PDSP 2025, Lecture 02, 12 August 2025

Set up the table as a list

- Assign a value to a variable: `variable = value`
- Each entry is a row
- Each row is a tuple of columns
- `(Venue, Team 1, Team 2, Toss winner, Match winner, Run target)`
- `Run target` is an integer, all other columns are text (`String`)

In [1]:
```
matchlist = [
    ("Chennai","RCB","CSK","RCB","CSK",174),
    ("Mohali","DC","PK","PK","PK",175),
    ("Kolkata","KKR","SRH","SRH","KKR",209),
    ("Jaipur","RR","LSG","RR","RR",194),
    ("Ahmedabad","GT","MI","MI","GT",169),
    ("Bengaluru","PK","RCB","RCB","RCB",177),
    ("Chennai","CSK","GT","GT","CSK",207),
    ("Hyderabad","SRH","MI","MI","SRH",278),
    ("Jaipur","RR","DC","DC","RR",186),
    ("Bengaluru","RCB","KKR","KKR","KKR",183),
    ("Lucknow","LSG","PK","LSG","LSG",200),
    ("Ahmedabad","SRH","GT","SRH","GT",163),
    ("Visakhapatnam","DC","CSK","DC","DC",192),
    ("Mumbai","MI","RR","RR","RR",126),
    ("Bengaluru","LSG","RCB","RCB","LSG",182),
    ("Visakhapatnam","KKR","DC","KKR","KKR",273),
    ("Ahmedabad","GT","PK","PK","PK",200),
    ("Hyderabad","CSK","SRH","SRH","SRH",166),
    ("Jaipur","RCB","RR","RR","RR",184),
    ("Mumbai","MI","DC","DC","MI",235),
    ("Lucknow","LSG","GT","LSG","LSG",164),
    ("Chennai","KKR","CSK","CSK","CSK",138),
    ("Mohali","SRH","PK","PK","SRH",183),
    ("Jaipur","RR","GT","GT","GT",197),
    ("Mumbai","RCB","MI","MI","MI",197),
    ("Lucknow","LSG","DC","LSG","DC",168),
    ("Mohali","PK","RR","RR","RR",148),
    ("Kolkata","LSG","KKR","KKR","KKR",162),
    ("Mumbai","CSK","MI","MI","CSK",207),
    ("Bengaluru","SRH","RCB","RCB","SRH",288),
    ("Kolkata","KKR","RR","RR","RR",224),
    ("Ahmedabad","GT","DC","DC","DC",90),
    ("Mohali","MI","PK","PK","MI",193),
    ("Lucknow","CSK","LSG","LSG","LSG",177),
    ("Delhi","SRH","DC","DC","SRH",267),
    ("Kolkata","KKR","RCB","RCB","KKR",223),
    ("Mohali","PK","GT","PK","GT",143),
    ("Jaipur","MI","RR","MI","RR",180),
    ("Chennai","CSK","LSG","LSG","LSG",211),
    ("Delhi","DC","GT","GT","DC",225),
    ("Hyderabad","RCB","SRH","RCB","RCB",207),
```

```
    ("Kolkata","KKR","PK","PK","PK",262),
    ("Delhi","DC","MI","MI","DC",258),
    ("Lucknow","LSG","RR","RR","RR",197),
    ("Ahmedabad","GT","RCB","RCB","RCB",201),
    ("Chennai","CSK","SRH","SRH","CSK",213),
    ("Kolkata","DC","KKR","DC","KKR",154),
    ("Lucknow","MI","LSG","LSG","LSG",145),
    ("Chennai","CSK","PK","PK","PK",163),
    ("Hyderabad","SRH","RR","SRH","SRH",202),
    ("Mumbai","KKR","MI","MI","KKR",170),
    ("Bengaluru","GT","RCB","RCB","RCB",148),
    ("Dharamsala","CSK","PK","PK","CSK",168),
    ("Lucknow","KKR","LSG","LSG","KKR",236),
    ("Mumbai","SRH","MI","MI","MI",174),
    ("Delhi","DC","RR","RR","DC",222),
    ("Hyderabad","LSG","SRH","LSG","SRH",166),
    ("Dharamsala","RCB","PK","PK","RCB",242),
    ("Ahmedabad","GT","CSK","CSK","GT",232),
    ("Kolkata","KKR","MI","MI","KKR",158),
    ("Chennai","RR","CSK","RR","CSK",142),
    ("Bengaluru","RCB","DC","DC","RCB",188),
    ("Delhi","DC","LSG","LSG","DC",209),
    ("Guwahati","RR","PK","RR","PK",145),
    ("Mumbai","LSG","MI","MI","LSG",215),
    ("Bengaluru","RCB","CSK","CSK","RCB",219),
    ("Hyderabad","PK","SRH","PK","SRH",215),
    ("Ahmedabad","SRH","KKR","SRH","KKR",160),
    ("Ahmedabad","RCB","RR","RR","RR",173),
    ("Chennai","SRH","RR","RR","SRH",176),
    ("Chennai","SRH","KKR","SRH","KKR",114)
]
```

