Python
WEEK 9
Pandas

AI Academy

COMPUTER PROGRAMMING WITH PYTHON

Instructor - James E. Robinson, III

Teaching Assistant - Travis Martin

LIGHTNING REVIEW

- Variables
- Input / Output
- Expressions
- Functions
- Conditional Control
- Looping
- Data Types
- Logging
- Functions
- Scope
- Recursion
- Decorator
- Recursion
- Dynamic Prg
- Exceptions
- Classes
- Objects
- Encapsulation
- Public v/s Private
- Dunder Methods
- Instances
- Inheritance
- Types of Inheritance
- Polymorphism
- Method Overriding
- Queue
- Stacks
- Graphs
- Trees
- Binary Trees
- Traversal Methods
- Searching
- Files
- Opening / Closing
- Reading / Writing
- Context Managers
- File Exceptions
- Handling Exceptions
- File Formats
- Numpy
- Ndarray

TOPICS COVERED

- Pandas
- Series
  - Creation
  - Indexing
  - Modification
- Dataframe
  - Creation
  - Attributes
  - Functions
  - Selection
  - Apply Function
  - Missing Data

NEW MODULE INTRODUCTION

PANDAS

PANDAS

PANEL DATA - PYTHON LIBRARY USED FOR HANDLING DATA FROM MEASUREMENTS OVER TIME

Pandas
Pandas was created to manipulate data that includes mixed data types, custom indexing, missing and inconsistent data. It allows you to shape and manipulate such data into forms suitable for analysis.

Pandas is built on NumPy and as such it can handle large data sets efficiently.

FUN FACT
One of the larger datasets Pandas has produced is its documentation - a 2000 page PDF

SERIES

ONE-DIMENSIONAL NDARRAY WITH AXIS LABELS

The type() of series in pandas is pandas.core.series.Series

Characteristics
Pandas Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called index.

The object supports both integer and label-based indexing.

SERIES - CREATION

BY IMPORTING PANDAS MODULE

Example

```
import pandas as pd
s = pd.Series([10,20,30])
print(s)
print(type(s))
```

OUTPUT

```
0   10
1   20
2   30
dtype: int64
<class 'pandas.core.series.Series'>
```

Example

```
import pandas as pd
s = pd.Series([10,20,30],index=['A','B','C'],name="Values")
print(s)
print(type(s))
```

OUTPUT

```
A   10
B   20
C   30
Name: Values, dtype: int64
<class 'pandas.core.series.Series'>
```

SERIES - INDEXING

BY LABELS AND INDEXES

Example

```
import pandas as pd
s = pd.Series([40, 50, 60])
s.index = ['A','B','C']
s.name = 'Values'
print(s,"\n")
print(s['B'],"\n")
print(s[1:],"\n")
print(s[-1],"\n")
```

OUTPUT

```
A    40
B    50
C    60
Name: Values, dtype: int64

50

B    50
C    60
Name: Values, dtype: int64

60
```

SERIES - MODIFICATION

FUNCTIONS THAT MODIFY A SERIES DATA

update
Updates values in a series

rename
Renames indexes of series

replace
Replaces particular value in series

Example

```
import pandas as pd
s = pd.Series([40, 50, 60])
print(s)
s.update(pd.Series([10, 20, 30]))
s.index = ['A','B','C']
s.name = 'Values'
print(s)
s.rename({'A': 'X'},inplace=True)
print(s)
s.replace(20,15,inplace=True)
print(s)
```

OUTPUT

```
0  40
1  50
2  60
dtype: int64
A  10
B  20
C  30
Name: Values, dtype: int64
X  10
B  20
C  30
Name: Values, dtype: int64
X  10
B  15
C  30
Name: Values, dtype: int64
```

DATAFRAME

TWO-DIMENSIONAL SIZE-MUTABLE, POTENTIALLY HETEROGENEOUS TABULAR DATA STRUCTURE

The type() of dataframe in pandas is pandas.core.frame.DataFrame

Characteristics

Data is aligned in a tabular fashion in rows and columns.

