

# Python

## Deep Learning



- open source software tools for developing deep neural networks are called frameworks.
- The two most popular frameworks are TensorFlow and PyTorch.
- Both frameworks are most commonly accessed through Python.
- Python is a high-level, interpreted, general-purpose programming language.
- We will cover the key Python concepts most used in deep learning.
- We also cover two important Python packages: NumPy and Pandas



- Variables are dynamically typed.
- Enter an integer, it is typed as an integer. Add a decimal it is typed as float.

```
a = 2  
b = 3  
c = a + b  
print(c)
```

---

5



```
print(a/b)
```

0

```
d = 3.0  
print(a/d)
```

0.6666666666666666



```
a = [2, 4, 6, 8]  
print(a[0])
```

2

```
print(a[-1])  # neg. index cnts from right
```

8

```
a[1:3] = [] # remove elements 1 and 2  
print(a)
```

[2, 8]



```
a = 2, 4, 6, 8  
print(a)
```

```
(2, 4, 6, 8)
```

```
print(a[0])
```

```
2
```

```
a[0] = 9
```

```
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: 'tuple' object does not support  
    item assignment
```



```
dict = {'input': [1, 3, 7], 'target': [3, 7, 15]}  
print(dict['input'])
```

---

```
[1, 3, 7]
```

```
print(dict.items())
```

---

```
[('input', [1, 3, 7]), ('target', [3, 7, 15])]
```

```
print(dict.keys())
```

---

```
['input', 'target']
```



```
x = -5
if x<0:
    y = -1
elif x>0:
    y = 1
else:
    y = 0
print(y)
```

---

-1





# While loop

```
a = [1, 2, 3, 4]
i, x = 0, 0  #Multiple assignment
while i < len(a):
    x = x + a[i]
    i = i + 1

print(x)
```

---

10



```
x = 0
for num in a:
    x = x + num
```

```
print(x)
```

---

10

```
a = ['one', 'two', 'three', 'four']
for q in a:
    print(q)
```

---

one

two

three

four



# For loop with list comprehension

```
a = [1, 2, 3, 4]
b = [x**2 for x in a]
print(b)
```

---

```
[1, 4, 9, 16]
```



# Logical operators

```
x, y = True, False  
print(x and y)  
print(x or y)  
print(not x)
```

---

```
False  
True  
False
```



```
def f(y,q):  
    return [y*y, 1/q]
```

```
zw = f(4.0,2.0)  
print(zw)
```

---

```
[16.0, 0.5]
```



# Iterators and generators

```
def bytwo(x):  
    n = len(x)  
    for i in range(0, n-1, 2):  
        yield x[i:i+2]
```

```
a=[1, 2, 3, 4]  
zz = bytwo(a)  
for qq in zz:  
    print(qq)
```

---

[1, 2]

[3, 4]



File `Logic.py`:

```
def a(x,y):  
    print(x and y)  
  
def o(x,y):  
    print(x or y)
```

We can call this module's functions:

```
import logic  
print(logic.a(True,False))  
print(logic.o(True,False))
```

---

```
False  
True
```



```
class simplenet:  
    def __init__(self,weight,bias):  
        self.w = weight  
        self.b = bias  
  
    def sim(self,p):  
        return self.w*p + self.b
```

```
net = simplenet(4.0,2.0)  
print(net.sim(3.0))
```

---

14.0





- NumPy is a module for scientific computing (like MATLAB) in Python.
- It works well with the deep learning frameworks.
- The key object in NumPy is the multidimensional array (tensor).



```
import numpy as np
x = np.array([[1, 2, 3],[4, 5, 6]])
print(x)
print(x.ndim)
print(x.shape)
print(x.dtype)
```

---

```
[[1 2 3]
 [4 5 6]]
2
(2, 3)
int64
```



```
a = np.arange(6)
print(a)
b = a.reshape(2,3)
print(b)
c = np.arange(0,12,2).reshape(2,3)
print(c)
d = np.arange(0,24,4).reshape(3,2)
print(d)
```

---

```
[0 1 2 3 4 5]
[[0 1 2]
 [3 4 5]]
[[ 0  2  4]
 [ 6  8 10]]
[[ 0  4]
 [ 8 12]
 [16 20]]
```



## Hadamard multiplication

```
print(b*c)
```

---

```
[[ 0  2  8]  
 [18 32 50]]
```

## Standard matrix multiplication

```
print(np.matmul(c,d))
```

---

```
[[ 80 104]  
 [224 320]]
```



# Addition and broadcasting

```
e = np.arange(3)
print(e)
print(b+c)
print(b+e)
```

---

```
[0 1 2]
[[ 0  3  6]
 [ 9 12 15]]
[[0 2 4]
 [3 5 7]]
```



```
print(b[1, 2])  
print(b[0])  
print(b[[0, 1],[1, 2]])
```

---

```
5  
[0 1 2]  
[1 5]
```

```
print(a[0:5:2])  
print(a[:5])  
print(a[-4:])
```

---

```
[0 2 4]  
[0 1 2 3 4]  
[2 3 4 5]
```



# Sums and products across dimensions

```
print(np.sum(b))  
print(np.sum(b,axis=0))  
print(np.prod(b,axis=1))
```

---

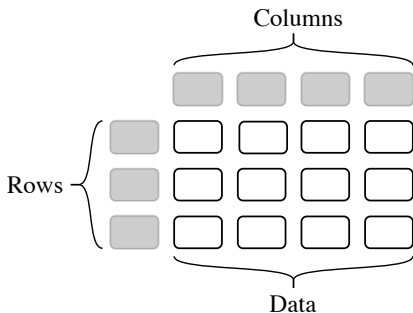
15

[3 5 7]

[ 0 60]



- Much deep learning workflow is devoted to data wrangling.
- This includes loading, formatting and preprocessing data.
- Pandas is a Python module for data wrangling.
- The main data structure of Pandas is DataFrame.





