly

- Iterate through the list
- `for` variable `in` listname `:`
  - `:` indicates a block of statements (commands) to follow
  - Indent the commands to be performed within each iteration
- `count = count + 1`
  - `=` assigns a new value to a variable, not to be confused with equality
  - rhs is old value of `count`, used to update the value of `count`

In [8]:
```
count = 0
for match in matchlist:
    count = count + 1  # Compute current count + 1 and reassign it to cou
```

In [9]:
```
count
```

Out[9]:
```
104
```

- Python does not require you to "declare" variables in advance
- Can introduce new names on the fly
- This can be a source of errors, if you mistype a name

In [10]:
```
count = 0
for row in matchlist:
    countt = count + 1  # Compute current count + 1 and reassign it to co
```

In [11]:
```
count
```

Out[11]:
```
0
```

- If you are lucky, the mistyped name will appear on the right and Python will flag an error

In [12]:
```
count = 0
for row in matchlist:
    count = ccount + 1  # Compute current count + 1 and reassign it to co
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call las
t)
Cell In[12], line 3
      1 count = 0
      2 for row in matchlist:
----> 3     count = ccount + 1  # Compute current count + 1 and reassign i
t to count

NameError: name 'ccount' is not defined
```

What was the maximum number of goals scored in a match?

- Initialize `maxgoals` to `0`
- Update `maxgoals` whenever run target in current row exceeds current maximum
  - Extract run target from tuple of values of current row using positional index
  - In Python, indices always start with 0, so nine entries in the tuple have indices 0,1,,..,8
- `if` specifies conditional execution
  - `if` conditional-expression `:`
    - Expression evaluates to `True` or `False`

- As before, `:` and indentation indicate scope of conditional execution
- Indented commands (update of `maxgoals`) happens only when `if` condition evaluates to `True`

In [13]:
```
maxgoals = 0
for row in matchlist:
    # Goals are in columns 4 to 7, add up these columns to get goals for
    # Positions in a list start with 0, so these are values from index 3
    if row[3] + row[4] + row[5] + row[6] > maxgoals:
        maxgoals = row[3] + row[4] + row[5] + row[6]
```

In [14]:
```
maxgoals
```

Out[14]: 10

- `row[3] + row[4] + row[5] + row[6]` is clumsy to use each time
- Better to assign a separate variable for `row[3] + row[4] + row[5] + row[6]`

In [15]:
```
maxgoals = 0
for row in matchlist:
    goals = row[3] + row[4] + row[5] + row[6]
    if goals > maxgoals:
        maxgoals = goals
```

In [16]:
```
maxgoals
```

Out[16]: 10

- Can also initialize `maxgoals` to first target in the table
- After this, can use `for` to iterate through the entire table
  - Not a problem to read the first row of the table again
  - Avoids complication of skipping the first row and running `for` from the second row

In [17]:
```
row = matchlist[0]
maxgoals = row[3] + row[4] + row[5] + row[6]
for row in matchlist:
    goals = row[3] + row[4] + row[5] + row[6]
    if goals > maxgoals:
        maxgoals = goals
```

In [18]:
```
maxgoals
```

Out[18]: 10

What was the average number of goals scored in a match?

- Iterate to compute sum of all run targets

- Average is total divided by count

In [19]:
```
count = 0
goalsum = 0
for row in matchlist:
    count = count+1
    goals = row[3] + row[4] + row[5] + row[6]
    goalsum = goalsum + goals
goalavg = goalsum/count
```

In [20]:
```
goalavg
```

Out[20]:
```
3.201923076923077
```

- Can also maintain running average instead of running total

In [21]:
```
count = 0
currentavg = 0
for row in matchlist:
    totalsofar = count*currentavg
    count = count+1
    goals = row[3] + row[4] + row[5] + row[6]
    currentavg = (totalsofar + goals)/count
```

In [22]:
```
currentavg
```

Out[22]:
```
3.201923076923077
```

- Alternatively, avoid creating running total altogether
- Note that `count` has already been incremented, so multiply `currentavg` by `count-1`

In [23]:
```
count = 0
currentavg = 0
for row in matchlist:
    count = count+1
    goals = row[3] + row[4] + row[5] + row[6]
    currentavg = (currentavg*(count-1) + goals)/count
```

In [24]:
```
currentavg
```

Out[24]:
```
3.201923076923077
```

How many matches had above average number of goals?

- *Filtered* iteration
- Update the count only if the current run target is $>$ average

In [25]:
```
count = 0
goalsum = 0
```

```
for row in matchlist:
    count = count+1
    goals = row[3] + row[4] + row[5] + row[6]
    goalsum = goalsum + goals
goalavg = goalsum/count

aboveavgcount = 0
for row in matchlist:
    goals = row[3] + row[4] + row[5] + row[6]
    if goals > goalavg:
        aboveavgcount = aboveavgcount + 1
```

In [26]:
```
aboveavgcount
```

Out[26]:
```
40
```

How many cities were venues?