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

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

Out[2]:
```
[('Chennai', 'RCB', 'CSK', 'RCB', 'CSK', 174),
 ('Mohali', 'DC', 'PK', 'PK', 'PK', 175),
 ('Kolkata', 'KKR', 'SRH', 'SRH', 'KKR', 209),
 ('Jaipur', 'RR', 'LSG', 'RR', 'RR', 194),
 ('Ahmedabad', 'GT', 'MI', 'MI', 'GT', 169),
 ('Bengaluru', 'PK', 'RCB', 'RCB', 'RCB', 177),
 ('Chennai', 'CSK', 'GT', 'GT', 'CSK', 207),
 ('Hyderabad', 'SRH', 'MI', 'MI', 'SRH', 278),
 ('Jaipur', 'RR', 'DC', 'DC', 'RR', 186),
 ('Bengaluru', 'RCB', 'KKR', 'KKR', 'KKR', 183),
 ('Lucknow', 'LSG', 'PK', 'LSG', 'LSG', 200),
 ('Ahmedabad', 'SRH', 'GT', 'SRH', 'GT', 163),
 ('Visakhapatnam', 'DC', 'CSK', 'DC', 'DC', 192),
 ('Mumbai', 'MI', 'RR', 'RR', 'RR', 126),
 ('Bengaluru', 'LSG', 'RCB', 'RCB', 'LSG', 182),
 ('Visakhapatnam', 'KKR', 'DC', 'KKR', 'KKR', 273),
 ('Ahmedabad', 'GT', 'PK', 'PK', 'PK', 200),
 ('Hyderabad', 'CSK', 'SRH', 'SRH', 'SRH', 166),
 ('Jaipur', 'RCB', 'RR', 'RR', 'RR', 184),
 ('Mumbai', 'MI', 'DC', 'DC', 'MI', 235),
 ('Lucknow', 'LSG', 'GT', 'LSG', 'LSG', 164),
 ('Chennai', 'KKR', 'CSK', 'CSK', 'CSK', 138),
 ('Mohali', 'SRH', 'PK', 'PK', 'SRH', 183),
 ('Jaipur', 'RR', 'GT', 'GT', 'GT', 197),
 ('Mumbai', 'RCB', 'MI', 'MI', 'MI', 197),
 ('Lucknow', 'LSG', 'DC', 'LSG', 'DC', 168),
 ('Mohali', 'PK', 'RR', 'RR', 'RR', 148),
 ('Kolkata', 'LSG', 'KKR', 'KKR', 'KKR', 162),
 ('Mumbai', 'CSK', 'MI', 'MI', 'CSK', 207),
 ('Bengaluru', 'SRH', 'RCB', 'RCB', 'SRH', 288),
 ('Kolkata', 'KKR', 'RR', 'RR', 'RR', 224),
 ('Ahmedabad', 'GT', 'DC', 'DC', 'DC', 90),
 ('Mohali', 'MI', 'PK', 'PK', 'MI', 193),
 ('Lucknow', 'CSK', 'LSG', 'LSG', 'LSG', 177),
