# SERIES

1. **Series**: Jest to jednowymiarowa struktura danych w bibliotece Pandas. Każdy punkt danych ma przypisany etykietę, a zbiór tych etykiet nazywany jest indeksem.
2. **Indeksacja**: Możesz uzyskiwać dostęp do elementów w serii za pomocą ich etykiet lub ich indeksów numerycznych.
3. **Tworzenie**: Serię można tworzyć na podstawie listy, słownika lub innych struktur danych.
4. **Operacje statystyczne**: Pandas oferuje wiele metod do obliczania różnych statystyk dla serii, takich jak suma, średnia, mediana, odchylenie standardowe itp.

* **pd.Series(data, index)**: Tworzy nowy obiekt typu Series na podstawie danych i opcjonalnego indeksu.
* **.head(n)**: Zwraca pierwsze n elementów serii, gdzie n jest liczbą całkowitą.
* **.tail(n)**: Zwraca ostatnie n elementów serii, gdzie n jest liczbą całkowitą.
* **.index**: Zwraca indeks serii.
* **.values**: Zwraca wartości serii jako tablicę NumPy.
* **.describe()**: Generuje podstawowe statystyki opisowe dla serii, takie jak liczba elementów, średnia, odchylenie standardowe, minimum, maksimum itp.
* **.sum()**: Oblicza sumę wszystkich wartości w serii.
* **.mean()**: Oblicza średnią wartość w serii.
* **.median()**: Oblicza medianę wartości w serii.
* **.max()** i **.min()**: Zwraca maksymalną i minimalną wartość w serii.
* **.idxmax()** i **.idxmin()**: Zwraca indeks maksymalnej i minimalnej wartości w serii.
* **.sort\_values()**: Sortuje wartości serii.
* **.unique()**: Zwraca unikalne wartości w serii.
* **.value\_counts()**: Zwraca liczbę wystąpień każdej unikalnej wartości w serii.
* **.apply(func)**: Zastosowuje funkcję do każdej wartości w serii.
* **.map(dict)**: Zastępuje wartości w serii zgodnie z określonymi w słowniku.
* **.append(series)**: Dołącza inną serię do bieżącej serii.
* **.drop(labels)**: Usuwa elementy serii na podstawie etykiet.

# DATAFRAME

1. **DataFrame**: Jest to dwuwymiarowa struktura danych składająca się z wierszy i kolumn, podobna do arkusza kalkulacyjnego.
2. **Tworzenie**: DataFrame można tworzyć poprzez dodawanie kolumn do struktury danych. Każda kolumna jest tworzona na podstawie kolekcji danych reprezentowanej jako lista.
3. **Operacje statystyczne**: Pandas oferuje wiele metod do obliczania różnych statystyk dla DataFrame, takich jak suma, średnia, mediana, opisowe statystyki itp.

* **pd.DataFrame()**: Tworzy nowy obiekt typu DataFrame.
* **.head(n)** i **.tail(n)**: Zwracają odpowiednio pierwsze i ostatnie n wiersze DataFrame.
* **.describe()**: Generuje podstawowe statystyki opisowe dla DataFrame.
* **.mean()**, **.median()**, **.max()**, **.min()**: Obliczają odpowiednio średnią, medianę, maksimum i minimum wartości w DataFrame.
* **.sum()**: Oblicza sumę wartości w kolumnach DataFrame.
* **.isin(values)**: Filtruje wiersze na podstawie wartości w kolumnie, zwracając DataFrame zawierający tylko pasujące wiersze.
* **.concat()**: Łączy DataFrame wzdłuż osi.
* **.read\_csv()**: Wczytuje dane z pliku CSV do DataFrame.

# ROWS AND COLUMNS

1. **Specyfikowanie nazw wierszy i kolumn lub ich numerów**: Możesz wybierać określone wiersze i kolumny, używając numerów wierszy i kolumn lub ich nazw.
2. **Podawanie zakresu wierszy i kolumn**: Możesz wybierać zakresy wierszy i kolumn, używając numerów lub nazw, oddzielając je dwukropkiem.
3. **Wyświetlanie dowolnych pierwszych lub ostatnich wierszy danych**: Możesz wyświetlać pierwsze lub ostatnie kilka wierszy danych za pomocą metod head() lub tail().

* **.iloc[row, column]**: Wybiera element DataFrame na podstawie jego indeksu numerycznego.

::2,::3 -> co drugi wiersz, co trzecia kolumna

* **.loc[row, column]**: Wybiera element DataFrame na podstawie jego etykiety wiersza i kolumny.
* **.head(n)**: Zwraca pierwsze n wierszy DataFrame.
* **.tail(n)**: Zwraca ostatnie n wierszy DataFrame.