- Maintain a list of venues
  - Need to initialize to empty list to tell Python this value can be used as a list
  - Built-in check `v in l` returns `True` if value `v` is present in list `l`
  - Add new venue to the list --- `l1 + l2` concatenates two lists into a single list

In [27]:
```
venuelist = []
for row in matchlist:
    venue = row[0]  # Leftmost column
    if not(venue in venuelist):
        venuelist = venuelist + [venue]
```

In [28]:
```
venuelist
```

Out[28]:
```
['Mexico City',
 'Zapopan',
 'Toronto',
 'Inglewood',
 'Foxborough',
 'East Rutherford',
 'Philadelphia',
 'Seattle',
 'Atlanta',
 'Santa Clara',
 'Houston',
 'Kansas City',
 'Miami Gardens',
 'Arlington',
 'Guadalupe',
 'Vancouver']
```

In [29]:
```
len(venuelist)
```

Out[29]:
```
16
```

- Can instead use a *dictionary*
  - A collection of key:value pairs

- Initialize to empty dictionary
- Check if venue already exists as a key
  - Create a new key if the current venue is not a key, by assigning a value
  - Value assigned to the key is unimportant here

In [30]:
```
venuedict = {}
for row in matchlist:
    venue = row[0]  # Leftmost column
    if not(venue in venuedict.keys()):
        venuedict[venue] = "Something"
```

In [31]:
```
venuedict
```

Out[31]:
```
{'Mexico City': 'Something',
 'Zapopan': 'Something',
 'Toronto': 'Something',
 'Inglewood': 'Something',
 'Foxborough': 'Something',
 'East Rutherford': 'Something',
 'Philadelphia': 'Something',
 'Seattle': 'Something',
 'Atlanta': 'Something',
 'Santa Clara': 'Something',
 'Houston': 'Something',
 'Kansas City': 'Something',
 'Miami Gardens': 'Something',
 'Arlington': 'Something',
 'Guadalupe': 'Something',
 'Vancouver': 'Something'}
```

- Can extract keys of a dictionary
  - Similar to, but not quite a list

In [32]:
```
venuedict.keys()
```

Out[32]:
```
dict_keys(['Mexico City', 'Zapopan', 'Toronto', 'Inglewood', 'Foxboroug
h', 'East Rutherford', 'Philadelphia', 'Seattle', 'Atlanta', 'Santa Clar
a', 'Houston', 'Kansas City', 'Miami Gardens', 'Arlington', 'Guadalupe',
'Vancouver'])
```

- By convention, Python assumes you mean `d.keys()` if you say `v in d` for a dictionary `d`

In [33]:
```
venuedict2 = {}
for row in matchlist:
    venue = row[0]  # Leftmost column
    if not(venue in venuedict2):  # Short form for venue in venuedict2.ke
        venuedict2[venue] = "Something"
```

In [34]:
```
venuedict2
```

Out[34]:
```
{'Mexico City': 'Something',
 'Zapopan': 'Something',
 'Toronto': 'Something',
 'Inglewood': 'Something',
 'Foxborough': 'Something',
 'East Rutherford': 'Something',
 'Philadelphia': 'Something',
 'Seattle': 'Something',
 'Atlanta': 'Something',
 'Santa Clara': 'Something',
 'Houston': 'Something',
 'Kansas City': 'Something',
 'Miami Gardens': 'Something',
 'Arlington': 'Something',
 'Guadalupe': 'Something',
 'Vancouver': 'Something'}
```

- Can also count the number of matches at each venue
- For each key, maintain a counter
  - When the key is created, initialize the counter to 1
  - If the key already exists, increment the counter

In [35]:
```
venuedict3 = {}
for row in matchlist:
    venue = row[0]  # Leftmost column
    if not(venue in venuedict3):
        venuedict3[venue] = 1
    else:
        venuedict3[venue] = venuedict3[venue] + 1
```

In [36]:
```
venuedict3
```

Out[36]:
```
{'Mexico City': 6,
 'Zapopan': 2,
 'Toronto': 5,
 'Inglewood': 8,
 'Foxborough': 8,
 'East Rutherford': 9,
 'Philadelphia': 6,
 'Seattle': 7,
 'Atlanta': 7,
 'Santa Clara': 7,
 'Houston': 6,
 'Kansas City': 7,
 'Miami Gardens': 9,
 'Arlington': 8,
 'Guadalupe': 5,
 'Vancouver': 4}
```

- Can invert the condition and exchange the two parts of the `if`
- More readable if we avoid negated conditions!