 ('Delhi', 'SRH', 'DC', 'DC', 'SRH', 267),
 ('Kolkata', 'KKR', 'RCB', 'RCB', 'KKR', 223),
 ('Mohali', 'PK', 'GT', 'PK', 'GT', 143),
 ('Jaipur', 'MI', 'RR', 'MI', 'RR', 180),
 ('Chennai', 'CSK', 'LSG', 'LSG', 'LSG', 211),
 ('Delhi', 'DC', 'GT', 'GT', 'DC', 225),
 ('Hyderabad', 'RCB', 'SRH', 'RCB', 'RCB', 207),
 ('Kolkata', 'KKR', 'PK', 'PK', 'PK', 262),
 ('Delhi', 'DC', 'MI', 'MI', 'DC', 258),
 ('Lucknow', 'LSG', 'RR', 'RR', 'RR', 197),
 ('Ahmedabad', 'GT', 'RCB', 'RCB', 'RCB', 201),
 ('Chennai', 'CSK', 'SRH', 'SRH', 'CSK', 213),
 ('Kolkata', 'DC', 'KKR', 'DC', 'KKR', 154),
 ('Lucknow', 'MI', 'LSG', 'LSG', 'LSG', 145),
 ('Chennai', 'CSK', 'PK', 'PK', 'PK', 163),
 ('Hyderabad', 'SRH', 'RR', 'SRH', 'SRH', 202),
 ('Mumbai', 'KKR', 'MI', 'MI', 'KKR', 170),
 ('Bengaluru', 'GT', 'RCB', 'RCB', 'RCB', 148),
 ('Dharamsala', 'CSK', 'PK', 'PK', 'CSK', 168),
 ('Lucknow', 'KKR', 'LSG', 'LSG', 'KKR', 236),
 ('Mumbai', 'SRH', 'MI', 'MI', 'MI', 174),
 ('Delhi', 'DC', 'RR', 'RR', 'DC', 222),
 ('Hyderabad', 'LSG', 'SRH', 'LSG', 'SRH', 166),
 ('Dharamsala', 'RCB', 'PK', 'PK', 'RCB', 242),
 ('Ahmedabad', 'GT', 'CSK', 'CSK', 'GT', 232),
 ('Kolkata', 'KKR', 'MI', 'MI', 'KKR', 158),
```

```
('Chennai', 'RR', 'CSK', 'RR', 'CSK', 142),
('Bengaluru', 'RCB', 'DC', 'DC', 'RCB', 188),
('Delhi', 'DC', 'LSG', 'LSG', 'DC', 209),
('Guwahati', 'RR', 'PK', 'RR', 'PK', 145),
('Mumbai', 'LSG', 'MI', 'MI', 'LSG', 215),
('Bengaluru', 'RCB', 'CSK', 'CSK', 'RCB', 219),
('Hyderabad', 'PK', 'SRH', 'PK', 'SRH', 215),
('Ahmedabad', 'SRH', 'KKR', 'SRH', 'KKR', 160),
('Ahmedabad', 'RCB', 'RR', 'RR', 'RR', 173),
('Chennai', 'SRH', 'RR', 'RR', 'SRH', 176),
('Chennai', 'SRH', 'KKR', 'SRH', 'KKR', 114)]
```

