PDSP 2023, Lecture 02, 10 August 2023

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
- `(graduation year, programme, domain, pay package)`
- Pay package is an integer, all other columns are text (String)

In [1]:
```
placements = [
("2017-18","B.Sc","CS",1800000),
("2017-18","M.Sc Appl. Math","CS",1170000),
("2017-18","M.Sc Appl. Math","Manufacturing",730000),
("2017-18","M.Sc Appl. Math","Banking-Finance",1490000),
("2017-18","M.Sc Appl. Math","Banking-Finance",1000000),
("2017-18","M.Sc Appl. Math","Logistics",1250000),
("2017-18","M.Sc Appl. Math","Logistics",1250000),
("2017-18","M.Sc Appl. Math","Banking-Finance",1155000),
("2017-18","M.Sc Appl. Math","Banking-Finance",1155000),
("2017-18","M.Sc Appl. Math","CS",2000000),
("2018-19","B.Sc","Banking-Finance",700000),
("2018-19","B.Sc","Banking-Finance",1480000),
("2018-19","M.Sc Appl. Math","Manufacturing",730000),
("2019-20","M.Sc Data Science","CS",2000000),
("2019-20","M.Sc Data Science","CS",1800000),
("2019-20","M.Sc Data Science","Analytics",1700000),
("2019-20","M.Sc Data Science","Analytics",1700000),
("2019-20","M.Sc Data Science","Analytics",1350000),
("2019-20","M.Sc Data Science","Banking-Finance",1344000),
("2019-20","M.Sc Data Science","Banking-Finance",1335827),
("2019-20","M.Sc Data Science","Banking-Finance",1335827),
("2019-20","M.Sc Data Science","Banking-Finance",1335827),
("2019-20","M.Sc Data Science","Banking-Finance",1335827),
("2019-20","M.Sc Data Science","Banking-Finance",1335827),
("2019-20","M.Sc Data Science","Banking-Finance",1335827),
("2019-20","M.Sc Data Science","Banking-Finance",1335827),
("2019-20","M.Sc Data Science","Banking-Finance",1335827),
("2019-20","M.Sc Data Science","Banking-Finance",1335827),
("2019-20","M.Sc Data Science","Banking-Finance",1335827),
("2019-20","M.Sc Data Science","Banking-Finance",1335827),
("2019-20","M.Sc Comp. Science","Banking-Finance",1335827),
("2019-20","M.Sc Comp. Science","Banking-Finance",1335827),
("2019-20","M.Sc Data Science","CS",1301968),
("2019-20","M.Sc Data Science","CS",1301968),
("2019-20","M.Sc Comp. Science","Logistics",1000000),
("2019-20","M.Sc Data Science","Banking-Finance",1600000),
("2020-21","M.Sc Comp. Science","Banking-Finance",1500000),
("2020-21","M.Sc Data Science","Analytics",900000),
("2020-21","Ph.D Comp. Science","Banking-Finance",1454545),
("2020-21","M.Sc Data Science","Analytics",1300000),
("2020-21","M.Sc Data Science","Analytics",700000),
("2020-21","M.Sc Data Science","Analytics",700000),
("2020-21","M.Sc Data Science","Analytics",700000),
("2020-21","M.Sc Data Science","Analytics",700000),
("2020-21","M.Sc Data Science","Banking-Finance",1350000),
("2020-21","M.Sc Data Science","Banking-Finance",1350000),
("2020-21","M.Sc Data Science","Analytics",1680000),
("2020-21","M.Sc Data Science","Banking-Finance",1360000),
("2020-21","M.Sc Data Science","Banking-Finance",1300000),
("2020-21","M.Sc Data Science","Banking-Finance",1300000),
("2020-21","M.Sc Data Science","Banking-Finance",900000),
("2020-21","M.Sc Data Science","Banking-Finance",1842632),
("2020-21","M.Sc Data Science","Banking-Finance",1842632),
("2020-21","M.Sc Data Science","Banking-Finance",1842632),
("2020-21","M.Sc Data Science","Banking-Finance",1842632),
("2020-21","M.Sc Data Science","Banking-Finance",1842632),
("2020-21","M.Sc Data Science","CS",1150000),
("2020-21","M.Sc Data Science","Analytics",850000),
("2020-21","M.Sc Data Science","Analytics",850000),
("2020-21","M.Sc Comp. Science","Analytics",850000),
("2020-21","M.Sc Data Science","CS",1200000),
("2020-21","M.Sc Data Science","Analytics",1608000),
("2020-21","M.Sc Data Science","Analytics",1608000),
("2020-21","M.Sc Data Science","CS",1100000),
("2020-21","M.Sc Data Science","Analytics",1500000),
("2020-21","M.Sc Data Science","Banking-Finance",1550000),
("2020-21","M.Sc Data Science","Banking-Finance",1550000),
("2020-21","M.Sc Comp. Science","Banking-Finance",1350000),
("2020-21","M.Sc Data Science","Banking-Finance",1200000),
("2020-21","M.Sc Data Science","Analytics",900000),
("2020-21","M.Sc Data Science","Banking-Finance",3300000),
("2020-21","Ph.D Physics","Banking-Finance",3000000),
("2020-21","M.Sc Data Science","Banking-Finance",1500000),
]
```

