

Question 1: what is printed?

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

Question 2: what is printed?

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

Question 3: what is printed?

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

Question 4: what is printed?

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

Question 5: 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 6: what is printed?

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

Question 7: which expressions would cause a KeyError exception?

```
d = {1:"one", 2:"two", 3:"three"}
• d[1]
• d[-1]
• d["one"]
```

Question 8: what is printed?

file.txt:

A	B	C	
D	E	F	G
H	I		

```
f = open("file.txt")
txt = f.read()
parts = txt.split("\n")
print(parts[1].split(" ")[1])
f.close()
```

Question 9: what is printed? (assume file.txt exists before)

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

Question 10: 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 11: what is the range of possible values for total?

```
from numpy.random import choice

nums = Series(choice(4, 10))
total = nums.sum()
```

Question 12: what does each expression evaluate to?

```
s = Series([5,6,7,8])
(a) s - 5
(b) s / s
(c) s[-3:]
(d) s[:3] + s[-3:]
(e) s == 7
(f) s[s == 7]
(g) s % 2 == 0
(h) s[s % 2 == 0]
(i) s[s < 7].sum()
(j) s - s.mean()
(k) s - s[s < 8].mean()

s["total"] = s.sum()
(l) s["total"]

letters = Series(["W", "X", "Y", "Z"], index=[3, 2, 1, 0])
(m) letters[1]
(n) letters.loc[1]
(o) letters.iloc[1]
```

Question 13: what does each evaluate to?

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

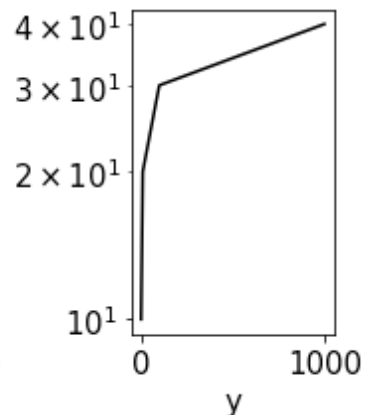
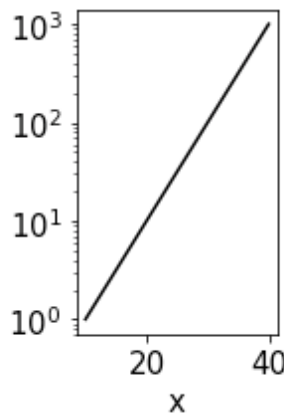
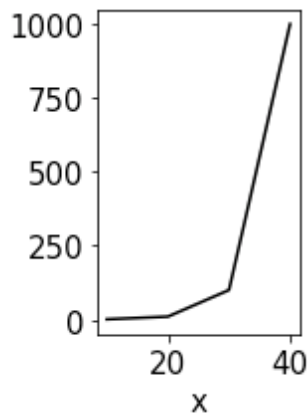
- (a) pts["x"][2]
- (b) pts.loc[3].sum()
- (c) pts["x"][2] - pts.loc[2]["x"]
- (d) pts["y"].sum()
- (e) pts["x"].mean()
- (f) pts["x"] - pts["x"]
- (g) pts["x"] - pts["x"].mean()

pts DataFrame

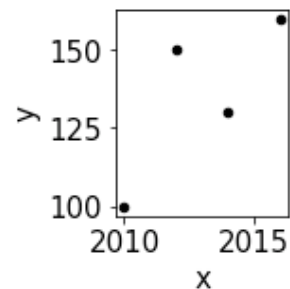
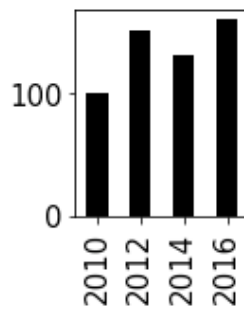
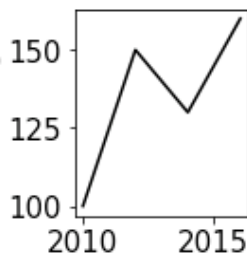
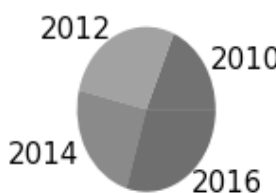
	x	y
0	10	1
1	20	10
2	30	100
3	40	1000

Question 14: which plot is generated?

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

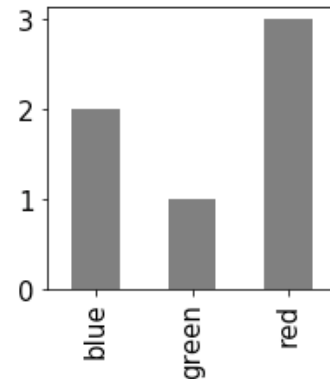


Question 15: label each as pie, scatter, line, or bar



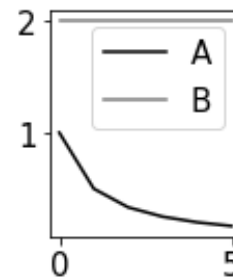
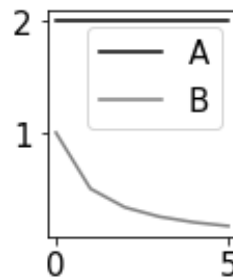
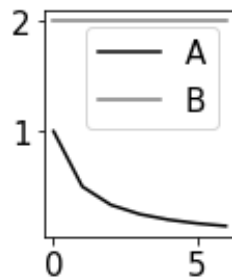
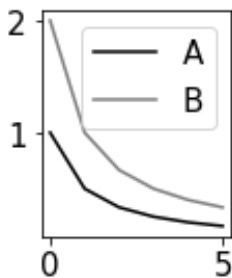
Question 16: which call creates the bar plot? (A, B, or C)

```
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
```



Question 17: which plot is created by the code?

```
data = {"A":[], "B":[]}  
for i in range(6):  
    data["A"].append(1/(i+1))  
    data["B"].append(2)  
DataFrame(data).plot.line()
```



Question 18: what does each expression yield, given this setup?

```
df = DataFrame({  
    "year": [2015, 2016, 2017, 2018],  
    "cats": [20, 15, 10, 12],  
    "dogs": [30, 15, 15, 18],  
})  
df["next"] = df["year"] + 1
```

- (a) `df["cats"][2]`
- (b) `df["dogs"] + df["cats"]`
- (c) `df.set_index("year")["cats"][2015]`
- (d) `df.set_index("next")["dogs"][2016]`
- (e) `df["cats"][1:] - df["cats"]`
- (f) `df.set_index("year")["dogs"] - df.set_index("next")["dogs"]`