- Any expression can be evaluated in a Jupyter notebook cell

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

Out[3]: 10

In [4]:
```
x = 9
y = x + 5
```

In [5]:
```
y
```

Out[5]: 14

Questions

- How many matches were played?
- What was the maximum run target?
- What was the average run target?
- How many matches had above average run targets?
- How many cities were venues?
- Which team played as Team 1 at maximum number of venues?
- Is winning the toss an advantage?

How many matches were played?

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

In [6]:
```
len(matchlist)
```

Out[6]: 71

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 [7]:
```
count = 0
for row in matchlist:
    count = count + 1  # Compute current count + 1 and reassign it to cou
```

In [8]:
```
count
```

Out[8]:
```
71
```

- 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 [9]:
```
count = 0
for row in matchlist:
    countt = count + 1  # Compute current count + 1 and reassign it to co
```

In [10]:
```
count
```

Out[10]:
```
0
```

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

In [11]:
```
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[11], 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 run target?

- Initialize `maxtarget` to `-1`
- Update `maxtarget` 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 six entries in the tuple have indices 0,1,2,3,4,5
- `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 `maxtarget`) happens only when `if` condition evaluates to `True`

In [12]:
```
maxtarget = -1
for row in matchlist:
    target = row[5] # Sixth component, indexed 0,1,..,5
    if target > maxtarget:
        maxtarget = target
```

In [13]:
```
maxtarget
```

Out[13]: 288

- No need to assign a separate variable for `row[5]

In [14]:
```
maxtarget = -1
for row in matchlist:
    if row[5] > maxtarget:
        maxtarget = row[5]
```

In [15]:
```
maxtarget
```

Out[15]: 288

- Can also initialize `maxtarget` to first target in the table
- Not a problem to read the first row of the table again

In [16]:
```
firstrow = matchlist[0]
maxtarget = firstrow[5]
for row in matchlist:  # Not a problem to read first row again
    if row[5] > maxtarget:
        maxtarget = row[5]
```

In [17]:
```
maxtarget
```

Out[17]: 288

What was the average run target?

- Iterate to compute sum of all run targets
- Average is total divided by count

In [18]:
```
count = 0
targetsum = 0
```

```
for row in matchlist:
    count = count+1
    targetsum = targetsum + row[5]
targetavg = targetsum/count
```

In [19]:
```
targetavg
```

Out[19]:
```
190.59154929577466
```

- Can also maintain running average instead of running total

In [20]:
```
count = 0
currentavg = 0
for row in matchlist:
    totalsofar = count*currentavg
    count = count+1
    currentavg = (totalsofar + row[5])/count
```

In [21]:
```
currentavg
```

Out[21]:
```
190.59154929577466
```

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

In [22]:
```
count = 0
currentavg = 0
for row in matchlist:
    count = count+1
    currentavg = (currentavg*(count-1) + row[5])/count
```

In [23]:
```
currentavg
```

Out[23]:
```
190.59154929577466
```

How many matches had above average run targets?

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

In [24]:
```
count = 0
targetsum = 0
for row in matchlist:
    count = count+1
    targetsum = targetsum + row[5]
targetavg = targetsum/count

aboveavgcount = 0
for row in matchlist:

```

```
    if row[5] > targetavg:
        aboveavgcount = aboveavgcount + 1
```

In [25]:
```
aboveavgcount
```

Out[25]:
```
34
```

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 [26]:
```
venuelist = []
for row in matchlist:
    venue = row[0]  # Leftmost column
    if not(venue in venuelist):
        venuelist = venuelist + [venue]
```

In [27]:
```
venuelist
```

Out[27]:
```
['Chennai',
 'Mohali',
 'Kolkata',
 'Jaipur',
 'Ahmedabad',
 'Bengaluru',
 'Hyderabad',
 'Lucknow',
 'Visakhapatnam',
 'Mumbai',
 'Delhi',
 'Dharamsala',
 'Guwahati']
```

In [28]:
```
len(venuelist)
```

Out[28]:
```
13
```

- 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 [29]:
```
venuedict = {}
for row in matchlist:
    venue = row[0]  # Leftmost column
```

```
    if not(venue in venuedict.keys()):
        venuedict[venue] = "Something"
```

In [30]:
```
venuedict
```

Out[30]:
```
{'Chennai': 'Something',
 'Mohali': 'Something',
 'Kolkata': 'Something',
 'Jaipur': 'Something',
 'Ahmedabad': 'Something',
 'Bengaluru': 'Something',
 'Hyderabad': 'Something',
 'Lucknow': 'Something',
 'Visakhapatnam': 'Something',
 'Mumbai': 'Something',
 'Delhi': 'Something',
 'Dharamsala': 'Something',
 'Guwahati': 'Something'}
```

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

In [31]:
```
venuedict.keys()
```

Out[31]:
```
dict_keys(['Chennai', 'Mohali', 'Kolkata', 'Jaipur', 'Ahmedabad', 'Benga
luru', 'Hyderabad', 'Lucknow', 'Visakhapatnam', 'Mumbai', 'Delhi', 'Dhar
amsala', 'Guwahati'])
```

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

In [32]:
```
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 [33]:
```
venuedict2
```

