Activity A19

1. In the provided exercise, we are working with data related to heart rates for different individuals. We want to create a pandas Series to store this data, where each name serves as the index label, and their corresponding heart rate (in beats per minute) serves as the value in the Series. In the provided script **A19-rates.py**, a list of names of individuals and a list of their corresponding average heart rates (in beats per minute) have been provided.

```
names = ['Alice', 'Bob', 'Charlie', 'David', 'Eva', 'Frank', 'Grace',
'Hannah', 'Ian']

heart_rates = [72, 65, 70, 75, 78, 80, 85, 90, 95]
```

Edit the script A19-rates.py and do the following:

Q1 Import pandas and NumPy

Q2 Create a pandas Series with names as index labels and heart rates as values. Print the Series to screen.

```
Alice      72
Bob        65
Charlie    70
David      75
Eva        78
Frank      80
Grace      85
Hannah     90
Ian        95
dtype: int64
```

Q3 Print the index labels as a 1D array to the screen:

```
['Alice' 'Bob' 'Charlie' 'David' 'Eva' 'Frank' 'Grace' 'Hannah' 'Ian']
```

Q4 Print the values of the Series as 1D array to the screen:

```
[72 65 70 75 78 80 85 90 95]
```

Q5 Slice the Series to obtain heart rates for the first 4 individuals and print the result to the screen. Write in a comment line why in this case using iloc[] is more convenient.

```
Alice      72
Bob        65
Charlie    70
David      75
dtype: int64
```

Q6 Use loc[] with fancy indexing to select heart rates for individuals named Bob, Eva, and Grace. Write in a comment linecwhy using loc[] is more convenient in this case.

```
Bob      65
Eva      78
Grace    85
dtype: int64
```

Q7 Use Boolean indexing with loc[] to select heart rates greater than 70 and less than 90 beats per minute.

```
Alice    72
David    75
Eva      78
Frank    80
Grace    85
dtype: int64
```

Q8 Now, do the same as in point 7, but use iloc[]

```
Alice    72
David    75
Eva      78
Frank    80
Grace    85
dtype: int64
```

Here one additional exercise:

Q9 The following dictionary, called **data,** contains temperature data for different cities.

```
data = {'New York': 75,
    'Los Angeles': 80,
    'Chicago': 70,
    'Houston': 85,
    'Phoenix': 90,
    'Philadelphia': 78,
    'San Antonio': 87,
    'San Diego': 82,
    'Dallas': 88}
```

Create a pandas Series out of dictionary data using only a subset of the dictionary, specified in this list:

```
selected_cities = ['New York', 'Los Angeles', 'Houston', 'Chicago',
'Philadelphia']
```

Print the result to screen:

```
New York        75
Los Angeles     80
Houston         85
Chicago         70
Philadelphia    78
dtype: int64
```

2. In physics, fundamental parameters are those quantities that are considered foundational and independent. Examples of fundamental parameters include mass, length, and time. These parameters are characterized by their independence, universal nature, and role as primary measurements from which other physical quantities can be derived.
Derived parameters are quantities that can be calculated or derived from fundamental parameters. Examples include velocity and acceleration, which are derived from length and time.

In a script called **A19-fundamental.py**, this dictionary has been defined:

```
physics_data = {
    'Parameter': ['Mass', 'Length', 'Time', 'Velocity', 'Acceleration'],
    'Value': [0.1, 2.5, 0.05, 10, 9.8],
    'Unit': ['kg', 'm', 's', 'm/s', 'm/s^2'],
    'Fundamental': [True, True, True, False, False]}
```

Edit the script and do the following:

Q1 Import pandas and if you need NumPy

Q2 Create a pandas' DataFrame out of the dictionary with physics data, and display it to the screen:

```
       Parameter  Value   Unit  Fundamental
0           Mass   0.10     kg         True
1         Length   2.50      m         True
2           Time   0.05      s         True
3       Velocity  10.00    m/s        False
4   Acceleration   9.80  m/s^2        False
```

Q3 Display the 1D array of row labels.

```
[0 1 2 3 4]
```

Q4 Display the 1D array of column labels.

```
['Parameter' 'Value' 'Unit' 'Fundamental']
```

Q5 Print to screen the column Parameter – use loc[]

```
0            Mass
1          Length
2            Time
3        Velocity
4    Acceleration
Name: Parameter, dtype: object
```

Q6 Using loc[], extract the rows for the parameters Length, Velocity, and Acceleration.

```
      Parameter  Value   Unit  Fundamental
1        Length    2.5      m         True
3      Velocity   10.0    m/s        False
4  Acceleration    9.8  m/s^2        False
```

Q7 Extract columns Parameter and Unit by using [ ]

```
      Parameter   Unit
0          Mass     kg
1        Length      m
2          Time      s
3      Velocity    m/s
4  Acceleration  m/s^2
```

Q8 Extract column Fundamental by using the dot notation.

```
0     True
1     True
2     True
3    False
4    False
Name: Fundamental, dtype: bool
```

Q9 Using Boolean indexing with loc[], extract data for Fundamental parameters (where they are True) with Value greater than or equal to 1.

```
  Parameter  Value Unit  Fundamental
1    Length    2.5    m         True
```

Q10 Using a Boolean and fancy indexing with loc[], extract rows with Unit 'kg' or 's' and columns 'Parameter' and 'Unit'.

```
  Parameter Unit
0      Mass   kg
2      Time    s
```

Submit to A19:

A19-rates.py

A19-fundamental.py