28/35

CS 301 - Spring 2019
Instructors: Tyler Caraza-Harter and Caroline Hardin

Exam 2 — 15%

(Last) Surname: Lin (First) Given name: Ya-Yen

NetID (email): lin383 @wisc.edu

Fill in these fields (left to right) on the scantron form (use #2 pencil):

1. LAST NAME (surname) and FIRST NAME (given name), fill in bubbles
2. IDENTIFICATION NUMBER is your Campus ID number, fill in bubbles
3. Under *ABC* of SPECIAL CODES, write your lecture number, fill in bubbles:
   001 - MWF 9:55pm (Caroline)
   002 - MWF 1:20pm (Tyler afternoon)
   003 - MWF 8:50am (Tyler morning)
4. Under *F* of SPECIAL CODES, write **A** and fill in bubble **6**

If you miss step 4 above (or do it wrong), the system may not grade you against the correct answer key, and your grade will be no better than if you were to randomly guess on each question. So don't forget!

Many of the problems in this exam are related to the course projects, but some questions assume the availability of slightly different functions (e.g., for accessing the data). We won't have any trick questions where we call a function that doesn't exist and you need to notice. Thus, if you see a call to a function we haven't explicitly defined in the problem, assume the function was properly implemented (perhaps immediately before the code snippet we DO show) and is available to you.

You may only reference your notesheet. You may not use books, your neighbors, calculators, or other electronic devices on this exam. Please place your student ID face up on your desk. Turn off and put away portable electronics now.

Use a #2 pencil to mark all answers. When you're done, please hand in these sheets in addition to your filled-in scantron.

(Blank Page)

General

C 1. Using `asserts` is most useful for dealing with which category of errors?

A. `syntax` B. `runtime` C. `semantic` D. `exceptional`

D 2. What is true about frames?

A. there is always exactly one frame per function definition

B. every object is associated with exactly one frame

C. all of the above In recursion, there are multiple frames of the same function object in a stack.

D. none of the above

B 3. What will be in the file after this code runs? The file doesn't exist when the code starts.

```
f = open("file.txt", "w")
f.write("Hello")
f.close()

f = open("file.txt", "w")
f.write("World")
f.close()
```

A. `"Hello"`

B. `"World"`

C. `"HelloWorld"`

D. `"Hello World"`

E. `"Hello" and "World" on different lines`

D 4. What will be printed?

```
def f(x, y):
    if len(x) == 0:
        return -1
    elif x[0] == y:
        return x
    else:
        return f(x[1:], y)
print(f("abcd", "c"))
```

abcd, c

bcd, c

cd, c

A. `-1` B. `""` C. `"c"` D. `"cd"` E. `"abcd"`

5. What type of objects will be immutable?

A. `list` B. `dict` C. `namedtuple` D. `recordclass`

6. What is printed by the print call? Be careful!

```
word = "banana"
counts = dict() # line A

for letter in word:
    if not letter in counts:
        counts[letter] = 1
    else:
        counts[letter] += 1 # line B
print(counts["a"]) # what does this print?
```

A. 1

B. 3

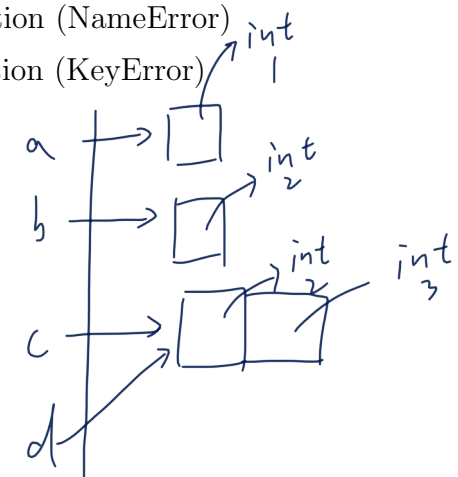
C. 6

D. Nothing: line A will cause an exception (NameError)

E. Nothing: line B will cause an exception (KeyError)

7. What will the following code print?

```
a = [1]
b = [2]
c = [2]
d = c
d.append(3)
print(b, c)
```



A. `[2,3] [2]` B. `[2] [2]` C. `[2,3] [2,3]` D. `[2] [2,3]` E. `[2] [2,[3]]`

Wine

When considering each question, assume the initial code executed is as follows (if a question contains code that modifies the objects, those changes **should not** be considered in other questions):