# DATA TO FILE

1. **Metody zapisywania danych do pliku**: Aby zapisać DataFrame do pliku, można użyć jednej z dostępnych metod.

* **.to\_csv('filename.csv', index=False)**: Zapisuje DataFrame do pliku CSV. Parametr index=False oznacza, że nie zostanie zapisany indeks DataFrame.
* **.to\_json('filename.json', orient='records')**: Zapisuje DataFrame do pliku JSON. Parametr orient='records' oznacza, że DataFrame zostanie zapisany jako lista rekordów.
* **.to\_html('filename.html', index=False)**: Zapisuje DataFrame do pliku HTML. Parametr index=False oznacza, że nie zostanie zapisany indeks DataFrame.

# CREATING QUERY

1. **Tworzenie zapytań**: Możesz tworzyć zapytania, aby wybierać konkretne dane z DataFrame na podstawie określonych kryteriów.

* **.query("condition")**: Tworzy zapytanie, które wybiera wiersze spełniające określone warunki.
* **.shape[0]:** pobranie liczby wierszy zwróconych przez zapytanie

**Przykłady zadań:**

1. Wyświetlenie listy uniwersytetów w Krakowie.

*universities.query("City=='Krakow'")*

1. Wyświetlenie listy uniwersytetów w Krakowie z co najmniej 10 000 studentami.

*universities.query("City=='Krakow' and Students >= 10000")*

# DATA GROUPING

1. **Grupowanie danych**: Pozwala na grupowanie danych w DataFrame na podstawie określonych kryteriów, takich jak kolumny, i wykonywanie na tych grupach operacji agregujących, takich jak suma, średnia, minimum, maksimum itp.

* **.groupby()**: Grupuje wiersze DataFrame na podstawie określonych kolumn.
* **.sum(), .mean(), .min(), .max(), .count(), .median()**: Wykonuje operacje agregujące na grupach danych.

Metoda *groupby*() w bibliotece Pandas służy do grupowania danych w obiekcie DataFrame na podstawie określonych kryteriów, na przykład wartości w kolumnach. Jest to przydatne narzędzie do analizy danych i wykonywania operacji agregujących na grupach danych. Oto kilka kroków, jak stosować *groupby*():

**1. Wybór kolumn do grupowania:**

Najpierw należy wybrać kolumny, które chcesz użyć do grupowania danych. Może to być jedna kolumna lub kombinacja kilku kolumn.

**2. Wywołanie metody groupby():**

Następnie należy użyć metody groupby() na obiekcie DataFrame i przekazać nazwę kolumny lub listę nazw kolumn, według których chcesz dokonać grupowania.

*grouped\_data = df.groupby('Column\_name')*

**3. Wykonanie operacji agregujących:**

Teraz, gdy masz obiekt *groupby*(), możesz wykonywać różne operacje agregujące na grupach danych, takie jak suma, średnia, minimum, maksimum itp. Możesz wykonać to za pomocą wbudowanych metod Pandas, takich jak sum(), mean(), min(), max(), count().

# DATA JOINING

1. **Łączenie danych**: W przypadku posiadania danych w różnych tabelach, czasami konieczne jest ich połączenie w jedną wspólną kolekcję danych. Można to zrobić za pomocą operacji łączenia danych.

* **pd.merge(left, right, on='key\_column', how='merge\_type')**: Metoda ta służy do łączenia dwóch obiektów DataFrame w oparciu o wspólne kolumny lub indeksy.

Metoda merge() w bibliotece Pandas służy do łączenia dwóch obiektów DataFrame w oparciu o wspólne kolumny lub indeksy. Pozwala to na tworzenie jednego, złączonego zbioru danych na podstawie określonych kryteriów. Oto sposób użycia metody merge():

**Składnia:**

*merged\_df = pd.merge(left, right, on='key\_column', how='merge\_type')*

**Parametry:**

* left, right: Obiekty DataFrame, które mają być połączone.
* on: Kolumna lub lista kolumn, które stanowią klucz łączenia.
* how: Typ łączenia, może to być 'left', 'right', 'outer', 'inner'. Domyślnie jest to 'inner'.

