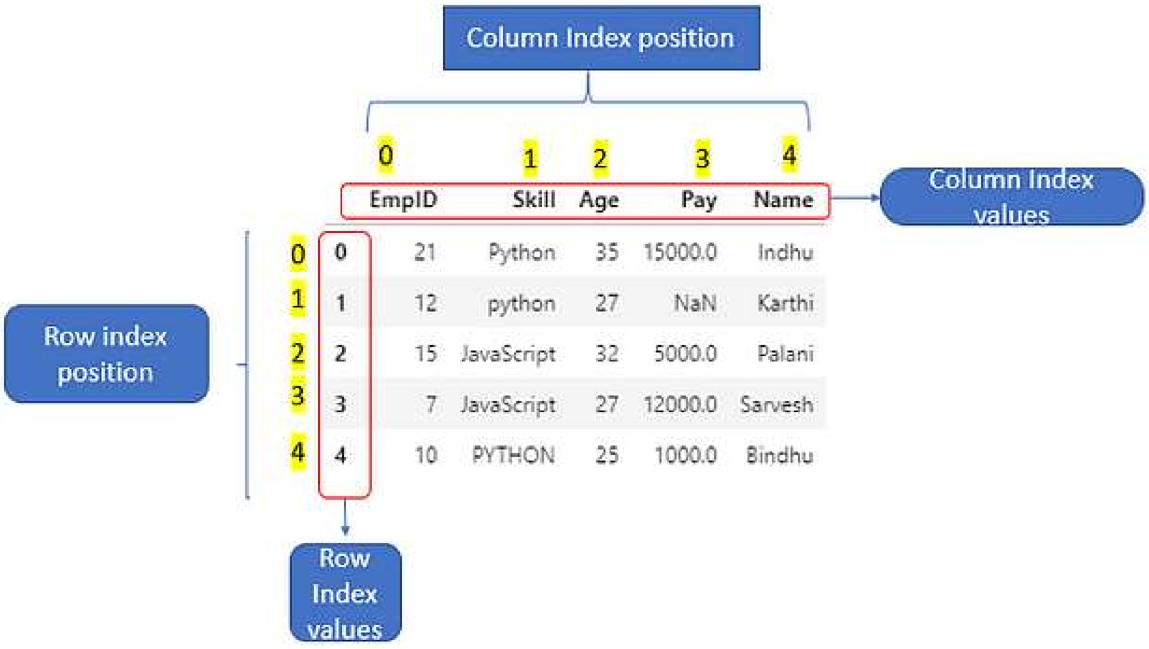
Indexing and Slicing Pandas Dataframe

Indexing and Slicing Pandas DataFrame can be done by their index position/index values.



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

Index position/Index Values -[Image by Author]

Refer to my story of Indexing vs Slicing in Python

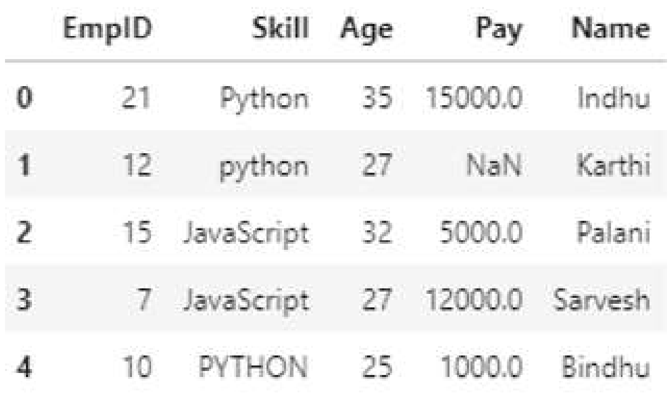
Different ways of Indexing

1. Standard Indexing
2. loc
3. iloc

How to create DataFrame from csv_file.

Let's see how to select rows and columns from the below-mentioned dataframe.

```
pandas pddf=pd.read_csv()df
```



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

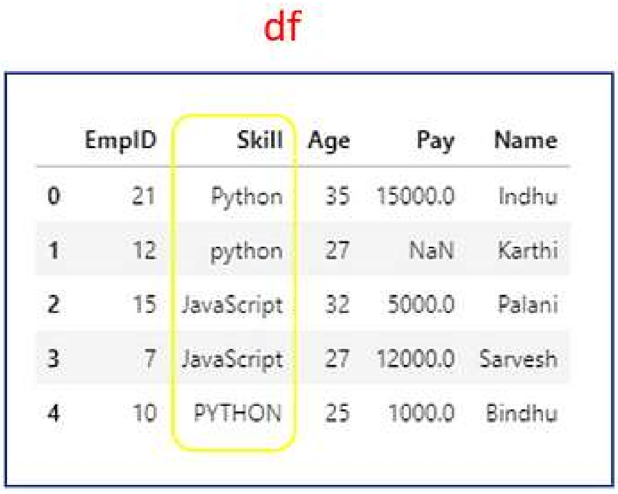
DataFrame (df)

Standard Indexing

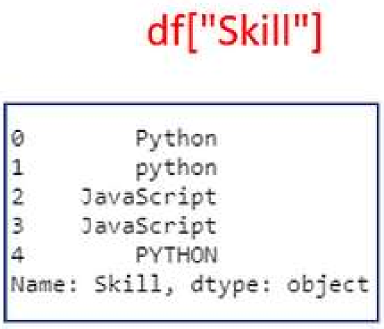
Standard indexing can be done by`[]` notation.

1.

```
df["Skill"]
```



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu



```
0         Python
1         python
2     JavaScript
3     JavaScript
4         PYTHON
Name: Skill, dtype: object
```

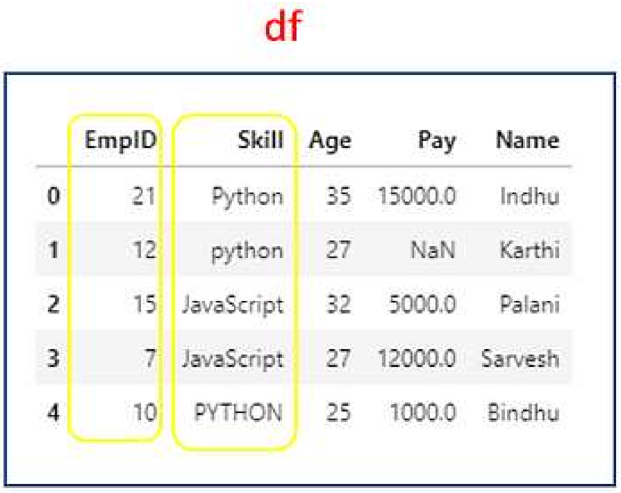
If we select one column, it will return a series.

```
type(df[])
```

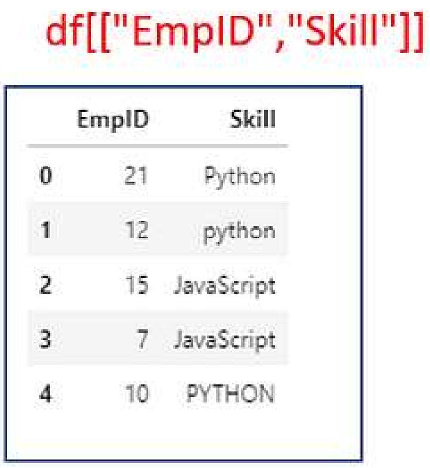
2.Selecting multiple columns

To select multiple columns, we have to give a list of column names.

```
df[["EmpID","Skill"]]
```



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu


df[["EmpID","Skill"]]

	EmpID	Skill
0	21	Python
1	12	python
2	15	JavaScript
3	7	JavaScript
4	10	PYTHON

If we select multiple columns, it will return a dataframe.

