חלק א'

28/12/20דניס.ט

#

נושא האינדקס או המפתח בעיברית ובחירת הנתונים הינם הליבה של ספריית ה-Pandas. אחת הסיבות לכך שעם השנים Pandas צמחה באופן משמעותי עם בקשת המשתמשים ומכך נוצרו הרבה דרכים לבחירת מידע מתוך ה-DataFrame או סדרות אחרות. קריאת דוקומנטציה בפני עצמה יכולה להוות אתגר ובמיוחד למתחילים, אילו מתכנתים וותיקים יותר בקלות ימצאו לעצמם את הטכניקה המועדפת לפעמים בלי להבין יש שיטה פשוטה יותר, מהירה ואמינה לבחור או לשנות את התנותים.

מכיוון שזה יכול להיות נושא מסובך ומבלבל, בחרתי לפרק אותו למספר כתבות קטנות יותר בכדי שלא להעמיס ולעבור מהבסיס לפתרון בעיות מורכבות בהדרגה. המתודות אשר משמשות לבחירה ואינדקס הן מהמבלבלות תוך עבודה עם Pandas בגלל התנהגותם המתשנה בהתאמלסוגים שונים של ארגומנטים.

#

אינדקס, נתחיל מהיסודות

#

מכיוון שזאת תהיה הכתבה הראשונה בדרה שמכסה את נושא האינדקס ובחירת המידע, הפוסט יעסוק בבחירה של מידע על פי תג האינדקס או הטיה של שלמים. בעתיד נעבור גל על חיתוך, אינקס בוליאני, שאילתות, שימוש ב-jsin חכתים רוחביים ועוד...

#

נתוני הבדיקה שלנו

#

כאשר אנו מתחילים, אנחנו רוצים לקבל נתונים. אבל במקום להכין נתונים מזוייפים לעבודה, בוא נייבא נתונים אמיתיים מהרשת שהם קצת יותר מעניינים. אני גר באיזור שיקגו, אז בדקתי בפורטל הנתונים של שיקגו כמה מערכי נתונים. הקטן ביותר שמצאתי היה מערך נתונים של ציוני דרך המפורסמים בשיקגו. יש בו שדות טקסט, מספרים ותאריכים. אני הולך לתמרן אותו קצת כדי שיהיה אפשר להסביר מושגים נוספים. דוגמאות אלו בוצעו בפייתון ו-Pandas.

```
>>> import pandas as pd
>>> import numpy as np
>>>
>>> df = pd.read_json("https://data.cityofchicago.org/resource/tdab-kixi.json")
>>> df.head(3)
                            landmark_name     id                   address  ... :@computed_region_awaf_s7ux  date_built               architect
0  Vassar Swiss Underwear Company Building  L-265  2543 - 2545 W Diversey Av  ...                        24.0         NaN                     NaN
1                     Mathilde Eliel House  L- 89            4122 S Ellis Av  ...                         1.0        1886        Adler & Sullivan
2                       Manhattan Building  L-139          431 S Dearborn St  ...                        48.0        1891  William LeBaron Jenney
[3 rows x 15 columns]
>>> df.dtypes
landmark_name                   object
id                              object
address                         object
landmark_designation_date       object
latitude                       float64
longitude                      float64
location                        object
:@computed_region_rpca_8um6    float64
:@computed_region_vrxf_vc4k    float64
:@computed_region_6mkv_f3dw    float64
:@computed_region_bdys_3d7i    float64
:@computed_region_43wa_7qmu    float64
:@computed_region_awaf_s7ux    float64
date_built                      object
architect                       object
dtype: object
>>>
>>> df['landmark_designation_date'] = pd.to_datetime(df['landmark_designation_date'])
>>>
>>> df = df[['landmark_name', 'id', 'address', 'landmark_designation_date',
...          'latitude', 'longitude', 'location', 'date_built', 'architect']]
>>> df.columns
Index(['landmark_name', 'id', 'address', 'landmark_designation_date',
       'latitude', 'longitude', 'location', 'date_built', 'architect'],
      dtype='object')
```

#

צירים

#

לשני מבני הנתונים העיקריים ב-PANDAS יש לפחות ציר אחד. לסדרה יש ציר אחד, האינדקס. ל-DataFrame שני צירים, האינדקס והעמודות. כדאי לציין כאן כי בכל הפונקציות DataFrame שניתן להכיל על שורות או העמודות, ציר-0 מתייחס לאינדקס וציר-1 לעמודות. אנו יכולים לבדוק אותם במדגם DataFrame שלנו. אנו נבחר בעמודה landmark_name כסדרה לדוגמא כדי להדגים את היסודות לסדרה. הינך יכול להבחין כי העמודה (שהיא סדרה) וכל ה-DataFrame חולקים את אותו האינדקס.

```
>>> s = df['landmark_name']
>>> print("Series index:", s.index)
Series index: RangeIndex(start=0, stop=317, step=1)
>>> print("DataFrame index:", df.index)
DataFrame index: RangeIndex(start=0, stop=317, step=1)
```