...continued on next page...

```
header = ["name", "flavor1", "flavor2"]
wines = [
    ["Cabernet Franc", "orange", "vanilla"],
    ["Cabernet Sauvignon", "vanilla", "cherry"],
    ["Malbec", "orange", None],
    ["MALBEC", "cherry", None],
]

def get_flavors(wines):
    for w in wines:
        # this was different on the original exam document,
        # but the proctors announced that this version
        # should be used
        yield w[header.index("flavor1")]
        yield w[header.index("flavor2")]

counts = {}
for wine in wines:
    name = wine[0]
    name = name.split(" ")[0].upper()
    if not name in counts:
        counts[name] = 0
    counts[name] += 1
```

list of lists

["Cabernet", "Franc"]

→ "CABERNET"

A

8. what is the type of `header`?

   A. `list` B. `tuple` C. `namedtuple` D. `dict` E. `csv`

wines[:1] = wines[0]

length = 1

so [1:] will return an empty list.

D

A

9. what does the following evaluate to?

   `wines[:1][1:]` = wines[0][1:]

   A. `[]`
   B. `["Cabernet Franc", "orange", "vanilla"]`
   C. `["Cabernet Sauvignon", "vanilla", "cherry"]`
   D. `["orange", "vanilla"]`
   E. `["cherry"]`

D

10. What does the following evaluate to?

    `wines[2][header.index("flavor1")]`

    A. `"flavor1"` B. `"vanilla"` C. `"cherry"` D. `"orange"` E. `None`

A

11. What does the following evaluate to?

    `header[wines[1].index("vanilla")]`

    A. `"flavor1"` B. `"vanilla"` C. `"cherry"` D. `"orange"` E. `None`

*A "function" with a yield statement will always return a generator object, regardless of what those yields are doing

12. To what type of object does `obj` refer?

```
obj = get_flavors(wines)
```

↳ generator object s = next(obj) ⇒ string

A. str B. regular function C. generator function D. generator object E. None

13. What will the following evaluate to?

```
list(get_flavors(wines))
```

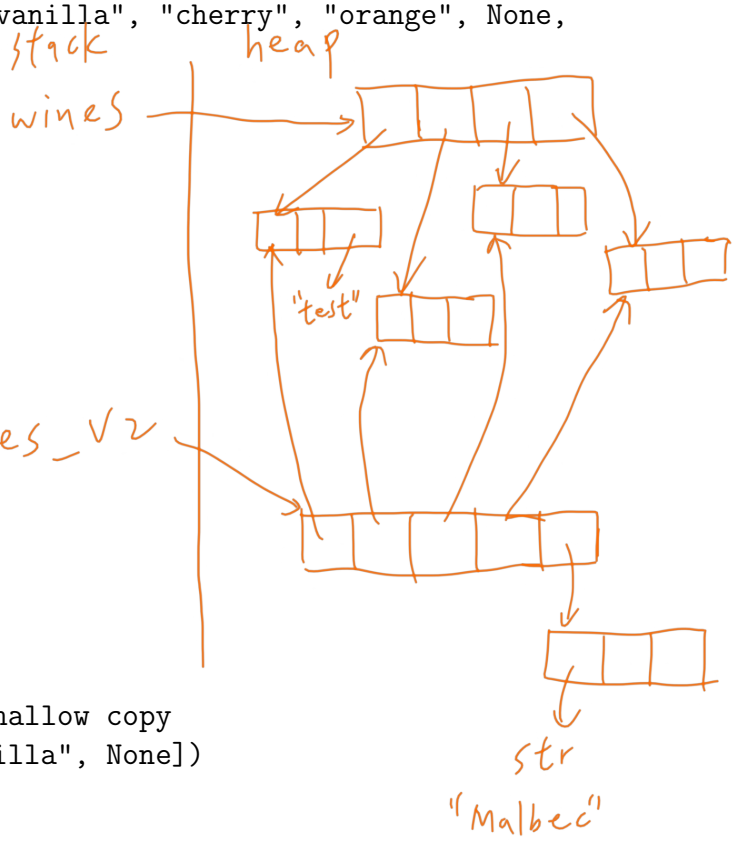
A. `["orange"]`

B. `["orange", "vanilla", "orange", "cherry"]`

C. `["orange", "vanilla", "cherry"]` (or another list containing these three in a different order)

D. `["orange", "vanilla", "vanilla", "cherry", "orange", "cherry"]`

E. `["orange", "vanilla", "vanilla", "cherry", "orange", None, "cherry", None]`



14. What of the following is True?

A. `1 in counts`

B. `2 in counts`

C. `4 in counts`

D. `"CABERNET" in counts`

E. `"malbec" in counts`

15. what is `len(counts)`?