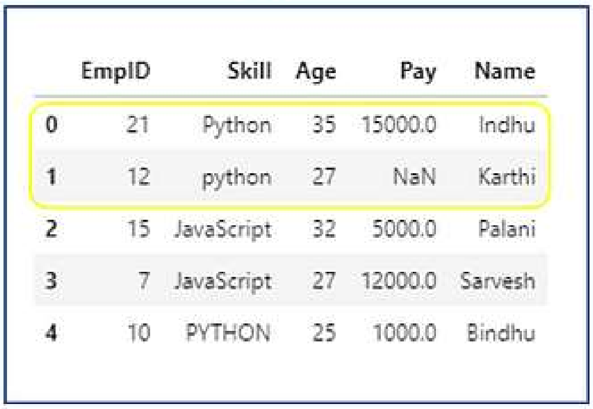
```
type(df[[,]])
```

3.Selecting rows using a slice object

```
df[0:2]
```

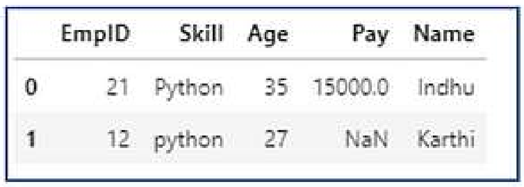
It will select row 0 and row 1. The end index is **exclusive**, the same as python slice.

df



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

df[0:2]



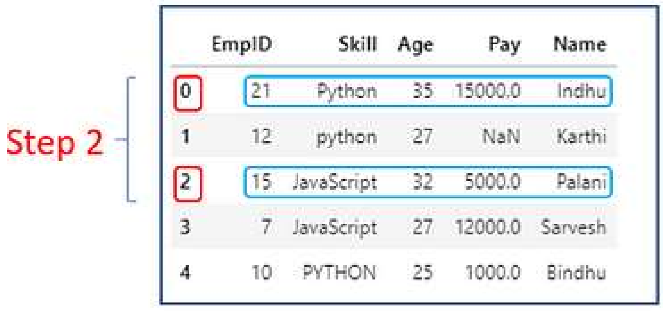
	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi

4. Step is also mentioned in slice object

```
df[0:4:2]
```

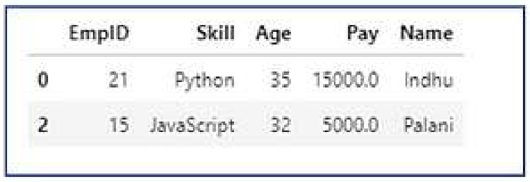
It will start at row 0 and increment by step 2 and end at row4(exclusive).Same as python slice.

df



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

df[0:4:2]

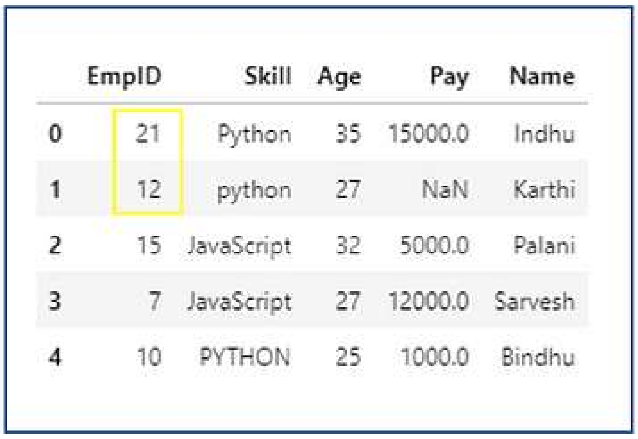


	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
2	15	JavaScript	32	5000.0	Palani

4. Selecting multiple rows and a single column

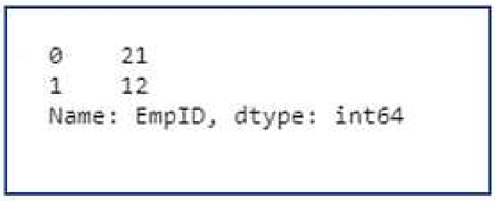
```
df[0:2][“EmpID”]
```

df



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

df[0:2]["EmpID"]

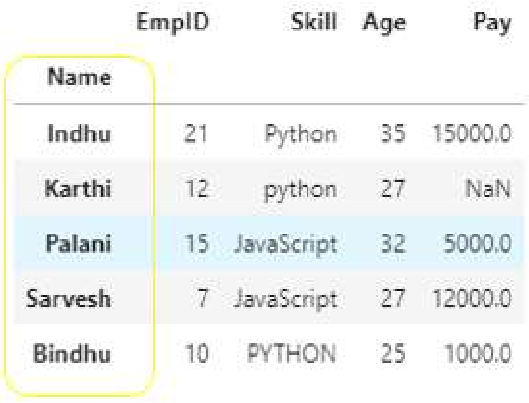


```
0    21
1    12
Name: EmpID, dtype: int64
```

5. Selecting rows using a slice of row_index values

First, we will set the column “`Name`” as row_index

`df1=df.set_index()df1`



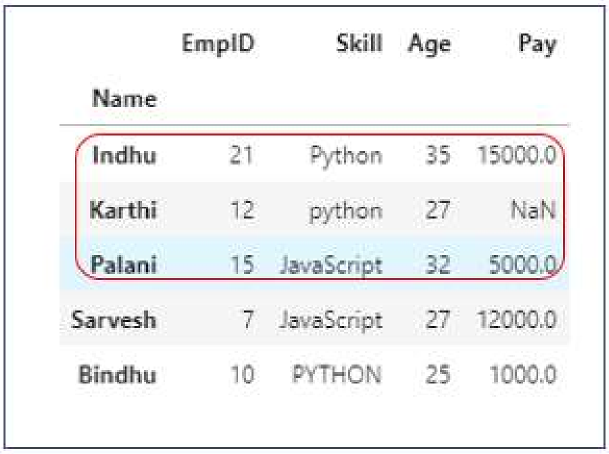
	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Karthi	12	python	27	NaN
Palani	15	JavaScript	32	5000.0
Sarvesh	7	JavaScript	27	12000.0
Bindhu	10	PYTHON	25	1000.0

df1

`df1[“Indhu”:”Palani”]`

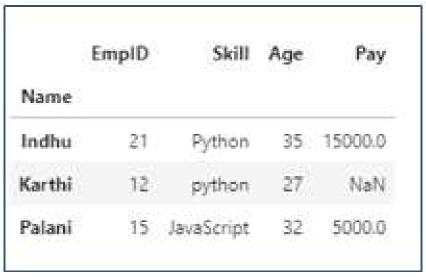
If we mention a slice of row_index values, the end index is **inclusive**.

df1



	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Karthi	12	python	27	NaN
Palani	15	JavaScript	32	5000.0
Sarvesh	7	JavaScript	27	12000.0
Bindhu	10	PYTHON	25	1000.0

df1["Indhu":"Palani"]

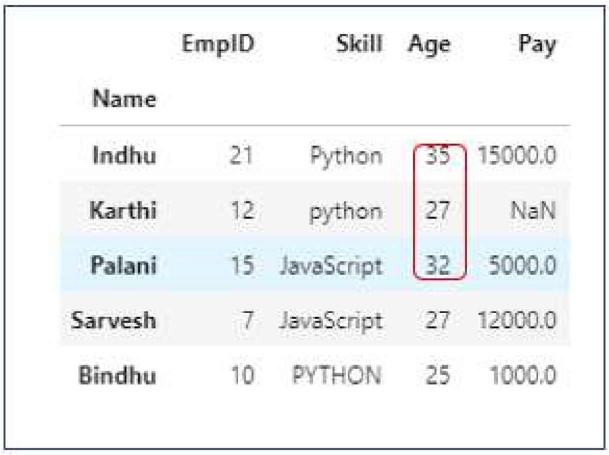


	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Karthi	12	python	27	NaN
Palani	15	JavaScript	32	5000.0

