

Here's a practical code demonstration for each string method shown in your image, along with sample output and explanations. The variable used will be:

```
python
s = "Hello, World! 123\n"
```

2. Case Conversion Methods

```
python
print(s.lower())           # Converts to lowercase
print(s.upper())           # Converts to uppercase
print(s.title())           # Capitalizes first letter of each
                           word
print(s.capitalize())       # Capitalizes first letter of string
print(s.swapcase())         # Swaps case of each character
print(s.casefold())         # Aggressive lowercase (for
                           comparisons)
Output:
```

```
text
hello, world! 123
```

```
hello, world! 123
```

3. Search & Replace

```
python
print(s.find("World"))          # Returns index of 7
print(s.rfind("l"))            # Returns last index of
                           'l': 11
print(s.index("World"))         # Returns 7 (raises error
                           if not found)
print(s.replace("World", "Earth"))# 'Hello, Earth! 123\n'
print(s.count("l"))             # Counts 'l': 3
Output:
```

```
text
7
11
7
Hello, Earth! 123
```

3

4. Splitting & Joining

```
python
print(s.split())
print(s.rsplit(", ", 1))
Hello', ' World! 123\n']
print(s.splitlines())
print(", ".join(["a", "b"]))
Output:
```

Splits on whitespace

Splits from right: [

[' Hello, World! 123']

Joins: 'a,b'

```
text
['Hello,', 'World!', '123']
['Hello', ' World! 123\n']
['Hello, World! 123']
a,b
```

5. Trimming & Padding

```
python
print(s.strip())
whitespace
print(s.lstrip())
print(s.rstrip())
whitespace
print("cat".center(7, "*"))
'*'*cat**'
print("cat".ljust(6, "-"))
print("cat".rjust(6, "-"))
print("42".zfill(5))
Output:
```

Removes leading/trailing

Removes leading whitespace
Removes trailing

Pads 'cat' to center:

Pads left: 'cat---'

Pads right: '---cat'

Pads zeros: '00042'

```
text
Hello, World! 123
Hello, World! 123
Hello, World! 123
```

```
**cat**  
cat---  
---cat  
00042
```

6. Validation Methods

```
python  
print(s.isalnum())           # False  
print("Hello".isalpha())     # True  
print("123".isdigit())      # True  
print("12.3".isdecimal())    # False  
print("42".isnumeric())      # True  
print("  \n".isspace())       # True  
print("ABC".isupper())       # True  
print("abc".islower())       # True  
print("Hello World".istitle()) # True  
print("foo_bar".isidentifier())# True  
print("abc".isascii())       # True  
print("abc!".isprintable())   # True  
Output:
```

```
text  
False  
True  
True  
False  
True  
True
```

7. Formatting Strings

```
python  
print("Hello, {}".format("Alice"))      # 'Hello, Alice'  
print("{} is {}".format_map({"name": "Bob", "age": 30})) # 'Bob is 30'  
name = "Eve"  
print(f"Hello, {name}")                 # 'Hello, Eve'
```

Output:

```
text
Hello, Alice
Bob is 30
Hello, Eve
```

8. Translation & Mapping

```
python
trans_table = str.maketrans("HW", "hw")
print("Hello World".translate(trans_table)) # 'hello world'
Output:
```

text
hello world

9. Partitioning & Slicing

```
python
print("abc:123:def".partition(":"))      # ('abc', ':',
'123:def')
print("abc:123:def".rpartition(":"))      # ('abc:123', ':',
'def')
print("abcdef"[2:5])                      # 'cde'
Output:
```

```
text
('abc', ':', '123:def')
('abc:123', ':', 'def')
cde
```

This covers all the string methods and functionality listed in your image, with example code and sample output. Let me know if you'd like more detail about any specific method or use case!