- The column names should be non-empty.
- The row names should be unique.
- The data stored in a data frame can be of numeric, factors, or character type.
- Each column should contain same number of data items.

Like the Series object, supports custom indexing. Additionally, supports column labels.

DATAFRAME - CREATION

BY IMPORTING PANDAS MODULE

Example

```
import pandas as pd
df = pd.DataFrame([10,20,30])
print(df)
print(type(df))
```

OUTPUT

```
    0
0  10
1  20
2  30
<class 'pandas.core.frame.DataFrame'>
```

Example

```
import pandas as pd
df = pd.DataFrame({'Name':['A','B','C'],
                   'Age':[10,20,30]})
print(df)
print(type(df))
```

OUTPUT

```
  Name  Age
0    A   10
1    B   20
2    C   30
<class 'pandas.core.frame.DataFrame'>
```

DATAFRAME - CREATION

BY IMPORTING PANDAS MODULE

Example

```
df = pd.DataFrame(
{
        "A": 1.0,
        "B": pd.Timestamp("20220102"),
        "C": pd.Series(1, index=list(range(4)), dtype="float32"),
        "D": np.array([3] * 4, dtype="int32"),
        "E": pd.Categorical(["test", "train", "test", "train"]),
        "F": "foo"
 }) # note all arrays must be same size
print(df)
```

OUTPUT

	A	B	C	D	E	F
0	1.0	2022-01-02	1.0	3	test	foo
1	1.0	2022-01-02	1.0	3	train	foo
2	1.0	2022-01-02	1.0	3	test	foo
3	1.0	2022-01-02	1.0	3	train	foo

DATAFRAME - ATTRIBUTES

FUNCTIONS THAT SHOW THE META VALUES OF THE DATAFRAME

dtypes
Data type of each column
index
Displays indexes of dataframe
columns
Shows all column names
size
Number of elements
shape
Dimensions of a dataframe

Example

print(df.dtypes)

print(df.index)

print(df.columns)

print(df.size)

print(df.shape)

OUTPUT

```
A     float64
B       int64
C     float32
D       int32
E    category
F      object
dtype: object
Int64Index([0, 1, 2, 3], dtype='int64')
Index(['A', 'B', 'C', 'D', 'E', 'F'], dtype='object')
24
(4, 6)
```

DATAFRAME - FUNCTIONS

FUNCTIONS THAT HELP UNDERSTAND A DATAFRAME

Example DF

	A	B	C	D	E	F
a	5	-6	15	0.5	1.1	-20
b	4	-2	16	1.0	1.2	-18
c	2	-4	17	2.0	1.3	-16
d	7	-10	11	2.5	1.4	-14
e	8	-3	14	1.5	1.5	-12
f	1	-8	19	4.5	1.6	-10
g	3	-9	18	3.5	1.7	-8
h	9	-5	13	3.0	1.8	-6
i	6	-7	12	4.0	1.9	-4

Top values of dataframe

df.head()

	A	B	C	D	E	F
a	5	-6	15	0.5	1.1	-20
b	4	-2	16	1.0	1.2	-18
c	2	-4	17	2.0	1.3	-16
d	7	-10	11	2.5	1.4	-14
e	8	-3	14	1.5	1.5	-12

Bottom values of dataframe

df.tail()

	A	B	C	D	E	F
e	8	-3	14	1.5	1.5	-12
f	1	-8	19	4.5	1.6	-10
g	3	-9	18	3.5	1.7	-8
h	9	-5	13	3.0	1.8	-6
i	6	-7	12	4.0	1.9	-4

DATAFRAME - FUNCTIONS

FUNCTIONS THAT HELP UNDERSTAND A DATAFRAME

Example DF

	A	B	C	D	E	F
a	5	-6	15	0.5	1.1	-20
b	4	-2	16	1.0	1.2	-18
c	2	-4	17	2.0	1.3	-16
d	7	-10	11	2.5	1.4	-14
e	8	-3	14	1.5	1.5	-12
f	1	-8	19	4.5	1.6	-10
g	3	-9	18	3.5	1.7	-8
h	9	-5	13	3.0	1.8	-6
i	6	-7	12	4.0	1.9	-4