6.Selecting multiple rows and single column using row_index values

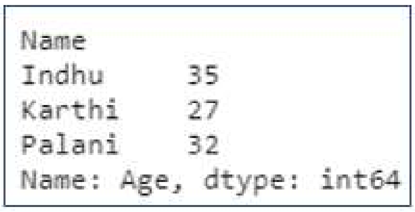
```
df1[“Indhu”:”Palani”][“Age”]
```

df1



	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Karthi	12	python	27	NaN
Palani	15	JavaScript	32	5000.0
Sarvesh	7	JavaScript	27	12000.0
Bindhu	10	PYTHON	25	1000.0

df1["Indhu":"Palani"]["Age"]



```
Name
Indhu     35
Karthi    27
Palani    32
Name: Age, dtype: int64
```

Note:

- We can select columns by specifying column_names only.
- We can select rows by mentioning the slice of row_index values /row_index position.
- While selecting rows, if we use a slice of row_index position, the end index is . But if we use a slice of row_index values/label, the end index is .
- If we select a single column or multiple rows with a single column, it will return a series.
- We have to select rows by mentioning slice only. If we mention row_index or list of row_index, it will raise.

iloc

`.iloc` is primarily integer position based (from `0` to `length-1` of the axis), but may also be used with a boolean array.
`.iloc` will raise `IndexError` if a requested indexer is out-of-bounds, except *slice* indexers which allow out-of-bounds indexing. -Python docs
Allowed inputs are:

- An integer e.g. `5`.
- A list or array of integers `[4, 3, 0]`
- A slice object with ints `1:7`.
- A boolean array (any `NA` values will be treated as `False`).
- A `callable` function with one argument (the calling Series or DataFrame) and that returns valid output for indexing (one of the above).

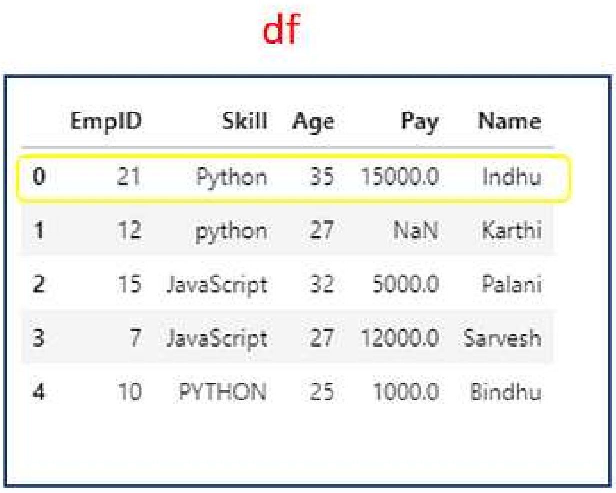
boolean array and callable function → will save this for future post.

Syntax

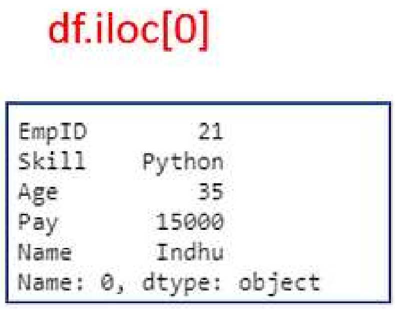
```
df.iloc[row_index_position,col_index_position]
```

1.Selecting a single row using iloc.

```
df.iloc[0]
```



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu



```
EmpID         21
Skill     Python
Age           35
Pay        15000
Name       Indhu
Name: 0, dtype: object
```

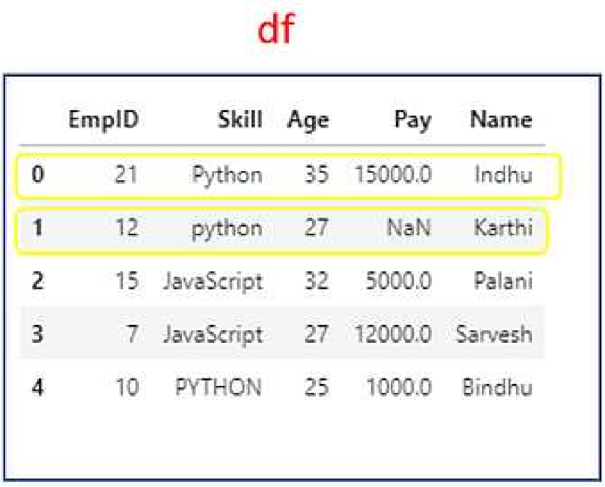
If we select a single row alone, it will return a series.

```
type(df.iloc[0])
```

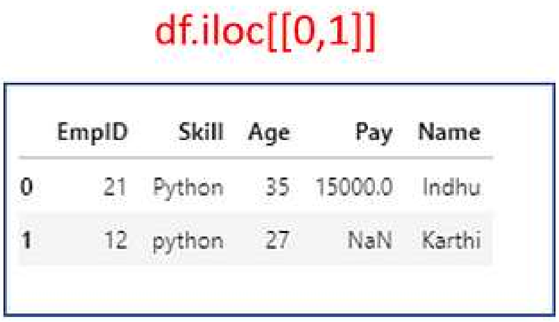
2. Selecting multiple rows using iloc

If we have to select multiple rows, have to specify a list of row_index.

```
df.iloc[[0,1]]
```



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

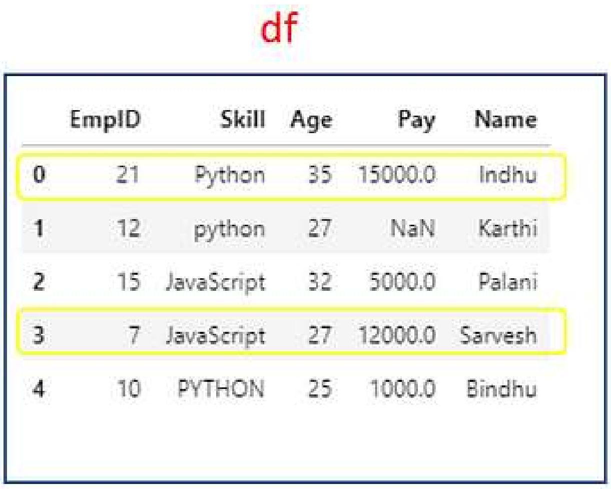


	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi

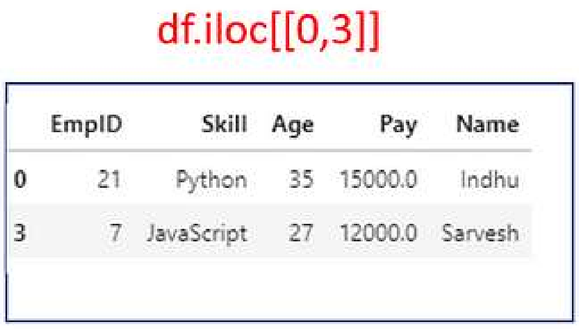
If we select multiple rows, it will return a dataframe.

```
type(df.iloc[[0,1]])
```

```
df.iloc[[0,3]]
```