Mnp%2FzJ0J%2B91JA9%2F27RqLgmbmq%2B5UniYQerVHmdDRTs85g43dCWPLPnOG
WVWA%2FSQ0BDvE30iQgsWLzI3%2BfjOXhWAoMJKt5OokL9d1TXx1btix6lmeBT%2
FjMowHTXEXMbK5IQDC2k1DcY%2FDy64iHt8OmWxqR8B%2BKkesweBZP05j3yUD7
zvJjIyKHJUmKaNc6RFvtA%2FIi5OsG01aXvs77G4pCtPb2YKqx%2F27vM13KzrZQ2%2
BMkD3LBjFTCsEHoQsBZday1GaMx5PITiMSsDTsUSahPvvZ%2BqS3DE2gXf8PZO7ym
S%2FZ7iWevQSV0b5shYrWv6Yo29302FCvhZQYRRr3rciTTrtWM37Q4ZASxULRL9xU
W7d53pwqTkvd0NUoEdpl5mbRWvzWxHjxjVIF9zsIxI2UIhaCuBzvx9QELDS6XUDPDah
IJbHVwwEGNVXRc0qvpo4WYLNrdriJT82D89MQtFXo8vWGmRTSWIHOvnryAczhnyD
rH4ziJw9IR7OBBeZQW7zs1uqI5bgP6fSshlY6Epf6UO%2FqQnV7gY%2FYN6P55IyJYSgs
h9BIZe3znHGJmrwOngbm%2BHFlagDPT5eK%2BFtLm0xkv84nQ6APnswcyXhDQtB1%
2F8udL5PsmRJsE%2FL%2BqEAwVb9Zlf8tGq%2Blcfmyg1yjIUbqeDAbuWhNC78neKV
6d1OGMNql8UGOpoBHmcnj6VwD%2B9R4uvY7hH9q1yd4UNzQ6V6540nTLj5zkuRK
W8fVPFoxsH5vwk6mxlRFnF4pmhHocxuthKYkIcSWkazIoUnXQxZmzUorUPWpVjAUm
GqqWpF35wvL0a6EpjBh61LtMQ%2FmR3k7dzSP0TUG9jmDMzdbuEA%2Buhjyted0oo8
NNc%2FNwGk9cTf1DQPA1QCKPIvyygPRd%2FOxg%3D%3D&Expires=1755698235

I'll create sample Pandas code snippets for each operation listed in your image. For demonstration, we'll use a DataFrame:

```
python
import pandas as pd

df = pd.DataFrame({
    'A': ['foo', 'bar', 'baz', 'foo', 'qux'],
    'B': [10, 20, 30, 40, 50],
    'C': ['apple', 'orange', 'banana', 'lemon', 'grape'],
    'date': pd.date_range('2024-01-01', periods=5, freq='D')
})
print(df)
Output:
```

```
text
      A      B      C        date
0   foo    10  apple  2024-01-01
1   bar    20 orange  2024-01-02
2   baz    30 banana 2024-01-03
3   foo    40 lemon  2024-01-04
4   qux    50 grape  2024-01-05
```

Data Import

```
python
# pd.read_csv('file.csv')
# pd.read_excel('file.xlsx', sheet_name='Sheet1')
# pd.read_sql(query, connection)
# pd.read_json('file.json')
```

```
# pd.read_parquet('file.parquet')
# (No output since these load data)
```

Data Selection

```
python
print(df['A'])                                # Single column selection
print(df.loc[1, 'B'])                          # Label-based selection
print(df.iloc[2, 1])                           # Integer-based selection
print(df.query('B > 20'))                     # SQL-like filtering
print(df[df['A'].isin(['foo', 'baz'])])        # Multiple values
Output:
```

```
text
0    foo
1    bar
2    baz
3    foo
4    qux
Name: A, dtype: object
```

```
20
```

```
30
```

```
      A    B      C       date
2  baz   30  banana 2024-01-03
3  foo   40  lemon  2024-01-04
4  qux   50  grape  2024-01-05
```

```
      A    B      C       date
0  foo   10  apple 2024-01-01
2  baz   30  banana 2024-01-03
3  foo   40  lemon 2024-01-04
```

Data Manipulation

```
python
print(df.groupby('A').agg({'B': ['mean', 'sum']}))    # Groupby
with aggregation
```

```
df2 = pd.DataFrame({'A': ['foo', 'baz'], 'D': [100, 200]})
print(df.merge(df2, on='A', how='left'))                 # Merge
DataFrames
```

```

print(pd.pivot_table(df, values='B', index='A'))      # Pivot
table

print(df.sort_values('B', ascending=False))           # Sort
values

print(df.melt(id_vars=['A'], value_vars=['B', 'C']))# Melt
DataFrame

print(df.apply(lambda x: x['B']*2, axis=1))          # Apply
function
Output:

```

text

	B	mean	sum
A			
bar	20.0	20	
baz	30.0	30	
foo	25.0	50	
qux	50.0	50	

	A	B	C	date	D
0	foo	10	apple	2024-01-01	100.0
1	bar	20	orange	2024-01-02	NaN
2	baz	30	banana	2024-01-03	200.0
3	foo	40	lemon	2024-01-04	100.0
4	qux	50	grape	2024-01-05	NaN

	B
A	
bar	20
baz	30
foo	25
qux	50

	A	B	C	date
4	qux	50	grape	2024-01-05
3	foo	40	lemon	2024-01-04
2	baz	30	banana	2024-01-03
1	bar	20	orange	2024-01-02
0	foo	10	apple	2024-01-01

A	variable	value
---	----------	-------

```
0  foo        B      10
1  bar        B      20
2  baz        B      30
3  foo        B      40
4  qux        B      50
5  foo        C  apple
6  bar        C orange
7  baz        C banana
8  foo        C lemon
9  qux        C grape
```

```
0      20
1      40
2      60
3      80
4     100
dtype: int64
```

Data Cleaning

```
python
print(df.dropna(subset=[ 'B' ], how='any'))          #
Drop NA in 'B'
print(df.fillna(method='ffill'))                      #
Forward fill NAs
print(df.replace({ 'A': { 'foo': 'FOO' } }))        #
Replace values
print(df.astype({ 'A': 'category' }))                 #
Convert dtype
print(df.interpolate(method='linear'))                #
Interpolate
(Assuming some missing values for practical example)
```

String Operations

```
python
print(df[ 'C' ].str.contains('a'))                  # Contains
pattern
print(df[ 'C' ].str.extract('(\w+)'))              # Extract pattern
print(df[ 'C' ].str.split('a').str[0])             # Split and take
first
print(df[ 'C' ].str.lower())                        # Lowercase
print(df[ 'C' ].str.strip())                         # Strip spaces
print(df[ 'C' ].str.replace(r'\s+', '_'))           # Replace spaces
```

Output:

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

```
0
0    apple
1  orange
2 banana
3   lemon
4   grape
```

```
0    apple
1    or
2    b
3   lemon
4   gr
Name: C, dtype: object
```

```
0    apple
1  orange
2 banana
3   lemon
4   grape
Name: C, dtype: object
```

```
0    apple
1  orange
2 banana
3   lemon
4   grape
Name: C, dtype: object
```

```
0    apple
1  orange
2 banana
3   lemon
4   grape
Name: C, dtype: object
```

Statistics

```
python
print(df.describe())                                # Summary
stats
print(df['B'].agg(['mean', 'median', 'std']))      # Aggregate
print(df['A'].value_counts(normalize=True))         # Value
counts
print(df.corr(method='pearson'))                   # Correlation
print(df.cov())
matrix
print(df['B'].quantile([0.25, 0.5, 0.75]))       # Quantiles
Output:
```

text

```
          B
count    5.000000
mean     30.000000
std      15.811388
min     10.000000
25%    20.000000
50%    30.000000
75%    40.000000
max     50.000000

mean        30.0
median      30.0
std         15.811388
Name: B, dtype: float64
```

```
foo      0.4
bar      0.2
baz      0.2
qux      0.2
Name: A, dtype: float64
```

```
          B
B    1.0
```

```
          B
B   250.0
```

```
0.25    20.0
```

```
0.50      30.0  
0.75      40.0  
Name: B, dtype: float64
```

Time Series

```
python
print(df.resample('D', on='date').mean())          # Resample
print(df.rolling(window=2).mean())                  # Rolling
mean
print(df.shift(periods=1))                         # Shift
print(pd.date_range('2024', periods=3, freq='M')) # Date range
print(df.asfreq('D', method='ffill'))              # Asfreq,
fill
print(df['date'].dt.strftime('%Y-%m-%d'))         # Datetime
formatting
```

