1 Brief introduction to python

These are the packages we are going to use during the first class.

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns  # visualization tool
```

1.1 For and while loops

Make a loop that prints all the numbers from 1 to 10: first, using `for` operator, then `while`.

Then, let's use a list:

```
lis = [1,3,6,7]
```

Print every element of this list.

1.2 Dictionaries

Let's create a dictionary:

```
dictionary = {'spain':'madrid', 'usa':'vegas'}
```

What values take the `key` and `value` attributes of this dictionary? Change value of `spain` to `barcelone`. Add an entry with key `france` and value `paris`. Remove entry for `usa`.

1.3 User defined functions

We can define functions with default argument values:

```
def f(a, b=1): # b = 1 is default argument
```

or flexible number of arguments:

```
def f(*args): # args can be one or more
def f(** kwargs) # **kwargs is a dictionary
```

Create a function that calculates sum of squares of two variables. Can we use nested function? How should we modify the code so that the function calculates sum of squares of any number of arguments?

2 Exploratory data analysis

Let's download our data:

```
from google.colab import files
uploaded = files.upload()

data = pd.read_csv('pokemon.csv')
```

2.1 First look

1. What is our data? How many features? List columns' names.

2. What type is every column? What are the data dimensions?

3. Can we convert type of `Speed` to float?

4. How to print first 5 lines? Last 5 lines? Value of `Type 1` for the first 5 entries? Lines from 10 to 20 of `HP` and `Attack`?

5. Using `pd.concat` create a new object `concat_data_row` containing first ten rows concatenated with last five rows of `data`. Create a object `concat_data_col` containing values of `Defense` and `Attack` for the ten last pokemons.

6. Print out all entries for which `Attack` is greater than 100 and `Defense` is greater than 200. How many are there?

7. Sort `data` by `Speed` value. Get back to the initial order.

8. Using `data.set_index` create `data1` with hierarchical indexing first by Type 1, then by `Type 2`. Print the pockemon with `Type 1` equal `Fire` and `Type 2` equal `Water`.

9. Using `data.loc` print lines from 10 to 1 (inverse order) of parameters from `HP` to `Defense`.

10. Create a new attribute `Speed level` equal to `high` if the speed is above the average and `low` if below.

11. Plot values of `Attack`, `Defense` and `Speed` on the same plot, then on subplots.

2.2 Univariate analysis

1. What features are categorical? Using `value_counts` check how many pokemons have `Type 1` equal `Poison`. How can we obtain this number using `sum` and logical operations?

2. Use `data.describe()` to print out general statistic information. What is the average speed of the pokemons?

3. Draw a speed distribution for the considered data set. What type of plot should we use? What parameters it has? Compare the obtained plot to the theoretical values of normal distribution.

4. Using `plt.subplots` plot distribution of `Defense` and its cumulative distribution. What can we say about it?

5. We can easily visualize and analyze variable distributions using `seaborn` library. Using the following packages, plot the distribution of `Defense` and fit it to normal distribution. Provide a Q-Q plot. What can we say about the distribution? How can we transform it?

```
from scipy import stats
from scipy.stats import norm
```

2.3 Bivariate analysis

1. Analyze relationship between two categorical variables `Type 1` and `Type 2` using `pd.crosstab`.

2. Plot a boxplot of continuous `Attack` values as function of categorical feature `Legendary`. What can we say about the values of `Defense` for Legendary and not Legendary pokemons? What other features could we study using box plots?

3. Draw a scatter plot between `Attack` and `Defense`.

4. Using `sns.pairplot` make a and analyze summary of scatter plots for quantitative variables.

5. Using `sns.heatmap` visualize pairwise correlations between different features. How could we visualize joint distribution?