In [2]:
```
placements
```

Out[2]:
```
[('2017-18', 'B.Sc', 'CS', 1800000),
 ('2017-18', 'M.Sc Appl. Math', 'CS', 1170000),
 ('2017-18', 'M.Sc Appl. Math', 'Manufacturing', 730000),
 ('2017-18', 'M.Sc Appl. Math', 'Banking-Finance', 1490000),
 ('2017-18', 'M.Sc Appl. Math', 'Banking-Finance', 1000000),
 ('2017-18', 'M.Sc Appl. Math', 'Logistics', 1250000),
 ('2017-18', 'M.Sc Appl. Math', 'Logistics', 1250000),
 ('2017-18', 'M.Sc Appl. Math', 'Banking-Finance', 1155000),
 ('2017-18', 'M.Sc Appl. Math', 'Banking-Finance', 1155000),
 ('2017-18', 'M.Sc Appl. Math', 'CS', 2000000),
 ('2018-19', 'B.Sc', 'Banking-Finance', 700000),
 ('2018-19', 'B.Sc', 'Banking-Finance', 1480000),
 ('2018-19', 'M.Sc Appl. Math', 'Manufacturing', 730000),
 ('2019-20', 'M.Sc Data Science', 'CS', 2000000),
 ('2019-20', 'M.Sc Data Science', 'CS', 1800000),
 ('2019-20', 'M.Sc Data Science', 'Analytics', 1700000),
 ('2019-20', 'M.Sc Data Science', 'Analytics', 1700000),
 ('2019-20', 'M.Sc Data Science', 'Analytics', 1350000),
 ('2019-20', 'M.Sc Data Science', 'Banking-Finance', 1344000),
 ('2019-20', 'M.Sc Data Science', 'Banking-Finance', 1335827),
 ('2019-20', 'M.Sc Data Science', 'Banking-Finance', 1335827),
 ('2019-20', 'M.Sc Data Science', 'Banking-Finance', 1335827),
 ('2019-20', 'M.Sc Data Science', 'Banking-Finance', 1335827),
 ('2019-20', 'M.Sc Data Science', 'Banking-Finance', 1335827),
 ('2019-20', 'M.Sc Data Science', 'Banking-Finance', 1335827),
 ('2019-20', 'M.Sc Data Science', 'Banking-Finance', 1335827),
 ('2019-20', 'M.Sc Data Science', 'Banking-Finance', 1335827),
 ('2019-20', 'M.Sc Data Science', 'Banking-Finance', 1335827),
 ('2019-20', 'M.Sc Data Science', 'Banking-Finance', 1335827),
 ('2019-20', 'M.Sc Data Science', 'Banking-Finance', 1335827),
 ('2019-20', 'M.Sc Comp. Science', 'Banking-Finance', 1335827),
 ('2019-20', 'M.Sc Comp. Science', 'Banking-Finance', 1335827),
 ('2019-20', 'M.Sc Data Science', 'CS', 1301968),
 ('2019-20', 'M.Sc Data Science', 'CS', 1301968),
 ('2019-20', 'M.Sc Comp. Science', 'Logistics', 1000000),
 ('2019-20', 'M.Sc Data Science', 'Banking-Finance', 1600000),
 ('2020-21', 'M.Sc Comp. Science', 'Banking-Finance', 1500000),
 ('2020-21', 'M.Sc Data Science', 'Analytics', 900000),
 ('2020-21', 'Ph.D Comp. Science', 'Banking-Finance', 1454545),
 ('2020-21', 'M.Sc Data Science', 'Analytics', 1300000),
 ('2020-21', 'M.Sc Data Science', 'Analytics', 700000),
 ('2020-21', 'M.Sc Data Science', 'Analytics', 700000),
 ('2020-21', 'M.Sc Data Science', 'Analytics', 700000),
 ('2020-21', 'M.Sc Data Science', 'Analytics', 700000),
 ('2020-21', 'M.Sc Data Science', 'Banking-Finance', 1350000),
 ('2020-21', 'M.Sc Data Science', 'Banking-Finance', 1350000),
 ('2020-21', 'M.Sc Data Science', 'Analytics', 1680000),
 ('2020-21', 'M.Sc Data Science', 'Banking-Finance', 1360000),
 ('2020-21', 'M.Sc Data Science', 'Banking-Finance', 1300000),
 ('2020-21', 'M.Sc Data Science', 'Banking-Finance', 1300000),
 ('2020-21', 'M.Sc Data Science', 'Banking-Finance', 900000),
 ('2020-21', 'M.Sc Data Science', 'Banking-Finance', 1842632),
 ('2020-21', 'M.Sc Data Science', 'Banking-Finance', 1842632),
 ('2020-21', 'M.Sc Data Science', 'Banking-Finance', 1842632),
 ('2020-21', 'M.Sc Data Science', 'Banking-Finance', 1842632),
 ('2020-21', 'M.Sc Data Science', 'Banking-Finance', 1842632),
 ('2020-21', 'M.Sc Data Science', 'CS', 1150000),
 ('2020-21', 'M.Sc Data Science', 'Analytics', 850000),
 ('2020-21', 'M.Sc Data Science', 'Analytics', 850000),
 ('2020-21', 'M.Sc Comp. Science', 'Analytics', 850000),
 ('2020-21', 'M.Sc Data Science', 'CS', 1200000),
 ('2020-21', 'M.Sc Data Science', 'Analytics', 1608000),
 ('2020-21', 'M.Sc Data Science', 'Analytics', 1608000),
 ('2020-21', 'M.Sc Data Science', 'CS', 1100000),
 ('2020-21', 'M.Sc Data Science', 'Analytics', 1500000),
 ('2020-21', 'M.Sc Data Science', 'Banking-Finance', 1550000),
 ('2020-21', 'M.Sc Data Science', 'Banking-Finance', 1550000),
 ('2020-21', 'M.Sc Comp. Science', 'Banking-Finance', 1350000),
 ('2020-21', 'M.Sc Data Science', 'Banking-Finance', 1200000),
 ('2020-21', 'M.Sc Data Science', 'Analytics', 900000),
 ('2020-21', 'M.Sc Data Science', 'Banking-Finance', 3300000),
 ('2020-21', 'Ph.D Physics', 'Banking-Finance', 3000000),
 ('2020-21', 'M.Sc Data Science', 'Banking-Finance', 1500000)]
```