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

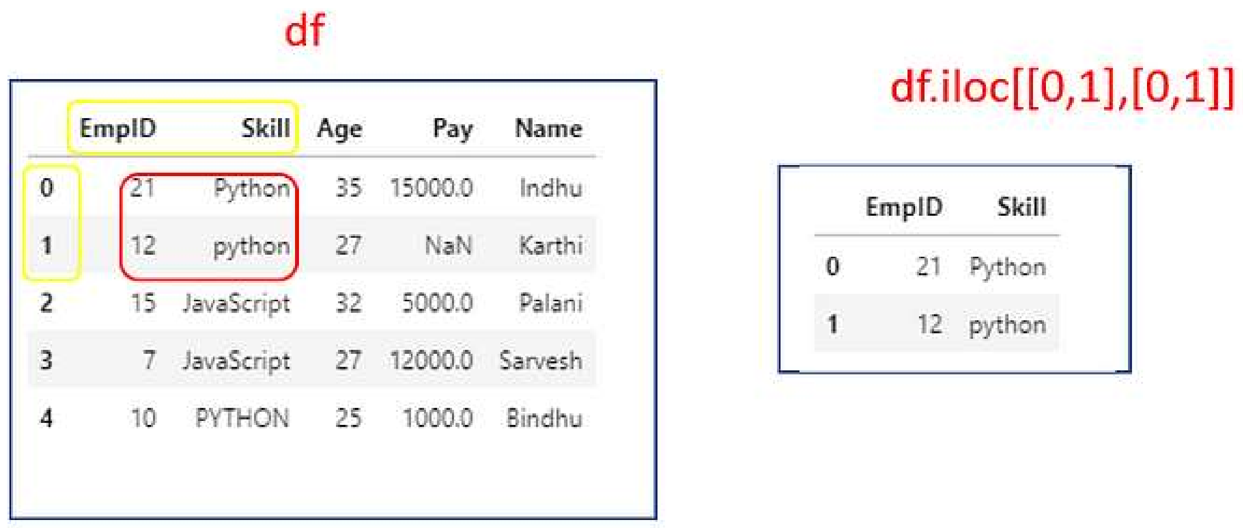


	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
3	7	JavaScript	27	12000.0	Sarvesh

3. Selecting multiple rows and multiple columns using iloc

```
df.iloc[[0,1],[0,1]]
```

Both row and column are mentioned as index positions only.



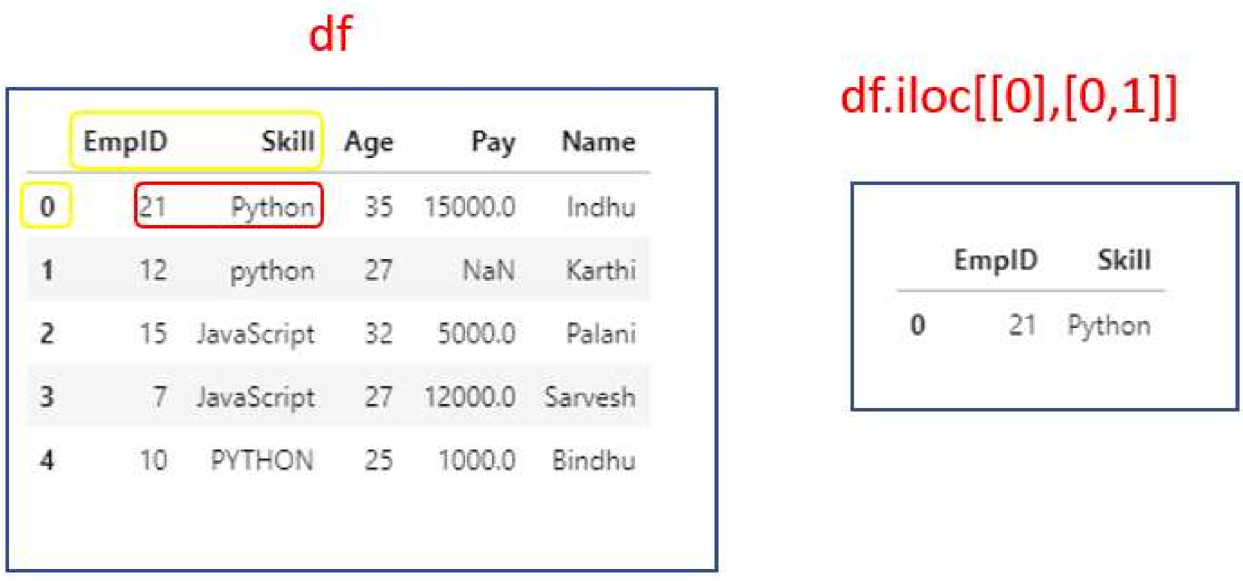
	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

	EmpID	Skill
0	21	Python
1	12	python

Image by Author

4. Selecting a single row and multiple columns using iloc

```
df.iloc[[0],[0,1]]
```

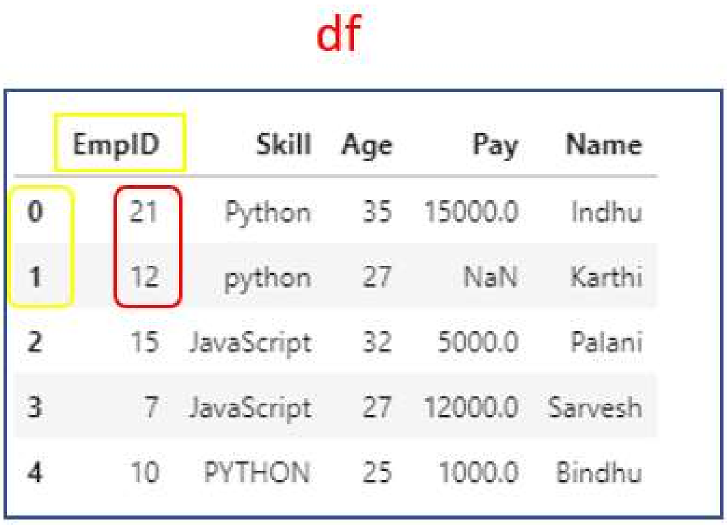


	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

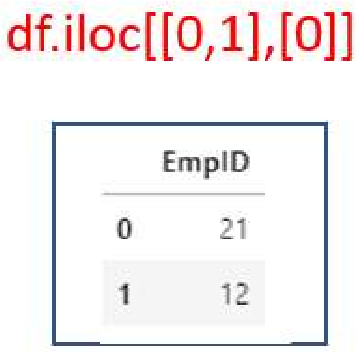
	EmpID	Skill
0	21	Python

5.Selecting multiple rows and single column using iloc

```
df.iloc[[0,1],[0]]
```


df

	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

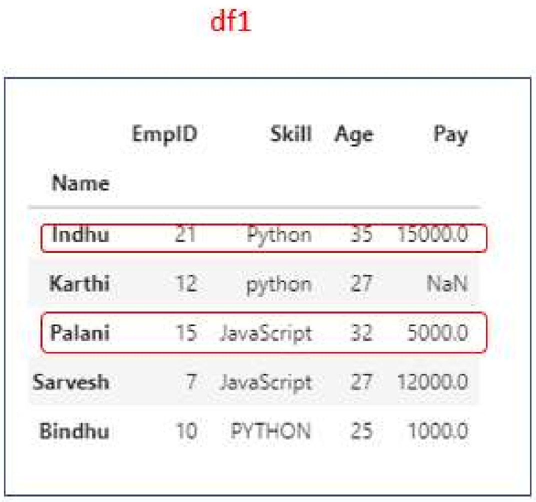

df.iloc[[0,1],[0]]

	EmpID
0	21
1	12

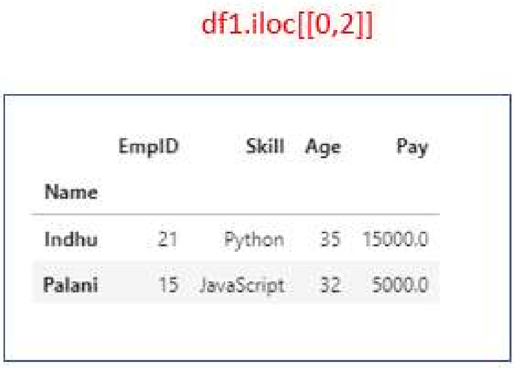
6. Selecting rows by using the row_index position after setting the column as row_index.

```
df1.iloc[[0,2]]
```

