Question X: what is printed?

```
x = 0
def reset():
    x = 0
def inc():
    global x
    x += 1
inc()
reset()
inc()
print(x)
```

Question X: what is printed?

```
def fraction(top=1, bottom=1):
    return top/bottom
print(fraction(bottom=2))
```

Question X: what is printed?

```
stats = {}
results = []
for i in range(5):
    stats["score"] = 100+i
    results.append(stats)
print(results[2]["score"])
```

Question X: what is printed?

```
# assume nums.json contains this:
# [200, 300, 100]
r = requests.get("https://example.com/nums.json")
nums = r.text
print(nums[1])
```

Question X: how many columns does this table have?

```
<table>
<tr><td>A1</td><td>A2</td>
<td>B1</td></tr><tr><td>B2</td>
<td>C1</td><td>C2</td></tr>
<table>
```

Question X: what is printed?

```
def mystery(n):
    if n == 0:
        return 1
    return 2 * mystery(n-1)
print(mystery(3))
```

Question X: which expressions would cause a KeyError exception?

```
d = {1:"one", 2:"two", 3:"three"}
```

- `d[1]`
- `d[-1]`
- `d["one"]`

Question X: does it run forever?

```
while True:
    f = open("file.txt")
    print(f.read())
f.close()
```

Question X: does it run forever?

```
while True:
    f = open("file.txt")
    print(f.read())
    f.close()
```

Question X: what is printed? (assume file.txt exists before)

```
f = open("file.txt")
try:
    print("A")
    f.write("hey")
    print("B")
except:
    print("C")
f.close()
```

Question X: what are the query results?

```
SELECT * FROM shirts WHERE price < 15;

SELECT size FROM shirts WHERE color = 'green';

SELECT MAX(price) FROM shirts;

SELECT size AVG(price) FROM shirts
GROUP BY size;

SELECT size, COUNT() as c FROM shirts
GROUP BY size
HAVING c < 2;
```

shirts table

size	color	price
S	red	14
S	blue	18
M	green	12
L	red	15
L	red	25
L	blue	50

Question X: will happen if you manually re-run In[2], given the following notebook state?

```
In [1]: s = Series([1, 2, 3])

In [2]: s = 1 / s

In [3]: s = 1 - s
```

Question X: what does each expression yield, given this setup?

```
s = [5,6,7,8]
```

- `s - 5`
- `s[-3:]`
- `s[:3] + s[-3:]`
- `s == 7`
- `s[s == 7]`
- `s % 2 == 0`
- `s[s % 2 == 0]`
- `s[s < 7].sum()`

Question X: what does each expression yield?

```
pts = DataFrame({
  "x": [10, 20, 30, 40],
  "y": [1, 10, 100, 1000],
})
```

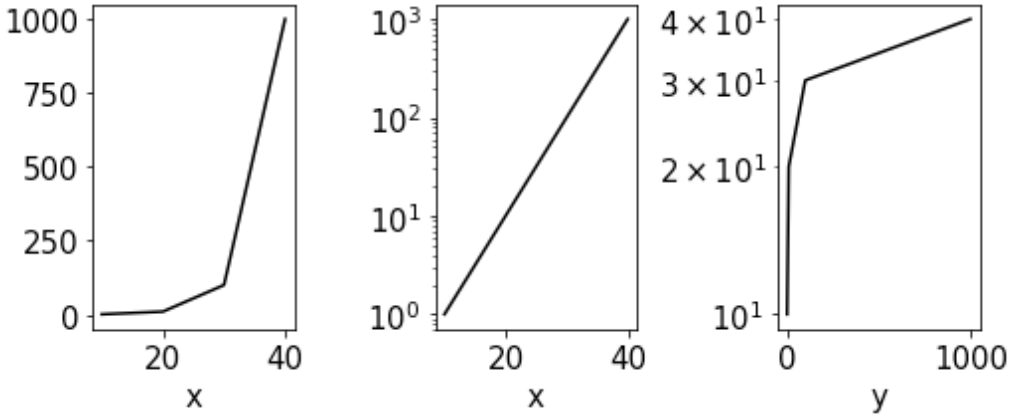
- `pts["x"][2]`
- `pts.loc[3].sum()`
- `pts["x"][2] - pts.loc[2]["x"]`
- `pts["y"].sum()`
- `pts["x"].mean()`
- `pts["x"] - pts["x"]`
- `pts["x"] - pts["x"].mean()`
- `((pts.loc[1]-pts.loc[0])**2).sum()**(1/2)`
  final answer can be mathematical expression

`pts` DataFrame

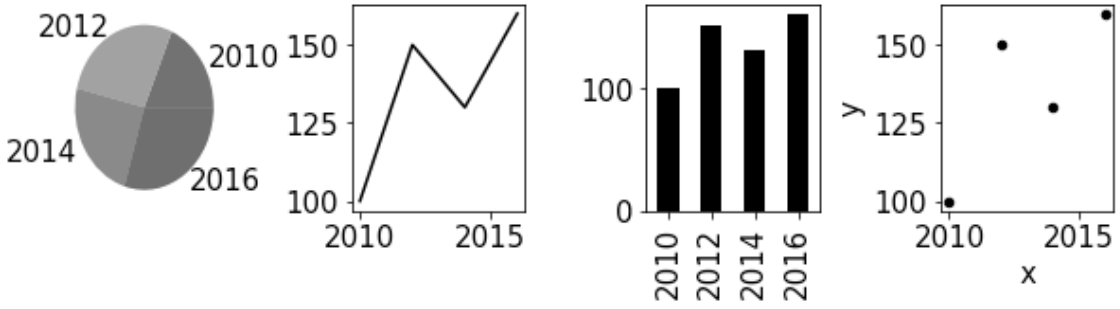
	x	y
0	10	1
1	20	10
2	30	100
3	40	1000

Question X: which plot is generated?

```
pts.plot.line(x="x", y="y", logy=True)
```



Question X: label each as pie, scatter, line, or bar



Question X: which call creates the bar plot? (A, B, or C)

```
s = Series(["red", "red", "red",
            "green", "blue", "blue"])
vc = s.value_counts()

vc.sort_values().plot.bar() # A

vc.sort_index().plot.bar() # B

s.set_index("color").plot.bar() # C
```