Count the number of entries

- Iterate through the list
- `for` variable `in` listname, followed by `:`
- Indent the comands to be performed within each iteration
- `count = count + 1` : rhs is old value of `count` , used to update the variable

In [3]:
```
count = 0
for item in placements:
    count = count + 1
```

In [4]:
```
count
```

Out[4]:
```
73
```

In [5]:
```
item
```

Out[5]:
```
('2020-21', 'M.Sc Data Science', 'Banking-Finance', 1500000)
```

Find the maximum pay package

- Initialize `maxpay` to `0`
- Update `maxpay` whenever pay package in current row exceeds current maximum
  - Extract pay package from tuple of values of current row using positional index
  - In Python, indices always start with 0, so four entries in the tuple have indices 0,1,2,3
- `if` specifies conditional execution
  - `if` conditional-expression -- expression evaluates to `True` or `False`
  - `:` and indentation indicate scope of conditional execution
  - Indented commands (update of `maxpay`) happens only when `if` condition evaluates to `True`

In [6]:
```
maxpay = 0
for item in placements:
    thispay = item[3]
    if thispay > maxpay:
        maxpay = thispay
```

In [7]:
```
maxpay = 0
for item in placements:
    if item[3] > maxpay:
        maxpay = item[3]
```

In [8]:
```
maxpay
```

Out[8]:
```
3300000
```

Average pay package

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

In [9]:
```
totalpay = 0
for item in placements:
    thispay = item[3]
    totalpay = totalpay + thispay
```

In [10]:
```
totalpay
```

Out[10]:
```
99847392
```

In [11]:
```
avgpay = totalpay / count
```

In [12]:
```
avgpay
```

Out[12]:
```
1367772.493150685
```

In [13]:
```
totalpay = 0
count = 0
for item in placements:
    count = count+1
    thispay = item[3]
    totalpay = totalpay + thispay
avgpay = totalpay / count
```

In [14]:
```
avgpay
```

Out[14]:
```
1367772.493150685
```

Number of placements with pay package above the average

- *Filtered* iteration
- Update the count only if the current pay package is $\geq$ average

In [15]:
```
above_avg_count = 0
for item in placements:
    if item[3] > avgpay:
        above_avg_count = above_avg_count + 1
```