Description of data

df.describe()

	A	B	C	D	E	F
count	9.000000	9.000000	9.000000	9.000000	9.000000	9.000000
mean	5.000000	-6.000000	15.000000	2.500000	1.500000	-12.000000
std	2.738613	2.738613	2.738613	1.369306	0.273861	5.477226
min	1.000000	-10.000000	11.000000	0.500000	1.100000	-20.000000
25%	3.000000	-8.000000	13.000000	1.500000	1.300000	-16.000000
50%	5.000000	-6.000000	15.000000	2.500000	1.500000	-12.000000
75%	7.000000	-4.000000	17.000000	3.500000	1.700000	-8.000000
max	9.000000	-2.000000	19.000000	4.500000	1.900000	-4.000000

DATAFRAME - FUNCTIONS

FUNCTIONS THAT HELP MODIFY DATAFRAME

Example DF

	A	B	C	D	E	F
a	5	-6	15	0.5	1.1	-20
b	4	-2	16	1.0	1.2	-18
c	2	-4	17	2.0	1.3	-16
d	7	-10	11	2.5	1.4	-14
e	8	-3	14	1.5	1.5	-12
f	1	-8	19	4.5	1.6	-10
g	3	-9	18	3.5	1.7	-8
h	9	-5	13	3.0	1.8	-6
i	6	-7	12	4.0	1.9	-4

Transpose of data

df.transpose()

	a	b	c	d	e	f	g	h	i
A	5.0	4.0	2.0	7.0	8.0	1.0	3.0	9.0	6.0
B	-6.0	-2.0	-4.0	-10.0	-3.0	-8.0	-9.0	-5.0	-7.0
C	15.0	16.0	17.0	11.0	14.0	19.0	18.0	13.0	12.0
D	0.5	1.0	2.0	2.5	1.5	4.5	3.5	3.0	4.0
E	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9
F	-20.0	-18.0	-16.0	-14.0	-12.0	-10.0	-8.0	-6.0	-4.0

DATAFRAME - FUNCTIONS

FUNCTIONS THAT HELP MODIFY DATAFRAME

Example DF

	A	B	C	D	E	F
a	5	-6	15	0.5	1.1	-20
b	4	-2	16	1.0	1.2	-18
c	2	-4	17	2.0	1.3	-16
d	7	-10	11	2.5	1.4	-14
e	8	-3	14	1.5	1.5	-12
f	1	-8	19	4.5	1.6	-10
g	3	-9	18	3.5	1.7	-8
h	9	-5	13	3.0	1.8	-6
i	6	-7	12	4.0	1.9	-4

Sort by index

df.sort_index(axis=1, ascending=False)

	F	E	D	C	B	A
a	-20	1.1	0.5	15	-6	5
b	-18	1.2	1.0	16	-2	4
c	-16	1.3	2.0	17	-4	2
d	-14	1.4	2.5	11	-10	7
e	-12	1.5	1.5	14	-3	8
f	-10	1.6	4.5	19	-8	1
g	-8	1.7	3.5	18	-9	3
h	-6	1.8	3.0	13	-5	9
i	-4	1.9	4.0	12	-7	6

Sort by value

df.sort_values(by="A")

	A	B	C	D	E	F
f	1	-8	19	4.5	1.6	-10
c	2	-4	17	2.0	1.3	-16
g	3	-9	18	3.5	1.7	-8
b	4	-2	16	1.0	1.2	-18
a	5	-6	15	0.5	1.1	-20
i	6	-7	12	4.0	1.9	-4
d	7	-10	11	2.5	1.4	-14
e	8	-3	14	1.5	1.5	-12
h	9	-5	13	3.0	1.8	-6