Data Export

```
python
# df.to_csv('output.csv', index=False)
# df.to_excel('output.xlsx', sheet_name='Sheet1')
# df.to_parquet('output.parquet')
# df.to_json('output.json', orient='records')
# (No output since these save files)
```

Advanced Features

```
python
print(df.pipe(lambda d: d.head(2)))                      # Method
chaining
print(df.eval('B + 2'))                                     #
Expression eval
print(df.memory_usage(deep=True))                           # Memory
usage
print(df.select_dtypes(include=['number']))                  # Select
numeric columns
print(df.nlargest(3, 'B'))                                  # Top
values
df2 = pd.DataFrame({'A': ['foo', 'bar'], 'L': [[1,2],[3,4]]})
print(df2.explode('L'))                                    # Expand
list column
Output:
```

```
text
      A    B        C        date
0   foo   10  apple  2024-01-01
1   bar   20 orange  2024-01-02
```

0	12
1	22
2	32
3	42
4	52

dtype: int64

```
Index      128
A          29
B          40
C          29
date       80
dtype: int64
```

	B
0	10
1	20
2	30
3	40
4	50

	A	B	C	date
4	qux	50	grape	2024-01-05
3	foo	40	lemon	2024-01-04
2	baz	30	banana	2024-01-03

	A	L
0	foo	1
0	foo	2
1	bar	3
1	bar	4

Each line shows the code snippet, its operation, and what the output means as an example on realistic data. If you want longer output or a specific focus, let me know!

2F%2F%2F%2F%2F8BEAEaDDY5OTc1MzMwOTcwNSIM0K7mXuDRI8xhm68zKs
4Eg2AyFgsrv80j9EBI%2BEPwMJ9V09mDEOUHg0sVDb5fQEEIZowEom07w2U3Olaibz
xPaAEmYRmQsEiEgdOJbdDoX0CklOngRIM2bAT%2BL98pj96YhA0W533SFbf2RxiF
MZ8exPmUry%2BVXPO6WliCnfPITbH5d3lRCt2HMydDUzJFEBetewvuNHCKtnQJrhk
xvQegrw%2BcyPhFGcyh3PTeJAwgjVUgr0ZrQziKejgPofNz9u2geNZlgDqc6QnhBoMTltj
KbvPkpx5GE%2F6LDUYUqCqsg%2BDhF3gWgfuQLOx5qse7EJNAAs%2B0CWT%2B
m42V8dz%2FN49Zf02gDOpLB9Tc04ZwFCc5QDpMekNRZE17jFPhBbMij%2F2YK5Vfl
Mgzp%2Ff5PCBMLckIzz9fgYWLJs582sJRZT2SMJumtLN%2BNVrkmxHoCM0P%2Fkw
6ZYm6lJ%2BhRh5dkm0Ofwnt5fb%2F8JiaJddIsVvvuJ%2F3aVB%2BysU2t1pR9dQ4Uv4
M0XKh5A%2BsRgbyfqRhrxtI201dplONY8%2F18iGJKYa1eLZKNvqChEs516rVuTwuQF
maZeuf8W%2BsSVjKSkb%2FKKsRfr7Emr%2BtaUktjsqHzbs%2FAgOonEDTRrZaQ8N6
xibHARvdKv0qcTHTTr8k3hdnGB%2FIm7HF3YjHkwyET2Kk0Vk2uuQQftoCndR5XfwT5
%2FjAY5JfaI9ff%2BZsK0DtNrdEFBnOl3S4OmuaJNbrDux6aBoRY%2FusXta7ysYCqKC
%2BWjFAnXz3Chs1zli%2B9DLEXdP4SdY0slkBIMULjupJhlCOSm9AQw5qKXxQY6m
wFrB8p9%2BliFrB SCHjXZBH3sPdXB1YHwIxsLIrQIo6tqE5XKhQawVyanjRVM82EdTe6
IVhbA5wozyailoBvjttFnrYnVI20L5o2aEQjnqKJusHvrRx8ol5X8sobSit6ZvUWvcKJBMaF
AbM8O9HRnyN6ZOGvO85n7PJOn%2Fd2mNPY4WYxy09OjAIRZY2oKG%2BWKAWP
SyCESLonGxD7RoA%3D%3D&Expires=1755698606