

```

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})

```

①

Expression	Result(s)
nums[0]	
nums.loc[0], nums.iloc[0]	
nums.loc[-1], nums.iloc[-1]	
x / y	

```

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

```

②

Expression	Result(s)
s[-1]	
s[-2:]	
s + s	
letters[0]	
s + letters	
s[1:] + s[:-1]	

```

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

```

③

Expression	Result(s)
v < 0	
v * v == 1	
v[v > 100]	
v[v % 2 == 0]	
v[(v>0) & (v<100)]	

**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:
<pre>path = "storms.csv" tab = pd.read_csv(path)  map = DataFrame({   "code": ["o", "p", "a"],   "where": ["other", "Pacific", "Atlantic"] })</pre>	<pre>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</pre>

④

Expression	Result(s)
<code>map["code"]</code>	
<code>map.code</code>	
<code>type(map.code), type(map.where)</code>	
<code>tab.year.mean()</code>	
<code>tab.year == 2018</code>	
<code>tab.name[tab.year == 2018]</code>	
<code>map["where"] == "Atlantic"</code>	
<pre>b = map["where"] == "other" code = map.code[b].item() nms = tab.name[tab.place==code]</pre>	# what are b, code, nms?

⑤

Expression	Result(s)
<code>tab.loc[0]</code>	
<code>tab.loc[4, "type"]</code>	
<pre>map.loc[0, "where"] = "mainland" place = map["where"][0]</pre>	# what is place?
<pre>tab.loc[:, "speed"] += 1 col = tab.speed</pre>	# 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 item, `s.item()` extracts it