Out[33]:
```
{'Chennai': 'Something',
 'Mohali': 'Something',
 'Kolkata': 'Something',
 'Jaipur': 'Something',
 'Ahmedabad': 'Something',
 'Bengaluru': 'Something',
 'Hyderabad': 'Something',
 'Lucknow': 'Something',
 'Visakhapatnam': 'Something',
 'Mumbai': 'Something',
 'Delhi': 'Something',
 'Dharamsala': 'Something',
 'Guwahati': '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 [34]:
```
venuedict3 = {}
for row in matchlist:
    venue = row[0]  # Leftmost column
    if not(venue in venuedict3):  # Short form for venue in venuedict3.ke
        venuedict3[venue] = 1
    else:
        venuedict3[venue] = venuedict3[venue] + 1
```

In [35]:
```
venuedict3
```

Out[35]:
```
{'Chennai': 9,
 'Mohali': 5,
 'Kolkata': 7,
 'Jaipur': 5,
 'Ahmedabad': 8,
 'Bengaluru': 7,
 'Hyderabad': 6,
 'Lucknow': 7,
 'Visakhapatnam': 2,
 'Mumbai': 7,
 'Delhi': 5,
 'Dharamsala': 2,
 'Guwahati': 1}
```

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

In [36]:
```
venuedict4 = {}
for row in matchlist:
    venue = row[0]  # Leftmost column
    if venue in venuedict4:  # Short form for venue in venuedict4.keys()
        venuedict4[venue] = venuedict4[venue] + 1
    else:
        venuedict4[venue] = 1
```

In [37]:
```
venuedict4
```

Out[37]:
```
{'Chennai': 9,
 'Mohali': 5,
 'Kolkata': 7,
 'Jaipur': 5,
 'Ahmedabad': 8,
 'Bengaluru': 7,
 'Hyderabad': 6,
 'Lucknow': 7,
 'Visakhapatnam': 2,
 'Mumbai': 7,
 'Delhi': 5,
 'Dharamsala': 2,
 'Guwahati': 1}
```

Which team played as Team 1 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 Team 1

In [38]:
```
teamvenuedict = {}
for row in matchlist:
    venue = row[0]
    team1 = row[1]

    # Update the dictionary for team1
    if not (team1 in teamvenuedict):
        teamvenuedict[team1] = [venue]
    else:
        if not (venue in teamvenuedict[team1]):
            teamvenuedict[team1] = teamvenuedict[team1] + [venue]
```

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

In [39]:
```
teamvenuedict = {}
for row in matchlist:
    venue = row[0]
    team1 = row[1]

    # Update the dictionary for team1
    if team1 in teamvenuedict:  # Update the value
        if not (venue in teamvenuedict[team1]):
            teamvenuedict[team1] = teamvenuedict[team1] + [venue]
    else: # Create a new key
         teamvenuedict[team1] = [venue]
```

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

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

Out[40]:
```
{'RCB': ['Chennai',
  'Bengaluru',
  'Jaipur',
  'Mumbai',
  'Hyderabad',
  'Dharamsala',
  'Ahmedabad'],
 'DC': ['Mohali', 'Visakhapatnam', 'Delhi', 'Kolkata'],
 'KKR': ['Kolkata', 'Visakhapatnam', 'Chennai', 'Mumbai', 'Lucknow'],
 'RR': ['Jaipur', 'Chennai', 'Guwahati'],
 'GT': ['Ahmedabad', 'Bengaluru'],
 'PK': ['Bengaluru', 'Mohali', 'Hyderabad'],
 'CSK': ['Chennai', 'Hyderabad', 'Mumbai', 'Lucknow', 'Dharamsala'],
 'SRH': ['Hyderabad',
  'Ahmedabad',
  'Mohali',
  'Bengaluru',
  'Delhi',
  'Mumbai',
  'Chennai'],
 'LSG': ['Lucknow', 'Bengaluru', 'Kolkata', 'Hyderabad', 'Mumbai'],
 'MI': ['Mumbai', 'Mohali', 'Jaipur', 'Lucknow']}
```

- 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 [41]:
```
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 [42]:
```
maxteam, maxvenues
```

Out[42]:
```
('RCB', 7)
```

Is winning the toss an advantage?

- Count the rows where the toss winner is the match winner
- Check if the fraction of such rows is above a given threshold

In [43]:
```
threshold = 2/3
count = 0
tossandwin = 0
for row in matchlist:
    count = count + 1
    tosswinner = row[3]
    matchwinner = row[4]
    if tosswinner == matchwinner: # Use == for equality check
        tossandwin = tossandwin + 1
```

- Is the ratio above the threshold?

In [44]:
```
tossandwin/count >= threshold
```

Out[44]:
```
False
```

In [45]:
```
tossandwin/count
```

Out[45]:
```
0.43661971830985913
```