1 Pandas basics: building a dataframe from lists, and retrieving data from the dataframe using row and column index references

Here we will repeat basic actions previously described for NumPy. There is significant overlap between NumPy and Pandas (not least because Pandas is built on top of NumPy). Generally speaking Pandas will be used more for data manipulation, and NumPy will be used more for raw calculations (but that is probably somewhat of an over-simplification!).

Pandas allows us to access data using index names or by row/column number. Using index names is perhaps more common in Pandas. You may find having the two different methods available a little confusing at first, but these dual methods are one thing that help make Pandas powerful for data manipulation.

As with NumPy, we will often be importing data from files, but here we will create a dataframe from existing lists.

1.1 Creating an empty data frame and building it up from lists

We start with importing pandas (using pd as the short name we will use) and then create a dataframe.

```
import pandas as pd
df = pd.DataFrame()
```

Let's create some data in lists and add them to the dataframe:

```
names = ['Gandolf','Gimli','Frodo','Legolas','Bilbo']
types = ['Wizard','Dwarf','Hobbit','Elf','Hobbit']
magic = [10, 1, 4, 6, 4]
aggression = [7, 10, 2, 5, 1]
stealth = [8, 2, 5, 10, 5]


df['names'] = names
df['type'] = types
df['magic_power'] = magic
df['aggression'] = aggression
df['stealth'] = stealth
```

We can print the dataframe. Notice that a column to the left has appeared with numbers. This is the index, which has been added automatically.

```
print(df)

OUT:
     names    type  magic_power  aggression  stealth
0  Gandolf  Wizard           10           7        8
1    Gimli   Dwarf            1          10        2
2    Frodo  Hobbit            4           2        5
3  Legolas     Elf            6           5       10
4    Bilbo  Hobbit            4           1        5
```

1.2 Setting an index column

We can leave the index as it is, or we can make one of the columns the index. Note that to change something in an existing dataframe we use 'inplace=True'

```
df.set_index('names', inplace=True)
print (df)
```

```
OUT:
           type  magic_power  aggression  stealth
names
Gandolf  Wizard           10           7        8
Gimli     Dwarf            1          10        2
Frodo    Hobbit            4           2        5
Legolas     Elf            6           5       10
Bilbo    Hobbit            4           1        5
```

1.3 Accessing data with *loc* and *iloc*

Dataframes have two basic methods of accessing data by row (or index) and by column (or header):

loc selects data by index name and column (header) name.

iloc selects data by row or column number

1.4 Selecting rows by index

The *loc* method selects rows by index name, like in Python dictionaries:

```
{print (df.loc['Gandolf']

OUT:
print (df.loc['Gandolf'])

type           Wizard
magic_power        10
aggression          7
stealth             8
Name: Gandolf, dtype: object
```

We can pass multiple index references to the *loc* method using a list:

```
to_find = ['Bilbo','Gimli','Frodo']
print (df.loc[to_find])

OUT:
         type  magic_power  aggression  stealth
names
Bilbo  Hobbit            4           1        5
Gimli   Dwarf            1          10        2
Frodo  Hobbit            4           2        5
```

Row slices may also be taken. For example let us take a row slice from Gimli to Legolas. Unusually for Python this slice includes both the lower and upper index references.

```
print (df.loc['Gimli':'Legolas'])

OUT:
           type  magic_power  aggression  stealth
names
Gimli     Dwarf            1          10        2
Frodo    Hobbit            4           2        5
Legolas     Elf            6           5       10
```

As with other Python slices a colon may be used to represent the start or end. :Gimli would take a slice from the beginning to Gimli. Bilbo: would take a row slice from Bilbo to the end.

1.5 Selecting records by row number

Rather than using an index, we can use row numbers, using the *iloc* method. As with most references in Python the range given starts from the lower index number and goes up to, but does not include, the upper index number.

```
           type  magic_power  aggression  stealth
names
Gandolf  Wizard           10           7        8
Gimli     Dwarf            1          10        2
```

Discontinuous rows may be accessed with *iloc* by building a list:

```
print (df.iloc[[0,1,4]])

OUT:

           type  magic_power  aggression  stealth
names
Gandolf  Wizard           10           7        8
Gimli     Dwarf            1          10        2
Bilbo    Hobbit            4           1        5
```

Or, building up a more complex list of row numbers:

```
rows_to_find = list(range(0,2))
rows_to_find += (list(range(3,5)))


print ('List of rows to find:',rows_to_find)
print()
print (df.iloc[rows_to_find])

OUT:
List of rows to find: [0, 1, 3, 4]

           type  magic_power  aggression  stealth
names
Gandolf  Wizard           10           7        8
Gimli     Dwarf            1          10        2
Legolas     Elf            6           5       10
Bilbo    Hobbit            4           1        5
```

1.6 Selecting columns by name

Columns are selected using square brackets after the dataframe:

```
print (df['type'])

OUT:
names
Gandolf    Wizard
Gimli       Dwarf
Frodo      Hobbit
Legolas       Elf
Bilbo      Hobbit
Name: type, dtype: object

print (df[['type','stealth']])
```

```
OUT:
           type  stealth
names
Gandolf  Wizard        8
Gimli     Dwarf        2
Frodo    Hobbit        5
Legolas     Elf       10
Bilbo    Hobbit        5
```

To take a slice of columns we need to use the *loc* method, using : to select all rows.

```
print (df.loc[:,'magic_power':'stealth'])
```

```
         magic_power  aggression  stealth
names
Gandolf           10           7        8
Gimli              1          10        2
Frodo              4           2        5
Legolas            6           5       10
Bilbo              4           1        5
```

1.7 Selecting columns by number

Columns may also be referenced by number using the *column* method (which allows slicing):

```
print (df[df.columns[1:4]])
```

```
         magic_power  aggression  stealth
names
Gandolf           10           7        8
Gimli              1          10        2
Frodo              4           2        5
Legolas            6           5       10
Bilbo              4           1        5
```

Or *iloc* may be used to select columns by number (the colon shows that we are selecting all rows):

```
print (df.iloc[:,1:3])
```

```
OUT:
         magic_power  aggression
names
Gandolf           10           7
Gimli              1          10
Frodo              4           2
Legolas            6           5
Bilbo              4           1
```

1.8 Selecting rows and columns simultaneously

We can combine row and column references with the *loc* method:

```
rows_to_find = ['Bilbo','Gimli','Frodo']
print (df.loc[rows_to_find,'magic_power':'stealth'])
```

```
OUT:
rows_to_find = ['Bilbo','Gimli','Frodo']

print (df.loc[rows_to_find,'magic_power':'stealth'])
```

```
       magic_power  aggression  stealth
names
Bilbo            4           1        5
Gimli            1          10        2
Frodo            4           2        5
```

Or with *iloc* (referencing row numbers):

```
print (df.iloc[0:2,2:4])
```

```
OUT:
print (df.iloc[0:2,2:4])

         aggression  stealth
names
Gandolf           7        8
Gimli            10        2
```