DATAFRAME - SELECTION

FUNCTIONS TO SELECT DATA FROM DATAFRAME

Example DF

	A	B	C	D	E	F
a	5	-6	15	0.5	1.1	-20
b	4	-2	16	1.0	1.2	-18
c	2	-4	17	2.0	1.3	-16
d	7	-10	11	2.5	1.4	-14
e	8	-3	14	1.5	1.5	-12
f	1	-8	19	4.5	1.6	-10
g	3	-9	18	3.5	1.7	-8
h	9	-5	13	3.0	1.8	-6
i	6	-7	12	4.0	1.9	-4

loc - selection by Label

df.loc['d']

```
A    7.0
B   -10.0
C    11.0
D    2.5
E    1.4
F   -14.0
Name: d, dtype: float64
```

iloc - selection by Position

df.iloc[3]

```
A    7.0
B   -10.0
C    11.0
D    2.5
E    1.4
F   -14.0
Name: d, dtype: float64
```

at - explicit selection by label

df.at['d','C']

```
11.0
```

iat - explicit selection by Position

df.iat[3,2]

```
11.0
```

DATAFRAME - SELECTION

FUNCTIONS TO FILTER CERTAIN DATA

Example DF

	A	B	C	D	E	F
a	5	-6	15	0.5	1.1	-20
b	4	-2	16	1.0	1.2	-18
c	2	-4	17	2.0	1.3	-16
d	7	-10	11	2.5	1.4	-14
e	8	-3	14	1.5	1.5	-12
f	1	-8	19	4.5	1.6	-10
g	3	-9	18	3.5	1.7	-8
h	9	-5	13	3.0	1.8	-6
i	6	-7	12	4.0	1.9	-4

Boolean Indexing

df[df>0]

	A	B	C	D	E	F
a	5	NaN	15	0.5	1.1	NaN
b	4	NaN	16	1.0	1.2	NaN
c	2	NaN	17	2.0	1.3	NaN
d	7	NaN	11	2.5	1.4	NaN
e	8	NaN	14	1.5	1.5	NaN
f	1	NaN	19	4.5	1.6	NaN
g	3	NaN	18	3.5	1.7	NaN
h	9	NaN	13	3.0	1.8	NaN
i	6	NaN	12	4.0	1.9	NaN

Boolean Indexing

df[df['A']>5]

	A	B	C	D	E	F
d	7	-10	11	2.5	1.4	-14
e	8	-3	14	1.5	1.5	-12
h	9	-5	13	3.0	1.8	-6
i	6	-7	12	4.0	1.9	-4

Boolean Indexing

df[df['C']%2 == 0]

	A	B	C	D	E	F
b	4	-2	16	1.0	1.2	-18
e	8	-3	14	1.5	1.5	-12
g	3	-9	18	3.5	1.7	-8
i	6	-7	12	4.0	1.9	-4

DATAFRAME - SELECTION

FUNCTIONS TO SELECT DATA FROM DATAFRAME

Example DF

	A	B	C	D	E	F
a	5	-6	15	0.5	1.1	-20
b	4	-2	16	1.0	1.2	-18
c	2	-4	17	2.0	1.3	-16
d	7	-10	11	2.5	1.4	-14
e	8	-3	14	1.5	1.5	-12
f	1	-8	19	4.5	1.6	-10
g	3	-9	18	3.5	1.7	-8
h	9	-5	13	3.0	1.8	-6
i	6	-7	12	4.0	1.9	-4

isin()

Returns truth value after comparing with dataframe

df['C'].isin([16,18,12])

a	False
b	True
c	False
d	False
e	False
f	False
g	True
h	False
i	True

Using isin() to filter dataframe

df[df['C'].isin([16,18,12])]

	A	B	C	D	E	F
b	4	-2	16	1.0	1.2	-18
g	3	-9	18	3.5	1.7	-8
i	6	-7	12	4.0	1.9	-4

DATAFRAME - APPLY

APPLY FUNCTION TO EACH COLUMN OF DATAFRAME

Example DF

	A	B	C	D	E	F
a	5	-6	15	0.5	1.1	-20
b	4	-2	16	1.0	1.2	-18
c	2	-4	17	2.0	1.3	-16
d	7	-10	11	2.5	1.4	-14
e	8	-3	14	1.5	1.5	-12
f	1	-8	19	4.5	1.6	-10
g	3	-9	18	3.5	1.7	-8
h	9	-5	13	3.0	1.8	-6
i	6	-7	12	4.0	1.9	-4