The row_index position only mentioned.


df1

	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Karthi	12	python	27	NaN
Palani	15	JavaScript	32	5000.0
Sarvesh	7	JavaScript	27	12000.0
Bindhu	10	PYTHON	25	1000.0


df1.iloc[[0,2]]

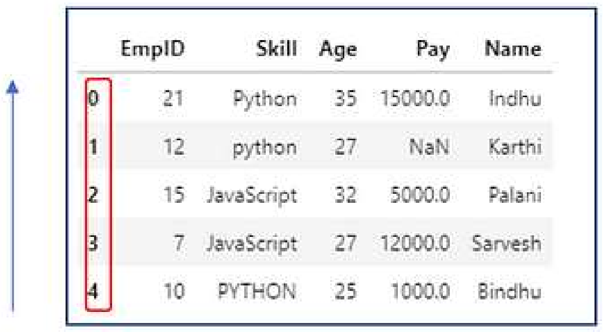
	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Palani	15	JavaScript	32	5000.0

7. Selecting rows by using slice object in iloc

```
df.iloc[::-1]
```

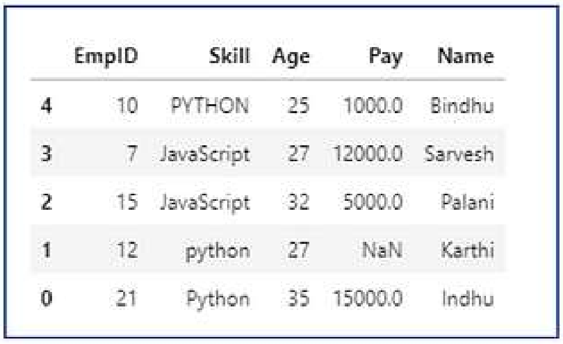
If the start and stop index not mentioned, by default it will start from row 0 and end at the last row.step -1 means in the reverse direction

df



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

df[0:4:2]



	EmpID	Skill	Age	Pay	Name
4	10	PYTHON	25	1000.0	Bindhu
3	7	JavaScript	27	12000.0	Sarvesh
2	15	JavaScript	32	5000.0	Palani
1	12	python	27	NaN	Karthi
0	21	Python	35	15000.0	Indhu

8. Selecting row and columns using slice object in iloc

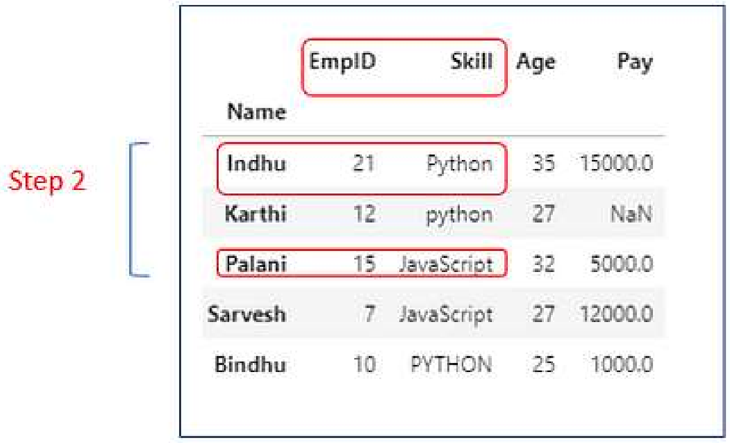
`df1.iloc[0:4:2,0:2]`

`0:4:2` Row_index position. start at row 0, stop at row 4 and increment by 2 (step=2)

`0:2` Column_index position . start at column 0, end at column 2.

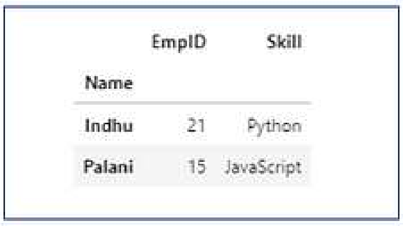
If we use the **index position** in the slice object, the stop index is `exclusive`

df1



	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Karthi	12	python	27	NaN
Palani	15	JavaScript	32	5000.0
Sarvesh	7	JavaScript	27	12000.0
Bindhu	10	PYTHON	25	1000.0

df1.iloc[0:4:2,0:2]



	EmpID	Skill
Name		
Indhu	21	Python
Palani	15	JavaScript

9.IndexError

If we specify column index labels in iloc, it will raise **IndexError**

`df1.iloc[[0,2],[]]`

If we mention row_index values and column_index values,it will raise **IndexError**

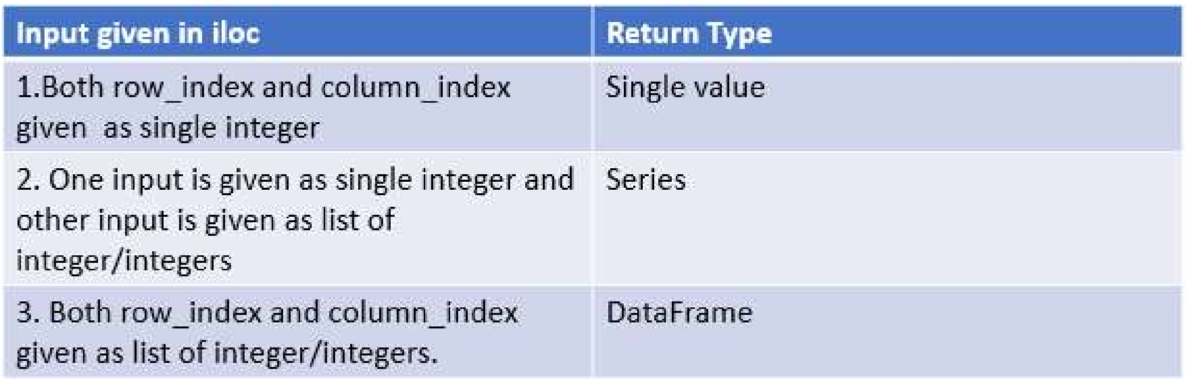
`df1.iloc[[,],[]]`

Note

- By using iloc, we can’t select a single column alone or multiple columns alone.
- We have to mention the row_index position and column_index position only.
- If we mention row_index values or column_index values,it will raise .

- When we use slice object in iloc, the stop index is `exclusive`
- If we select a single row, it will return a series.

Return Type



Input given in iloc	Return Type
1.Both row_index and column_index given as single integer	Single value
2. One input is given as single integer and other input is given as list of integer/integers	Series
3. Both row_index and column_index given as list of integer/integers.	DataFrame

Image by Author

loc

`loc` is primarily label based, but may also be used with a boolean array.
`loc` will raise `KeyError` when the items are not found. -Python docs

Allowed inputs are:

1. Single label `‘a’`
2. List of labels `[‘a’,’b’,’c’]`
3. A slice object with labels `[‘a’:’c’]` . Both start and stop index are
4. A boolean array (any `NA` values will be treated as `False`).
5. A `callable` function with one argument (the calling Series or DataFrame) and that returns valid output for indexing (one of the above).

Boolean array and callable function → will save this for future post.