1. **Łączenie na podstawie jednej kolumny:**

*merged\_df = pd.merge(df1, df2, on='common\_column')*

1. **Łączenie na podstawie różnych kolumn w obu DataFrame:**

*merged\_df = pd.merge(df1, df2, left\_on='column\_df1', right\_on='column\_df2')*

1. **Łączenie na podstawie indeksów:**

*merged\_df = pd.merge(df1, df2, left\_index=True, right\_index=True)*

1. **Rodzaje łączenia (how):**
   * 'inner': Zwraca tylko wiersze, które mają wspólne wartości klucza w obu obiektach DataFrame.
   * 'left': Zwraca wszystkie wiersze z lewego obiektu DataFrame i pasujące wiersze z prawego obiektu DataFrame.
   * 'right': Zwraca wszystkie wiersze z prawego obiektu DataFrame i pasujące wiersze z lewego obiektu DataFrame.
   * 'outer': Zwraca wszystkie wiersze z obu obiektów DataFrame, łącząc je w oparciu o klucz.

# DATA SORTING

1. **Sortowanie danych**: Sortowanie danych w obiekcie DataFrame można wykonać za pomocą funkcji sort\_values(). Dane mogą być sortowane rosnąco lub malejąco.

* **sort\_values(by='column\_name', ascending=True)**: Metoda ta sortuje dane w obiekcie DataFrame według wartości w określonej kolumnie.

**Przykłady zadań:**

1. **Sortowanie alfabetyczne**: Użyj funkcji sort\_values() do posortowania danych alfabetycznie według nazwy.
2. **Sortowanie według wartości**: Posortuj dane według określonej kolumny, np. według wyników egzaminu.
3. **Sortowanie wielokrotne**: Możesz także sortować dane według kilku kryteriów, na przykład najpierw według wyniku, a następnie według nazwiska.

**Sortowanie alfabetyczne:**

*# Sortowanie alfabetyczne według kolumny 'Name'*

*df\_sorted\_alphabetically = df.sort\_values(by='Name')*

**Sortowanie według wartości:**

*# Sortowanie według kolumny 'Total' (np. punktów w egzaminie)*

*# Domyślnie sortuje się rosnąco*

*df\_sorted\_by\_value = df.sort\_values(by='Total')*

*# Można również sortować malejąco, ustawiając parametr 'ascending' na False*

*df\_sorted\_by\_value\_descending = df.sort\_values(by='Total', ascending=False)*

**Sortowanie wielokrotne:**

*# Sortowanie najpierw według wyniku (kolumna 'Total'), a następnie według nazwiska (kolumna 'Name')*

*df\_sorted\_multiple = df.sort\_values(by=['Total', 'Name'])*

# EXTRA COLUMNS

1. **Rozszerzanie DataFrame**: Możesz rozszerzyć istniejący DataFrame dodając kolumny, których wartości zależą od wartości już istniejących kolumn.

* **Dodawanie kolumn**: Użyj operatora indeksowania [ ] i przypisz nową kolumnę do DataFrame, np. *df['New\_Column'] = ....*

**Przykłady:**

1. **Dodawanie kolumny z sumą punktów**: Możesz dodać nową kolumnę, która będzie sumą punktów uzyskanych w różnych testach.

*df['Total'] = df['Test1'] + df['Test2']*

1. **Dodawanie kolumny z wartościami obliczonymi na podstawie innych kolumn**: Możesz obliczyć średnią arytmetyczną z istniejących kolumn i dodać ją jako nową kolumnę.

*df['ArithmeticMean'] = df['Total'] / 2*

**Przykłady zadań:**

1. **Dodawanie kolumny z limitem prędkości**: Utwórz kolumnę z limitem prędkości i oblicz różnicę między prędkością pojazdu a limitem.

*df['Limit'] = 50*

*df['Exceeded'] = df['KMH'] - df['Limit']*

W ten sposób można dynamicznie rozszerzać DataFrame o dodatkowe kolumny, co może być przydatne do przeprowadzania analizy danych i generowania nowych danych na ich podstawie.

data series: values = [‘a’,’b’,’c’] -> pd.Series() -> values.append(‘d’) -> pd.Series()

# DATAFRAME CHART

1. **Tworzenie wykresu z DataFrame**: Możesz generować wykresy na podstawie danych zawartych w DataFrame. Możesz tworzyć wykresy liniowe, słupkowe i wiele innych rodzajów wykresów.

* **plot()**: Metoda ta umożliwia generowanie wykresów na podstawie danych zawartych w DataFrame. Możesz użyć jej do generowania różnych rodzajów wykresów, takich jak wykresy liniowe, słupkowe itp.

**Przykłady:**

**df.index = [‘a’,’b’,’c’]**

**chart\_data.set\_indes([‘kolumna1’], inplace = True)**

**chart\_data.plot()**

1. **Tworzenie wykresu liniowego z dwóch serii danych**: Możesz generować wykresy liniowe z danych zawartych w dwóch serii danych.