In [37]:
```
venuedict4 = {}
for row in matchlist:
    venue = row[0]  # Leftmost column
    if venue in venuedict4:
```

```
        venuedict4[venue] = venuedict4[venue] + 1
    else:
        venuedict4[venue] = 1
```

In [38]:
```
venuedict4
```

Out[38]:
```
{'Mexico City': 6,
 'Zapopan': 2,
 'Toronto': 5,
 'Inglewood': 8,
 'Foxborough': 8,
 'East Rutherford': 9,
 'Philadelphia': 6,
 'Seattle': 7,
 'Atlanta': 7,
 'Santa Clara': 7,
 'Houston': 6,
 'Kansas City': 7,
 'Miami Gardens': 9,
 'Arlington': 8,
 'Guadalupe': 5,
 'Vancouver': 4}
```

Which team played as Home Team at maximum number of venues?

- Maintain a dictionary where keys are teams
- For each team, associate a list of venues where it has played as Home Team

In [39]:
```
teamvenuedict = {}
for row in matchlist:
    venue = row[0]
    hometeam = row[1]

    # Update the dictionary for hometeam
    if not (hometeam in teamvenuedict):
        teamvenuedict[hometeam] = [venue]
    else:
        if not (venue in teamvenuedict[hometeam]):
            teamvenuedict[hometeam] = teamvenuedict[hometeam] + [venue]
```

In [40]:
```
teamvenuedict
```

Out[40]:
```
{'MEX': ['Mexico City', 'Atlanta'],
 'KOR': ['Zapopan'],
 'CAN': ['Toronto', 'Vancouver', 'Houston'],
 'USA': ['Inglewood', 'Toronto', 'Santa Clara'],
 'QAT': ['Foxborough'],
 'BRA': ['East Rutherford', 'Houston', 'Arlington'],
 'HAI': ['Philadelphia'],
 'AUS': ['Seattle', 'Arlington'],
 'GER': ['Atlanta', 'Foxborough'],
 'NED': ['Santa Clara', 'Philadelphia', 'Guadalupe'],
 'CIV': ['Houston', 'Arlington'],
 'SWE': ['Kansas City'],
 'ESP': ['Miami Gardens', 'Santa Clara', 'Inglewood', 'East Rutherfor
d'],
 'BEL': ['Arlington', 'Houston', 'Seattle'],
 'KSA': ['Guadalupe'],
 'IRN': ['Vancouver'],
 'FRA': ['East Rutherford', 'Arlington', 'Foxborough', 'Miami Gardens'],
 'IRQ': ['Foxborough'],
 'ARG': ['Inglewood', 'Guadalupe', 'Miami Gardens', 'Kansas City'],
 'AUT': ['Santa Clara'],
 'POR': ['Atlanta', 'Kansas City', 'Toronto', 'Miami Gardens'],
 'ENG': ['Miami Gardens', 'Seattle', 'Atlanta'],
 'GHA': ['Houston'],
 'UZB': ['Seattle'],
 'CZE': ['Inglewood', 'Mexico City'],
 'SUI': ['Mexico City', 'Vancouver', 'Seattle'],
 'SCO': ['Arlington', 'East Rutherford'],
 'TUR': ['Guadalupe', 'Inglewood'],
 'ECU': ['Kansas City', 'Atlanta'],
 'TUN': ['Seattle', 'Kansas City'],
 'NZL': ['Miami Gardens', 'Arlington'],
 'URU': ['East Rutherford', 'Miami Gardens'],
 'NOR': ['Toronto', 'Foxborough', 'Miami Gardens'],
 'JOR': ['Foxborough', 'Santa Clara'],
 'COL': ['Philadelphia', 'Miami Gardens', 'Kansas City'],
 'PAN': ['Mexico City', 'East Rutherford'],
 'RSA': ['Zapopan', 'Inglewood'],
 'BIH': ['Toronto'],
 'MAR': ['Foxborough'],
 'PAR': ['Santa Clara', 'Philadelphia'],
 'CUW': ['Houston'],
 'JPN': ['Seattle'],
 'EGY': ['Guadalupe'],
 'CPV': ['Philadelphia'],
 'SEN': ['East Rutherford'],
 'ALG': ['Inglewood'],
 'COD': ['Atlanta'],
 'CRO': ['Philadelphia']}
```

- Again, may be more readable if we avoid (at least one) negated condition

In [41]:
```
teamvenuedict = {}
for row in matchlist:
    venue = row[0]
    hometeam = row[1]

    # Update the dictionary for hometeam
```

```
    if hometeam in teamvenuedict:
        if not (venue in teamvenuedict[hometeam]):
            teamvenuedict[hometeam] = teamvenuedict[hometeam] + [venue]
    else:
        teamvenuedict[hometeam] = [venue]
```

- Display the dictionary as a list of key-value pairs
- `{ k1:v1, k2:v2, ..., km:vm }`