Lambda Functions

Example

df.apply(lambda x: x.max()-x.min())

```
A    8.0
B    8.0
C    8.0
D    4.0
E    0.8
F   16.0
dtype: float64
```

Numpy Functions

Example

df.apply(np.cumsum)

	A	B	C	D	E	F
a	5	-6	15	0.5	1.1	-20
b	9	-8	31	1.5	2.3	-38
c	11	-12	48	3.5	3.6	-54
d	18	-22	59	6.0	NaN	-68
e	26	-25	73	7.5	5.1	-80
f	27	-33	92	12.0	6.7	-90
g	30	-42	110	15.5	8.4	-98
h	39	-47	123	18.5	10.2	-104
i	45	-54	135	22.5	12.1	-108

DATAFRAME - MISSING DATA

HANDLING MISSING DATA IN DATAFRAME

Example DF

	A	B	C	D	E	F
a	5	-6	15.0	0.5	1.1	-20.0
b	4	-2	NaN	1.0	1.2	NaN
c	2	-4	NaN	2.0	1.3	NaN
d	7	-10	11.0	2.5	1.4	-14.0
e	8	-3	NaN	1.5	1.5	NaN
f	1	-8	19.0	4.5	1.6	-10.0
g	3	-9	18.0	3.5	1.7	-8.0
h	9	-5	NaN	3.0	NaN	-6.0
i	6	-7	12.0	4.0	1.9	-4.0

dropna()

Drop rows with any NaN values

df.dropna()

	A	B	C	D	E	F
a	5	-6	15.0	0.5	1.1	-20.0
d	7	-10	11.0	2.5	1.4	-14.0
f	1	-8	19.0	4.5	1.6	-10.0
g	3	-9	18.0	3.5	1.7	-8.0
i	6	-7	12.0	4.0	1.9	-4.0

fillna()

Replace NaN with value

df.fillna(999)

	A	B	C	D	E	F
a	5	-6	15.0	0.5	1.1	-20.0
b	4	-2	999.0	1.0	1.2	999.0
c	2	-4	999.0	2.0	1.3	999.0
d	7	-10	11.0	2.5	1.4	-14.0
e	8	-3	999.0	1.5	1.5	999.0
f	1	-8	19.0	4.5	1.6	-10.0
g	3	-9	18.0	3.5	1.7	-8.0
h	9	-5	999.0	3.0	999.0	-6.0
i	6	-7	12.0	4.0	1.9	-4.0

DATAFRAME - MISSING DATA

HANDLING MISSING DATA IN DATAFRAME

Example DF

	A	B	C	D	E	F
a	5	-6	15.0	0.5	1.1	-20.0
b	4	-2	NaN	1.0	1.2	NaN
c	2	-4	NaN	2.0	1.3	NaN
d	7	-10	11.0	2.5	1.4	-14.0
e	8	-3	NaN	1.5	1.5	NaN
f	1	-8	19.0	4.5	1.6	-10.0
g	3	-9	18.0	3.5	1.7	-8.0
h	9	-5	NaN	3.0	NaN	-6.0
i	6	-7	12.0	4.0	1.9	-4.0

isna()

Boolean df result showing which items have NaN

pd.isna(df)

	A	B	C	D	E	F
a	False	False	False	False	False	False
b	False	False	True	False	False	True
c	False	False	True	False	False	True
d	False	False	False	False	False	False
e	False	False	True	False	False	True
f	False	False	False	False	False	False
g	False	False	False	False	False	False
h	False	False	True	False	True	False
i	False	False	False	False	False	False

WEEK SUMMARY

- Learned the module Pandas
- Learned concept of Series and Dataframes
- Learned basic attributes and functions of pandas
- Learned to modify series and dataframes
- Understood selection techniques
- Learned to apply functions of dataframe
- Learned to handle missing data

THANK YOU

FOR ADDITIONAL QUERIES OR DOUBTS
CONTACT:
jerobins@ncsu.edu

AI Academy