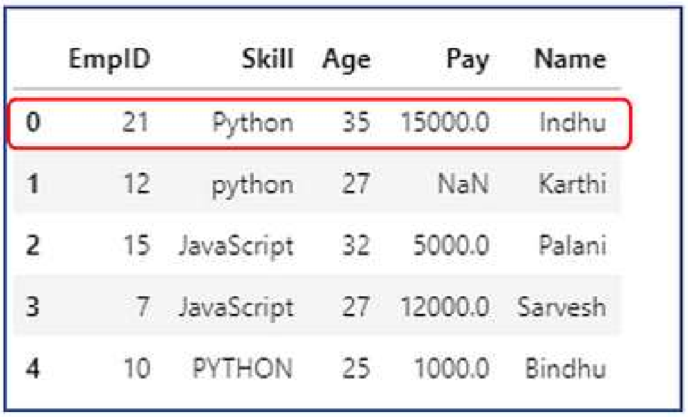
Syntax:

`df.loc[row_index_label,col_index_label]`

1.Selecting single row using loc

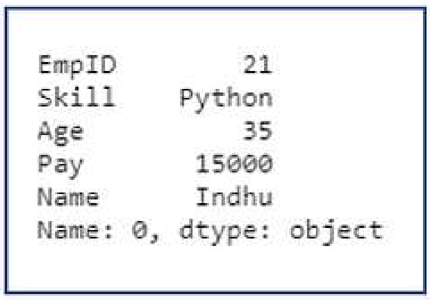
`df.loc[0]`

df



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

df.loc[0]

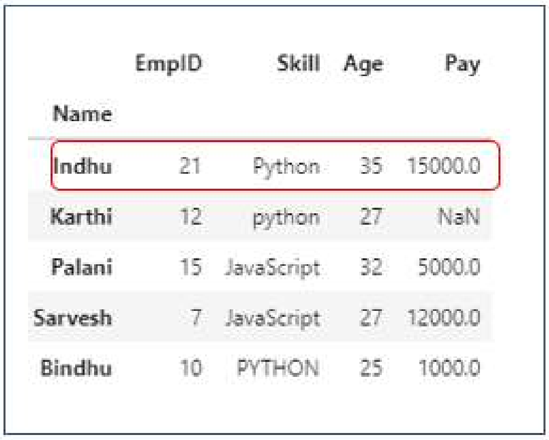


```
EmpID         21
Skill     Python
Age           35
Pay        15000
Name       Indhu
Name: 0, dtype: object
```

If we select a single row, it will return a series.

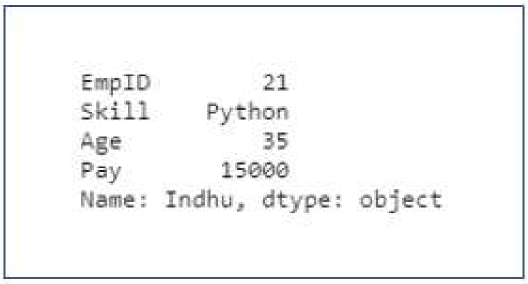
`df1.loc[“Indhu”]`

df1



	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Karthi	12	python	27	NaN
Palani	15	JavaScript	32	5000.0
Sarvesh	7	JavaScript	27	12000.0
Bindhu	10	PYTHON	25	1000.0

df1.loc["Indhu"]



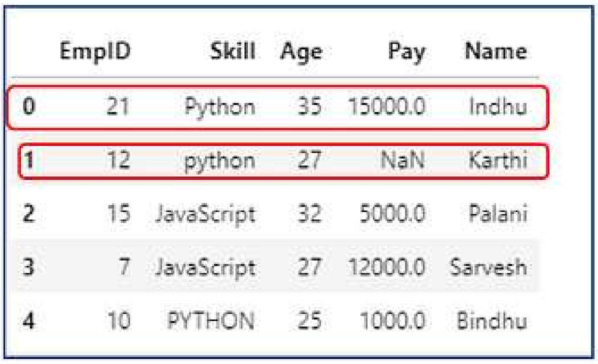
```
EmpID         21
Skill     Python
Age           35
Pay        15000
Name: Indhu, dtype: object
```

2.Selecting multiple rows using loc

To select multiple rows, we have to mention a list of labels.

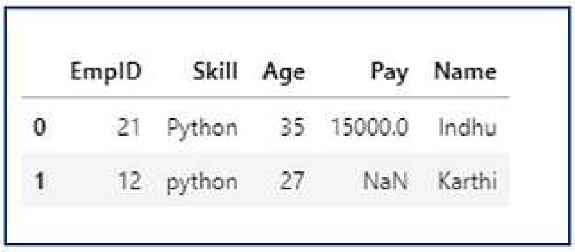
`df.loc[[0,1]]`

df



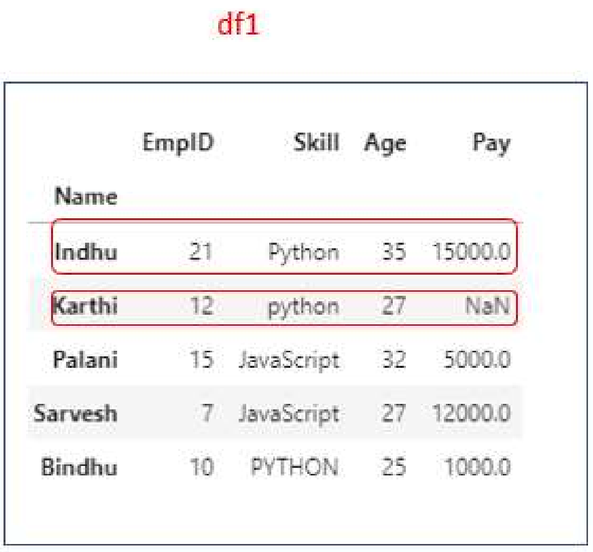
	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

df.loc[[0,1]]

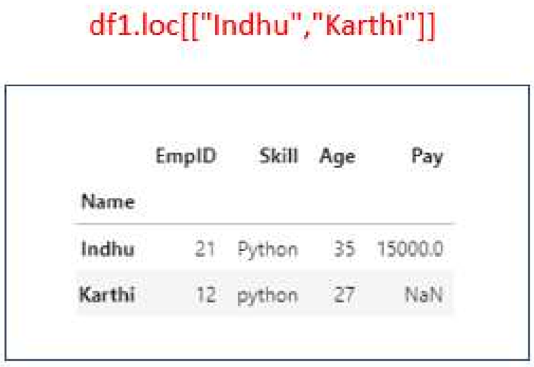


	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi

```
df1.loc[["Indhu","Karthi"]]
```



	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Karthi	12	python	27	NaN
Palani	15	JavaScript	32	5000.0
Sarvesh	7	JavaScript	27	12000.0
Bindhu	10	PYTHON	25	1000.0


df1.loc[["Indhu","Karthi"]]

	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Karthi	12	python	27	NaN

If we select multiple rows, it will return a dataframe.

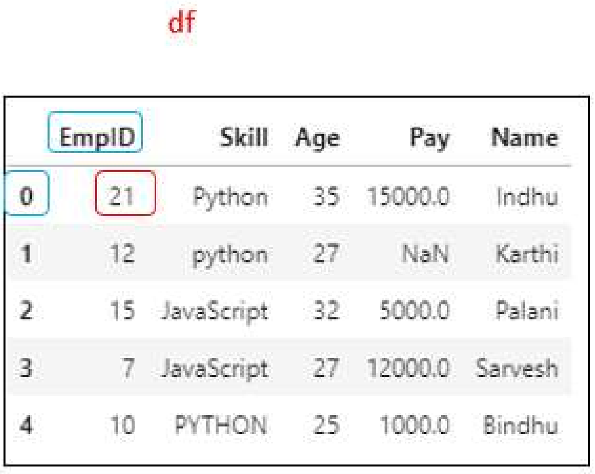
3. Selecting single row and single column using loc

```
df.loc[[0],'EmpID']
```

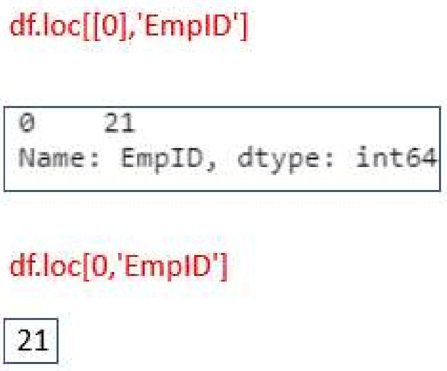
or

```
df.loc[0,'EmpID']
```

Integers are valid labels, but they refer to the label, not the position. Here `0` refers to the label.