#

אינדקס

#

ב-Pandas, אינדקס(או תת-מחלקה) מאפשר למבני הנתונחם המשתמשים בו תמיכה בחיפוש(או בבחירה), יישור נתונים(במיוחד על נתוני סדרות זמן, כאשר כל התצפיות צריכות להיות מיושרות עם זמן התצפית שלהן), ואינדקס מחדש(שינויי האינדקס הבסיסי לערכים שונים, תוך שמירה על יישור הנתונים). ישנם מספר סוגים של מדדים, אך לעת עתה, תבונן רק על RangeIndex הפשוטה שבה אנו משתמשים ב-DataFrame הנוכחי, עם ערכים שלמים.

#

בחירה בסיסית עם []

#

נתחיל בצורה בסיסית של בחירה, באמצעות המפעיל- [ ], אפשר בפייתון ממפה לפונקציה __getitem__ של ה-Class (לא לדאוג למי שלא מכיר או נתקל בעבודה מול אובייקטים בפייתון). בהסתמך על האובייקט ב-Pandas אם הוא סדרה או DataFrame והארגומנטים אשר נעביר לתוך הפונקצייה, אנו נקבל תוצאות שונות. נתחיל עם היסודות, תוך העברת ארגומנט אחד.

#

סדרה

#

עם סדרה, הבקשה תחזיר ערך סקלרי יחיד המתאים לערך באותה תווית האינדקס. במידה ונעביר ערך לתווית שאינה קיימת, נקבל שגיאה KeyError. כמו כן אם נעביר מספר שלם ולתווית יש ערך זהה, נקבל את הערך חזרה. אבל אם אין לנו ערך מספרי בתווית האינדקס, נקבל חזרה ערך לפי מיקום. זה מאוד נח אבל יכול להיות מבלבל. עכשיו ניתן ל- DataFrame (ולסדרה) אינדקס חדש מכיוון שה-RangeIndex יכול להפוך חלק ניכר מהדוגמאות הבאות למבלבלות מאוד. חשוב לנו מאוד להבדיל בין גישה לאלמנטים לפי תווית ולפי מיקום. אם תווית האינדקסים שלנו הם מספרים שלמים, לא נוכל לראות את ההבדל! מכיוון שמערך הנתונים הזה כבר כולל עמודות עם מזהה ייחודי, נשתמש בזה במקום אחר.

```
>>> df.index = df['id'].str.replace(' ', '')
>>> s = df['landmark_name']
>>> df.index
Index(['L-265', 'L-89', 'L-139', 'L-12', 'L-88', 'L-318', 'L-85', 'L-149',
       'L-286', 'L-71',
       ...
       'L-241', 'L-133', 'L-169', 'L-277', 'L-164', 'L-310', 'L-103', 'L-236',
       'L-65', 'L-224'],
      dtype='object', name='id', length=317)
```

כעת כאשר האינדקס שלנו אינו מכיל ערכים מסוג מספרים שלמים, כאשר אנו נקרא לו עם מספרים שלמים הם יחשבו כמו ארגומנטים לפי מיקום. אם לאינדקס היו ערכים מסוג מספרים שלמים, הם היו עולים ראשונים ולא הערכים לפי מיקום, מבלבל? זאת אחת הסיבות להמשיך לקרוא ולראות מדוע יש דרכים טובות יותר לעשות זאת.

```
>>> print("The value for L-265:", s['L-265'])
The value for L-265: Vassar Swiss Underwear Company Building
>>> print("The first value:", s[0])
The first value: Vassar Swiss Underwear Company Building
>>> print("The value for L-139:", s['L-139'])
The value for L-139: Manhattan Building
>>> print("The third value:", s[2])
The third value: Manhattan Building
>>> try:
...     s['L-900']
... except KeyError as ke:
...     print("Exception: ", ke)
...
Exception:  'L-900'
```

אמנם לעתים רחוקות אני משתמש בה, אך קיימת שיטת get זמינה שתחזיר None אם הארגומנט לא נמצא באינדקס במקום להעלות KeyError.

```
>>> print("The first value:", s.get(0))
The first value: Vassar Swiss Underwear Company Building
>>> print("Is there a value at 'L-900'?: ", s.get('L-900'))
Is there a value at 'L-900'?:  None
```