*df.plot(title='Title', xlabel='X Label', ylabel='Y Label')*

1. **Tworzenie wykresu słupkowego z jednej kolumny danych**: Możesz wygenerować wykres słupkowy z jednej kolumny danych.

df['Column\_Name'].plot(kind='bar', title='Title', xlabel='X Label', ylabel='Y Label')

1. **Tworzenie wykresu słupkowego poziomego**: Możesz również tworzyć wykresy słupkowe poziome.

*df.plot(kind='barh', title='Title', xlabel='X Label', ylabel='Y Label')*

1. **Tworzenie podwykresów**: Możesz generować podwykresy, czyli wykresy, które są umieszczone wewnątrz jednego wykresu.

*df.plot(subplots=True, layout=(3,1), title='Title', xlabel='X Label', ylabel='Y Label')*

# SERIES CHART

1. **Tworzenie wykresu z serii danych**: Seria danych to jednowymiarowy obiekt w bibliotece pandas, który może być używany do tworzenia wykresów. Możesz tworzyć różne rodzaje wykresów, takie jak wykresy liniowe, słupkowe, kołowe itp., na podstawie serii danych.

* **plot()**: Metoda ta pozwala na generowanie wykresów na podstawie serii danych. Możesz użyć różnych wartości parametru kind, aby określić rodzaj wykresu, np. 'line' dla wykresu liniowego, 'bar' dla wykresu słupkowego, 'pie' dla wykresu kołowego itp.
* **index**: Atrybut serii danych, który pozwala określić etykiety osi X na wykresie.

**Przykłady:**

1. **Tworzenie wykresu liniowego z serii danych**: Możesz generować wykresy liniowe na podstawie serii danych.

*ds.plot()*

1. **Dodawanie etykiet na osi X**: Aby dodać etykiety na osi X, możesz przypisać etykiety indeksu do serii danych.

*ds.index = ['Label1', 'Label2', 'Label3']*

*ds.plot()*

1. **Tworzenie wykresu słupkowego z serii danych**: Możesz tworzyć wykresy słupkowe na podstawie serii danych.

*ds.plot(kind='bar')*

1. **Tworzenie wykresu kołowego z serii danych**: Możesz tworzyć wykresy kołowe na podstawie serii danych.

*ds.plot(kind='pie', autopct='%1.0f%%')*

# Procesowanie w arkuszu kalkulacyjnym

**Arkusz tabelaryczny**

1. **Tworzenie tabeli**: W arkuszu kalkulacyjnym Excel możesz tworzyć tabele, które ułatwiają organizację danych i wykonywanie operacji na nich.
2. **Dodawanie wierszy i kolumn**: Możesz dodawać nowe wiersze i kolumny do tabeli, co umożliwia rozszerzanie danych.
3. **Sortowanie danych**: Możesz sortować dane w tabeli według różnych kryteriów, np. alfabetycznie, według wartości liczbowych itp.
4. **Filtrowanie danych**: Możesz filtrować dane w tabeli, aby wyświetlić tylko te, które spełniają określone warunki.
5. **Obliczanie statystyk**: Możesz obliczać różne statystyki dla danych w tabeli, takie jak liczba, suma, minimum, maksimum, średnia itp.

**Tabela przestawna**

1. **Tworzenie tabeli przestawnej**: Tabela przestawna to interaktywne narzędzie do podsumowywania i analizowania dużych zestawów danych w arkuszach kalkulacyjnych, które pozwala na reorganizację, grupowanie i obliczanie wartości z danych.
2. **Tworzenie wierszy, kolumn i wartości**: Możesz tworzyć w tabeli przestawnej wiersze, kolumny i wartości, które pomagają w analizie danych.
3. **Tworzenie wykresów**: Możesz tworzyć wykresy na podstawie danych z tabeli przestawnej, co ułatwia wizualizację danych i zrozumienie trendów.

**Power Query**

1. **Importowanie danych**: Power Query pozwala na importowanie danych z różnych źródeł, takich jak pliki CSV, strony internetowe, bazy danych itp.
2. **Przetwarzanie danych**: Możesz manipulować i przetwarzać zaimportowane dane za pomocą Power Query, np. usuwając niepotrzebne kolumny, łącząc dane z innymi źródłami, dodając nowe kolumny itp.
3. **Tworzenie wykresów**: Po przetworzeniu danych za pomocą Power Query możesz tworzyć wykresy, które pomagają w wizualizacji i analizie danych.