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu



```
0     21
Name: EmpID, dtype: int64
```

```
21
```

4.Selecting single row and multiple columns using loc

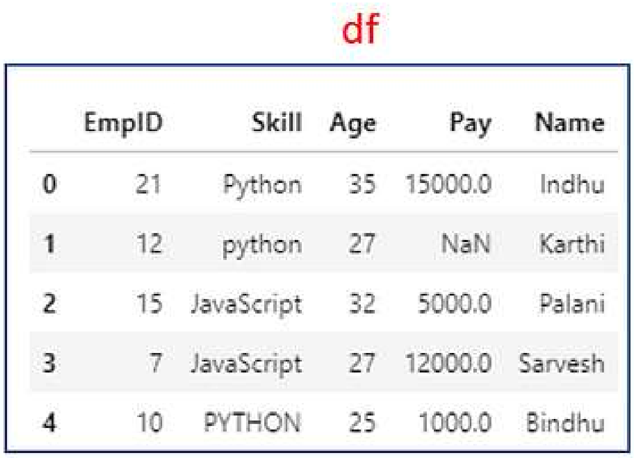
```
df.loc[[0],['EmpID','Skill']]
```

or

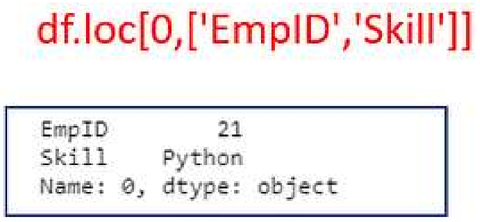
```
df.loc[0,['EmpID','Skill']]
```

If we mention row and column label as list means, it will return a dataframe

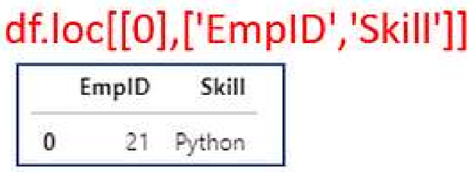
If we mention the row label as a single label and column label as list means, it will return a series.



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu

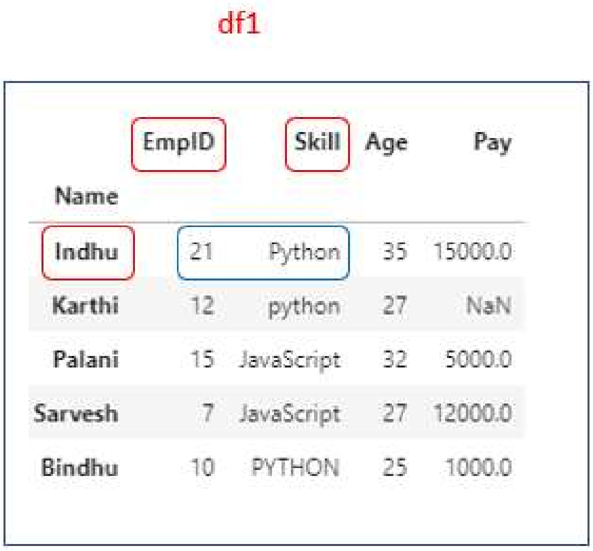


```
EmpID        21
Skill    Python
Name: 0, dtype: object
```

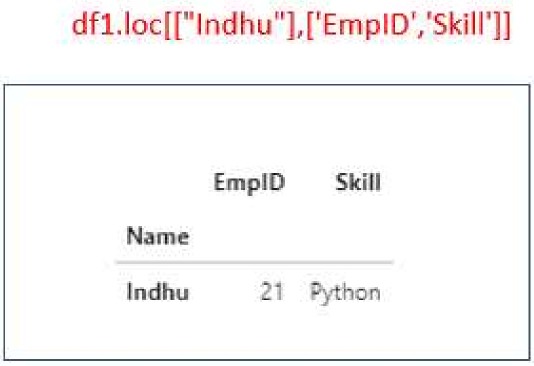

df.loc[[0],['EmpID','Skill']]

	EmpID	Skill
0	21	Python

```
df1.loc[["Indhu"],['EmpID','Skill']]
```



	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Karthi	12	python	27	NaN
Palani	15	JavaScript	32	5000.0
Sarvesh	7	JavaScript	27	12000.0
Bindhu	10	PYTHON	25	1000.0


df1.loc[["Indhu"],['EmpID','Skill']]

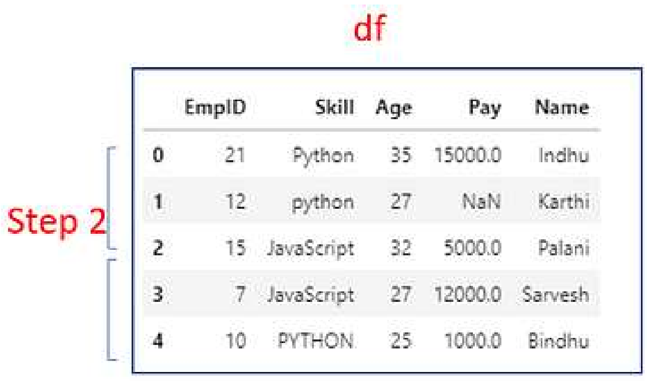
	EmpID	Skill
Name		
Indhu	21	Python

6. Selecting rows using a slice object in loc

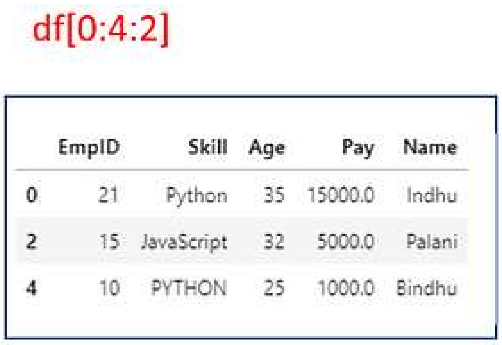
```
df.loc[0:4:2]
```

start at row 0 and stop at row 4, increment by 2. If we use the **index label** in the slice object, the end index is `inclusive`

If we use `loc` ,it is **purely label based indexing.** Integers are valid labels, but they refer to the label, not the position. Here `0` refers to the label.



	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
1	12	python	27	NaN	Karthi
2	15	JavaScript	32	5000.0	Palani
3	7	JavaScript	27	12000.0	Sarvesh
4	10	PYTHON	25	1000.0	Bindhu



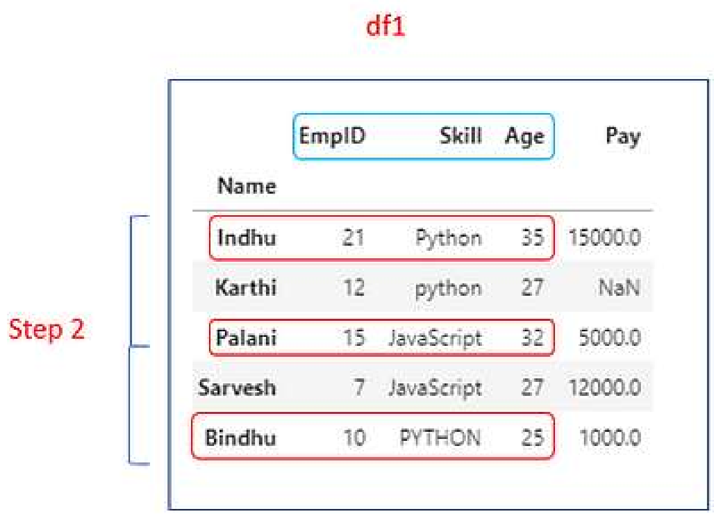
	EmpID	Skill	Age	Pay	Name
0	21	Python	35	15000.0	Indhu
2	15	JavaScript	32	5000.0	Palani
4	10	PYTHON	25	1000.0	Bindhu

7. Selecting rows and columns using slice object in loc

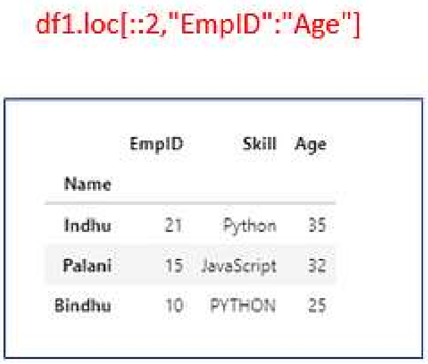
`df1.loc[::2,”EmpID”:”Age”]`

`::2` → Increment by step 2 from the first row to last row.