In [42]:
```
teamvenuedict
```

Out[42]:
```
{'MEX': ['Mexico City', 'Atlanta'],
 'KOR': ['Zapopan'],
 'CAN': ['Toronto', 'Vancouver', 'Houston'],
 'USA': ['Inglewood', 'Toronto', 'Santa Clara'],
 'QAT': ['Foxborough'],
 'BRA': ['East Rutherford', 'Houston', 'Arlington'],
 'HAI': ['Philadelphia'],
 'AUS': ['Seattle', 'Arlington'],
 'GER': ['Atlanta', 'Foxborough'],
 'NED': ['Santa Clara', 'Philadelphia', 'Guadalupe'],
 'CIV': ['Houston', 'Arlington'],
 'SWE': ['Kansas City'],
 'ESP': ['Miami Gardens', 'Santa Clara', 'Inglewood', 'East Rutherfor
d'],
 'BEL': ['Arlington', 'Houston', 'Seattle'],
 'KSA': ['Guadalupe'],
 'IRN': ['Vancouver'],
 'FRA': ['East Rutherford', 'Arlington', 'Foxborough', 'Miami Gardens'],
 'IRQ': ['Foxborough'],
 'ARG': ['Inglewood', 'Guadalupe', 'Miami Gardens', 'Kansas City'],
 'AUT': ['Santa Clara'],
 'POR': ['Atlanta', 'Kansas City', 'Toronto', 'Miami Gardens'],
 'ENG': ['Miami Gardens', 'Seattle', 'Atlanta'],
 'GHA': ['Houston'],
 'UZB': ['Seattle'],
 'CZE': ['Inglewood', 'Mexico City'],
 'SUI': ['Mexico City', 'Vancouver', 'Seattle'],
 'SCO': ['Arlington', 'East Rutherford'],
 'TUR': ['Guadalupe', 'Inglewood'],
 'ECU': ['Kansas City', 'Atlanta'],
 'TUN': ['Seattle', 'Kansas City'],
 'NZL': ['Miami Gardens', 'Arlington'],
 'URU': ['East Rutherford', 'Miami Gardens'],
 'NOR': ['Toronto', 'Foxborough', 'Miami Gardens'],
 'JOR': ['Foxborough', 'Santa Clara'],
 'COL': ['Philadelphia', 'Miami Gardens', 'Kansas City'],
 'PAN': ['Mexico City', 'East Rutherford'],
 'RSA': ['Zapopan', 'Inglewood'],
 'BIH': ['Toronto'],
 'MAR': ['Foxborough'],
 'PAR': ['Santa Clara', 'Philadelphia'],
 'CUW': ['Houston'],
 'JPN': ['Seattle'],
 'EGY': ['Guadalupe'],
 'CPV': ['Philadelphia'],
 'SEN': ['East Rutherford'],
 'ALG': ['Inglewood'],
 'COD': ['Atlanta'],
 'CRO': ['Philadelphia']}
```

- Run through the dictionary and record the team with max venues as Team 1
- Keep track of team name and the number of venues

In [43]:
```
maxteam = ""
maxvenues = 0
for team in teamvenuedict:
    if len(teamvenuedict[team]) > maxvenues:
```

```
        maxteam = team
        maxvenues = len(teamvenuedict[team])
```

- Can display more than one value at a time

In [44]:
```
maxteam, maxvenues
```

Out[44]:
```
('ESP', 4)
```

Is being Home Team an advantage?

- Count the rows where the home team won the match
- Check if the fraction of such rows is above a given threshold
- Should we count all matches or only those where there was a result (ignoring draws)?
- The code below counts matches with draws and matches with results separately

In [45]:
```
threshold = 0.6
result = 0   # Number of matches with a result
draw = 0     # Number of matches that were draws
homewin = 0  # Number of matches won by the home team
for row in matchlist:
    # Was it a draw?
    if row[3] + row[5] == row[4] + row[6]: # Note '==' for equality
        draw = draw + 1
    else: # There was a result
        result = result+1
    # Did the home team score more goals?
    if row[3] + row[5] > row[4] + row[6]:
        homewin = homewin + 1
```

- Is the ratio above the threshold?

In [46]:
```
homewin/result >= threshold
```

Out[46]:
```
True
```

In [47]:
```
homewin/result
```

Out[47]:
```
0.6071428571428571
```

In [48]:
```
draw
```

Out[48]:
```
20
```

In [49]:
```
result
```

Out[49]:
```
84
```

In [50]:
```
homewin
```

Out[50]: 51

- What if we had defined *home team advantage* in terms of all matches, including draws?
- Outcome is negative
- Important to check whether what you are computing matches the intent

In [51]:
```
homewin/(result + draw)
```

Out[51]: 0.49038461538461536