In [16]:
```
above_avg_count
```

Out[16]:
```
25
```

Check if there is a placement in Logistics

- Use a boolean variable
- Set to `True` if the value we are searching for is found
  - Note `==` for equality check, to distinguish from `=` for assignment
- Scans the entire list even if the value is found early

In [17]:
```
found = False
for item in placements:
    domain = item[2]
    if domain == "Logistics":
        found = True
```

In [18]:
```
found
```

Out[18]:
```
True
```

Conditional iteration -- `while`

- `while` conditional-expression
  - Like `if`, but at the end of the block, the condition is checked again
  - If the condition is `False`, the loop terminates
- Does not automatically proceed from one list item to the next
  - Maintain an index and extract list item by position
  - Recall that positions in Python start from `0`
  - Need to increment the index manually
- Two checks for loop termination
  - Required value has been found
  - No more items to process (index has reached the value of `count`)

In [19]:
```
count = len(placements)
found = False
rowindex = 0
while (found == False and rowindex < count):
    row = placements[rowindex]
    domain = row[2]
    if domain == "Logistics":
        found = True
    rowindex = rowindex + 1
```

In [20]:
```
found
```

Out[20]:
```
True
```

In [21]:
```
rowindex
```

Out[21]:
```
6
```

Alternatively, abort an iteration

- "break out" of a `for`

In [22]:
```
found = False
for item in placements:
    domain = item[2]
    if domain == "Logistics":
        found = True
        break
```

In [23]:
```
item
```

Out[23]:
```
('2017-18', 'M.Sc Appl. Math', 'Logistics', 1250000)
```

How many placements in Banking-Finance?

- Filtered iteration again
- Increment count only if domain is `"Banking-Finance"`

In [24]:
```
bfcount = 0
for item in placements:
    domain = item[2]
    if domain == "Banking-Finance":
        bfcount = bfcount + 1
```

In [25]:
```
bfcount
```

Out[25]:
```
41
```

Domain with maximum placements

First count the number of placements in each domain

- Use a *dictionary* to record domain-wise counts
  - A collection of `key:value` pairs
  - Initialize to empty dictionary, `{}`
  - For each row, check if current domain is already a key in the dictionary
    - If domain is present, increment the count
    - If domain is not present, create a new key, set count to 1

In [26]:
```
domaincounts = {}
for item in placements:
    domain = item[2]
    if domain in domaincounts.keys():
        domaincounts[domain] = domaincounts[domain] + 1
    else:
        domaincounts[domain] = 1
```

In [27]:
```
domaincounts
```

Out[27]:
```
{'CS': 10,
 'Manufacturing': 2,
 'Banking-Finance': 41,
 'Logistics': 3,
 'Analytics': 17}
```

Now find the domain with the maximum count

- Iterate through the keys of the dictionary
- Keep track of both the key (domain name) and the value (domain count)

In [28]:
```
maxdomaincount = 0

for domain in domaincounts.keys():
    if domaincounts[domain] > maxdomaincount:
        maxdomaincount = domaincounts[domain]
        maxdomainname = domain
```

In [29]:
```
maxdomainname
```

Out[29]:
```
'Banking-Finance'
```

Domain with best average pay package

Compute domain wise pay package total

- We already have domain wise counts
- Now compute total pay, domain wise

In [30]:
```
domainpay = {}
for item in placements:
    thispay = item[3]
    domain = item[2]
    if domain in domainpay.keys():
        domainpay[domain] = domainpay[domain] + thispay
    else:
        domainpay[domain] = thispay
```

In [31]:
```
domainpay
```

Out[31]:
```
{'CS': 14823936,
 'Manufacturing': 1460000,
 'Banking-Finance': 60467456,
 'Logistics': 3500000,
 'Analytics': 19596000}
```

Compute domain wise average pay

- Iterate through keys of domain counts/domain sums
- Create a new dictionary that records domain average pay

In [32]:
```
domainavgpay = {}
for domain in domaincounts.keys():
    domainavgpay[domain] = domainpay[domain]/domaincounts[domain]
```

In [33]:
```
domainavgpay
```

Out[33]:
```
{'CS': 1482393.6,
 'Manufacturing': 730000.0,
 'Banking-Finance': 1474816.0,
 'Logistics': 1166666.6666666667,
 'Analytics': 1152705.8823529412}
```

Compute domain with maximum average pay

- Iterate through keys of domain average pay dictionary
- Keep track of domain name and value

In [34]:
```
bestavgpay = 0
for domain in domainavgpay.keys():
    if domainavgpay[domain] > bestavgpay:
        bestavgpay = domainavgpay[domain]
        bestavgdomain = domain
```

In [35]:
```
bestavgdomain
```

Out[35]:
```
'CS'
```

In [36]:
```
bestavgpay
```

Out[36]:
```
1482393.6
```