# Panda can load many data types

- Comma-separated values (CSV)
- XLSX
- ZIP
- Plain Text (txt)
- JSON
- XML
- HTML
- HTML
- Images
- Hierarchical Data Format
- PDF
- DOCX
- MP3
- MP4
- SQL



## Reading CSV file

```
import pandas as pd  
sample_df = pd.read_csv('SampleDF.csv')
```

```
print(sample_df.shape)
```

---

```
(100, 7)
```

```
print(sample_df.columns)
```

---

```
Index(['Patient', 'Weeks', 'FVC', 'Percent',  
      'Age', 'Sex', 'SmokingStatus'],  
      dtype='object')
```



# Dataframe description

```
print(sample_df.describe())
```

	Weeks	FVC	Percent	Age
count	100.00000	100.000000	100.000000	100.000000
mean	35.89000	2759.820000	78.187965	67.110000
std	24.90868	925.766484	21.094723	6.738844
min	0.00000	969.000000	43.352279	49.000000
25%	16.50000	2118.000000	62.821569	64.000000
50%	32.00000	2597.500000	73.989508	68.000000
75%	49.00000	3267.000000	89.005946	72.000000
max	116.00000	5768.000000	153.145378	87.000000



```
sample_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 100 entries, 0 to 99
```

```
Data columns (total 7 columns):
```

#	Column	Non-Null Count	Dtype
0	Patient	100 non-null	object
1	Weeks	100 non-null	int64
2	FVC	100 non-null	int64
3	Percent	100 non-null	float64
4	Age	100 non-null	int64
5	Sex	100 non-null	object
6	SmokingStatus	100 non-null	object

```
dtypes: float64(1), int64(3), object(3)
```

```
memory usage: 5.6+ KB
```



```
print(sample_df.head())
```

	Patient	Weeks	FVC	Percent	Age	Sex	\
0	ID00213	32	2972	81.828194	70	Male	
1	ID00129	0	2253	59.622102	71	Male	
2	ID00130	12	1648	68.116062	65	Female	
3	ID00225	23	969	49.075715	77	Female	
4	ID00082	33	2885	98.666211	49	Female	

	SmokingStatus
0	Currently smokes
1	Never smoked
2	Never smoked
3	Never smoked
4	Currently smokes



## Select a column (series)

```
fvc = sample_df['FVC']  
fvc.describe()
```

---

count	100.000000
mean	2759.820000
std	925.766484
min	969.000000
25%	2118.000000
50%	2597.500000
75%	3267.000000
max	5768.000000

Name: FVC, dtype: float64



## Select multiple columns or certain rows

```
twocol = sample_df[['Age', 'FVC']]  
print(twocol.head())
```

---

	Age	FVC
0	70	2972
1	71	2253
2	65	1648
3	77	969
4	49	2885

```
older = sample_df[sample_df['Age']>75]  
print(older.shape)
```

---

```
(8, 7)
```



```
older_fvc = sample_df.loc[sample_df['Age']  
                           ]>75, ['Age', 'FVC']  
print(older_fvc.head())
```

---

	Age	FVC
3	77	969
7	83	3171
16	87	2220
59	77	1550
82	77	1818

```
sample_subset = sample_df.iloc[2:5, 1:3]  
print(sample_subset.head())
```

---

	Weeks	FVC
2	12	1648
3	23	969
4	33	2885





```
pivoted = sample_df.pivot(index='Patient', values = 'FVC', columns='
    Weeks')
print(pivoted.iloc[:5, :10])
```

Weeks	0	2	4	5	6	7	9	10	12	13
Patient										
ID000116	NaN	NaN	NaN	NaN	NaN	NaN	3541.0	NaN	NaN	3410.0
ID000156	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
ID000206	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
ID000276	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2472.0	NaN
ID000306	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN



```
melted = sample_df.melt(id_vars='Patient',  
                        value_vars='Age')  
print(melted.head())
```

	Patient	variable	value
0	ID00213637202257692916109	Age	70
1	ID00129637202219868188000	Age	71
2	ID00130637202220059448013	Age	65
3	ID00225637202259339837603	Age	77
4	ID00082637202201836229724	Age	49



# Operations on Dataframes

```
print(older_fvc.apply(np.mean, axis=0))
```

---

```
Age      79.125  
FVC      2284.500  
dtype: float64
```

```
print(older_fvc.apply(lambda x: x.max() - x.min(),  
                    axis=0))
```

---

```
Age      11  
FVC      3081  
dtype: int64
```

```
print(older_fvc.min(axis=0))
```

---

```
Age      76  
FVC      969  
dtype: int64
```

