```
-1    7
 0    8
 1    9
dtype: int64
```

```
A    1
B    2
C    3
dtype: int64
```

```
nums = Series([7,8,9], index=[-1,0,1])
x = Series({"A":1, "B":2, "C":3})
y = Series({"A":2, "C":12, "D":4})
```

```
A     2
C    12
D     4
dtype: int64
```

①

Expression	Result(s)
`nums[0]`	8
`nums.loc[0], nums.iloc[0]`	(8, 7)
`nums.loc[-1], nums.iloc[-1]`	(1, 7)
`x / y`	A 0.50 B NaN C 0.25 D NaN dtype: float64

```
0    A
1    B
2    C
3    D
dtype: object
```

```
s = Series(["A", "B", "C", "D"])
letters = Series(["x", "y", "z"], index=[1, 0, 3])
```

```
1    x
0    y
3    z
dtype: object
```

②

Expression	Result(s)
`s[-1]`	KeyError: -1
`s[-2:]`	2 C 3 D dtype: object
`s + s`	0 AA 1 BB 2 CC 3 DD dtype: object
`letters[0]`	'y'
`s + letters`	0 Ay 1 Bx 2 NaN 3 Dz dtype: object'
`s[1:] + s[:-1]`	0 NaN 1 BB 2 CC 3 NaN dtype: object

```
0     -1
1      1
2    200
3    191
4      4
dtype: int64
```

```
v = Series([-1, 1, 200, 191, 4])
```

③

Expression	Result(s)
`v < 0`	0 True 1 False 2 False 3 False 4 False dtype: bool
`v * v == 1`	0 True 1 True 2 False 3 False 4 False dtype: bool
`v[v > 100]`	2 200 3 191 dtype: int64
`v[v % 2 == 0]`	2 200 4 4 dtype: int64
`v[(v>0) & (v<100)]`	1 1 4 4 dtype: int64

note: `Series.loc[X]` looks for label X in the **index**. `Series.iloc[X]` looks for the **int position** X. These names are confusing. `iloc` supports negative indexing.

Code:	**storms.csv:**

`path = "storms.csv"`
`tab = pd.read_csv(path)`
 	
`map = DataFrame({`
` "code": ["o","p","a"],`
` "where": ["other","Pacific","Atlantic"]`
`})`
	name,year,type,speed,place
	alice,2016,tornado,100,o
	bob,2016,hurricane,200,p
	cindy,2017,tornado,150,o
	dan,2018,tornado,300,o
	eve,2018,hurricane,250,a

4

Expression	**Result(s)**
`map["code"]`	
`map.code`	
`type(map.code), type(map.where)`	
`tab.year.mean()`	
`tab.year == 2018`	
`tab.name[tab.year == 2018]`	
`map["where"] == "Atlantic"`	

`b = map["where"] == "other"`
`code = map.code[b].item()`
`nms = tab.name[tab.place==code]`
	# what are b, code, nms?

5

Expression	**Result(s)**
`tab.loc[0]`	
`tab.loc[4, "type"]`	

`map.loc[0,"where"] = "mainland"`
`place = map["where"][0]`
	# what is place?

`tab.loc[:, "speed"] += 1`
`col = tab.speed`
	# what is col?

note: s.COL is a shortcut for s["COL"], unless COL collides with a method name
also: when a Series s contains exactly one one item, s.item() extracts it