Change Value

Áp Dụng Nhiều Hàm với Pipe:	`df.pipe(tinh_tong).pipe(bo_cot, " a")`
Áp Dụng Hàm Cho Một Cột:	`df["col1"] = df[" col 1"].a pp ly( lambda row: row * 2)`
Áp Dụng Hàm Cho Từng Phần Tử:	`df.applymap(lambda val: " fai led " if val < 5 else " pas sed ")`
Gán Giá Trị Cho Nhiều Cột Mới:	`df = df.ass ign (co l3= lambda x: x.col1  100 + x.col2 ).a ssi gn( col 4=l ambda x: x.col2 x.col3)`
Thay Đổi Giá Trị của Series Bằng Dictionary:	`s.map({"a": 1, " b": 2, " c": 3})`
Chuyển Mỗi Phần Tử Trong Iterable Thành Một Dòng:	`df.explode("a")`
Tách Chuỗi Thành Nhiều Dòng:	`df.a = df.a.s tr.s pl it( " ,") df.exp lod e("a ")`
Forward Fill Trong pandas:	`df = df.fil lna (me tho d="f fil l")`
Thay Thế Giá Trị Trống Bằng Giá Trị Phổ Biến Nhất:	`df.fillna(df.mode().iloc[0])`
Mã Hóa Giá Trị Phân Loại Trong DataFrame:	`df["encoded_col1"] = df[" col 1"].a st ype ("ca teg ory " ).c at.c odes`
Tối Ưu Hóa Mã với Phép Toán Vector:	`df = df.fil lna (fi ll_ dict)`
Ném Ngoại lệ Cho Gán Gía Trị Liên Kết:	`pd.options.mode.chained_assignment = " rai se" df[df[ " col 1"] > 0][" col 2"] = 1`

Truy Cập Dữ Liệu

Truy Cập Nhóm Các Dòng với loc và iloc:	`s.iloc[0]` `s.loc[20]`
pd.Series.between: Chọn Các Dòng Chứa Giá Trị Trong Một Phạm Vi:	`s[s.between(0, 10)]`
pandas.Series.pct_change: Tính Phần Trăm Thay Đổi Giữa Giá Trị Hiện Tại và Trước Đó:	`df.a.pct_change()`
Tính Sự Khác Biệt Giữa Các Dòng của DataFrame:	`df.diff()`
DataFrame.diff và DataFrame.shift: Lấy Sự Khác Biệt Giữa Các Dòng Trong Một Cột:	`diff = df.diff()` `shift = diff.s hif t(-1)`
Chuyển pandas Series thành Mảng NumPy:	`s.values`
df.to_dict: Chuyển DataFrame thành Từ Điển:	`df.to_dict()`

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Truy Cập Dữ Liệu (cont)

Lấy Số Lần Xuất Hiện và Phần Trăm của Một Giá Trị Trong Một Cột:	`size.value_counts()` `size.value_counts(normalize=True)`
pandas.DataFrame.corrwith: Tính Tương Quan Giữa 2 DataFrame:	`df1.corrwith(df2)`
pandas.cut: Chia Giá Trị của DataFrame thành Các Khoảng Rời Rạc:	`pd.cut(df["a"], bins=bins)`
pandas.qcut: Chia Giá Trị của DataFrame thành Các Khoảng Có Số Lượng Phần Tử Bằng Nhau:	`pd.qcut(df["a"], q=3)`
DataFrame.cumsum: Lấy Tổng Cộng Tích Lũy qua Mỗi Cột:	`df.cumsum()`
pandas.DataFrame.cummax: Lấy Giá Trị Tối Đa Tích Lũy:	`nums.cummax()`
Lấy Tổng Của Tất Cả Cột Trong DataFrame:	`df.sum(axis=1)` `df.sum(axis=0)`

Làm Việc với Datetime

Chuyển Đổi Các Cột thành Đối Tượng Thời Gian Khi Sử Dụng pandas để Đọc Các Tệp CSV:	`pd.read_csv("data.csv", parse_ dat es= ["da te_ col umn _1", " dat e_c olu mn_ 2"])`
pandas' DateOffset: Thêm Khoảng Thời Gian vào Một Điểm Thời Gian của pandas:	`ts + DateOf fse t(m ont hs=3) ts + DateOf fse t(y ears=3, hours=3) ts + BD ay(n=6)`
DataFrame rolling: Tính Trung Bình Của n Dữ Liệu Trước Đó Sử Dụng pandas:	`df.rolling(3).mean()`
pandas Grouper: Nhóm Giá Trị Dựa trên Một Tần Suất Cụ Thể:	`df.groupby(pd.Grouper(key="date", freq="1 W")).mean()`
pandas.Series.dt: Truy Cập Các Thuộc Tính Thời Gian của Một Series của pandas:	`df["date"].dt.year df[" dat e"].d t.time`
Lấy Các Dòng trong Một Phạm Vi Năm:	`df.loc["2019":]`
pandas.reindex: Thay Thế Giá Trị Của Các Ngày Thiếu Trong Một Dãy Ngày Bằng 0:	`new_s = s.rein dex (ne w_i ndex, fill_v alue=0)`
Chọn Các Dòng Của DataFrame Trước hoặc Sau Một Ngày Cụ Thể:	`df[df.date <= " 202 1-0 7-2 1"]`
resample: Tái Mẫu Dữ Liệu Chuỗi Thời Gian:	`s.resample('2D').sum()`
Dịch Chỉ Số Của DataFrame Theo Số Kỳ Hạn Cụ Thể:	`df.shift(periods=1)`

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com