`”EmpID”:”Age”`-> It includes columns from `“EmpID”` to `“Age”`



	EmpID	Skill	Age	Pay
Name				
Indhu	21	Python	35	15000.0
Karthi	12	python	27	NaN
Palani	15	JavaScript	32	5000.0
Sarvesh	7	JavaScript	27	12000.0
Bindhu	10	PYTHON	25	1000.0

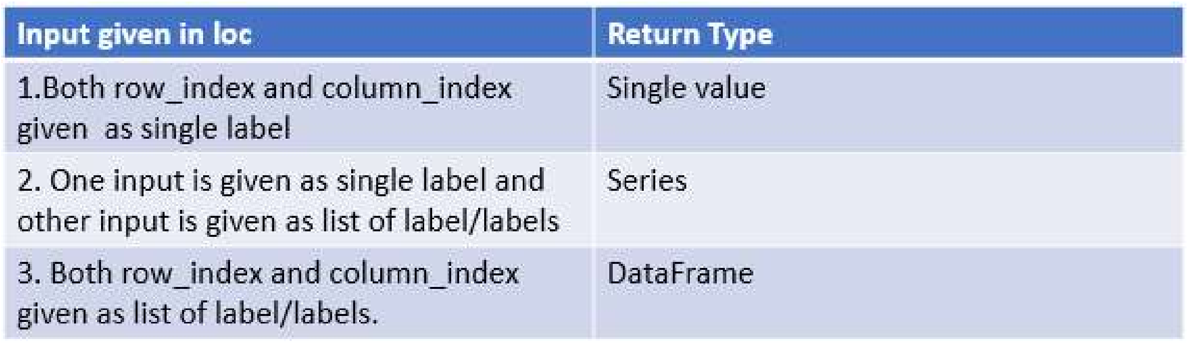

df1.loc[::2,"EmpID":"Age"]

	EmpID	Skill	Age
Name			
Indhu	21	Python	35
Palani	15	JavaScript	32
Bindhu	10	PYTHON	25

Note

- By using loc, we can’t select a single column alone or multiple columns alone.
- We have to mention the row_index label and column_index label only.
- If we mention row_index position or column_index position,it will raise .
- If we select a single row, it will return a series.
- If we give a slice object as row_index /column_index, it should not be written within list[].

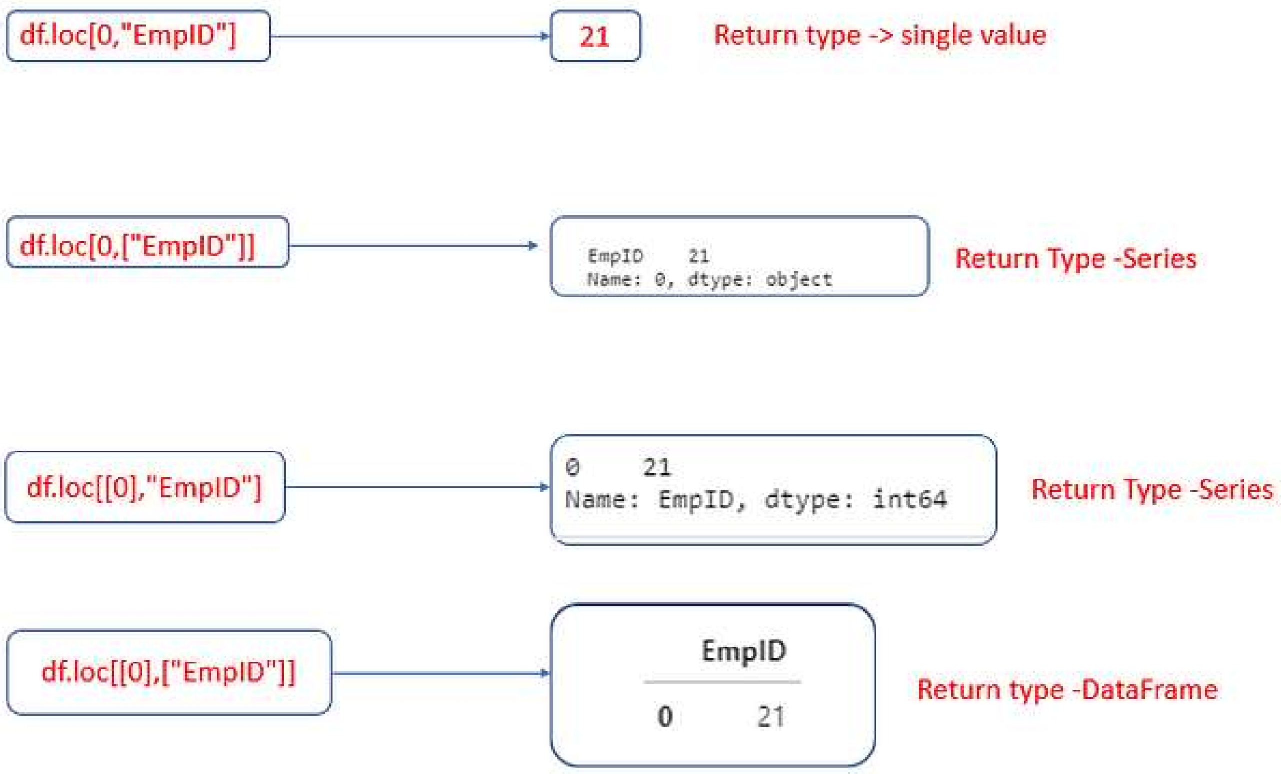
Return Type



Input given in loc	Return Type
1.Both row_index and column_index given as single label	Single value
2. One input is given as single label and other input is given as list of label/labels	Series
3. Both row_index and column_index given as list of label/labels.	DataFrame

Image by Author

Example:



```
df.loc[0,"EmpID"]
```

```
21
```

```
df.loc[0,["EmpID"]]
```

```
EmpID    21
Name: 0, dtype: object
```

```
df.loc[[0],"EmpID"]
```

```
0    21
Name: EmpID, dtype: int64
```

```
df.loc[[0],["EmpID"]]
```

```
   EmpID
0     21
```

Image by Author

Conclusion:

- Using `Standard indexing[]`, we can select a single column or multiple columns. But by using loc and iloc, we can't select a single column alone or multiple columns alone.
- Using `standard indexing[]`, we can select rows by using a slice object only. We can mention row_index values/positions in slice objects.If we use row_index values,end_index is inclusive.If we use the row_index position, the end index is exclusive
- Using loc, it's When slicing is used in loc, both start and stop index is.
- Using iloc, it's . These are `0-based` indexing. When slicing is used in iloc, the start bound is , while the upper bound is .

.