A. 1 B. 2 C. 3 D. 4 E. 5

16. What does the following print?

```
import copy
wines_v2 = copy.copy(wines) # shallow copy
wines_v2.append(["Malbec", "vanilla", None])
wines_v2[0][2] = "test"
print(wines[0][2], len(wines))
```

A. `vanilla 4` B. `vanilla 5` C. `test 4` D. `test 5` E. `None 5`

test 4

Soccer

When considering each question, assume the initial code executed is as follows (if a question contains code that modifies the objects, those changes **should not** be considered in other questions):

```
from collections import namedtuple
Player = namedtuple("Player", ["name", "rating", "age"])

e = Player("Evelyn", 87, 25)
k = Player("Kylo", 82, 24)
p = Player("Pamela", 78, 26)

def player_age(p):
    return p.age

def player_rating(p):
    return p.rating

players = [e, k, p]

try:
    f = open("ratings.txt", "w")
    for player in players:
        f.write(player.name + ":" + str(player.rating))
    f.close()
except Exception as e:
    print("save failed!")
```

B

17. T/F: if there an exception occurs during a call to the `write` method, the call to `close` will still occur.

    A. True B. False

D

18. Instead of using namedtuples, what should we use if we want assignment such as the following to succeed?

```
e.age += 1
```

    A. list B. tuple C. dict D. recordclass

B

19. If there are no exceptions/errors, how many lines will be in ratings.txt?

    A. `0` B. `1` C. `2` D. `3` E. `4`

20. What does the following evaluate to?

```
player_rating(e)
```

A. `None` B. `0` C. `1` D. `78` E. `87`

21. What numbers are printed, and in what order?

```
funcs = [player_rating, player_rating, player_age]
for player in players:
    f = funcs.pop(0)
    print(f(player))
```

A. 0, 0, 0 B. 25, 24, 26 C. 87, 82, 78 D. 87, 82, 26 E. 26, 82, 87

22. What does the following evaluate to (remember that Python sorts numbers from small to large by default)?

```
sorted(players, key=player_rating)[0].name
```

A. `"Evelyn"` B. `"Kylo"` C. `"Pamela"` D. `78` E. `87`

23. Which of the following is valid JSON? Be careful, we're asking about JSON, not Python!

A. `{1001:{"name":"Evelyn", "rating":None}}`

B. `{1001:{"name":"Evelyn", "rating":null}}`

C. `{"1001":{"name":"Evelyn", "rating":None}}`

D. `{"1001":{"name":"Evelyn", "rating":null}}`

24. What will the following print?

```
team = players
team.pop(2)
print(len(players))
```

A. 0 B. 1 C. 2 D. 3 E. 4

25. What is printed?

```
percents = [0.9, 0.8, 0.7]
scores = percents * 100
print(scores)
```

A. `[0.9, 0.8, 0.7]`

B. `[90.0, 80.0, 70.0]`

C. `[90, 80, 70]`

D. A very long list containing hundreds of entries

E. An error, because you may not multiply a list by an integer

Movies

When considering each question, assume the initial code executed is as follows (if a question contains code that modifies the objects, those changes **should not** be considered in other questions):

```
movies = [
  {"title": "A", "year": 18, "style": "short", "genres": ["g1"]},
  {"title": "B", "year": 18, "style": "long",  "genres": ["g2"]},
  {"title": "C", "year": 19, "style": "short", "genres": ["g3"]},
  {"title": "D", "year": 19, "style": "long", "genres": ["g1", "g2", "g3"]}
]
```

26. What does the following evaluate to? `movies[-1]["genres"][0]`

A. `"genres"` B. `"g1"` C. `["g1"]` D. `["g2"]` E. `["g1","g2","g3"]`

27. What does the following evaluate to?

```
movies[3]["genres"].index(movies[1]["genres"][0])
```

A. `0` B. `1` C. `2` D. `"g1"` E. `"g2"`

28. What does the following evaluate to?

```
movies[0]["genres"] + movies[1]["genres"]
```

A. `"g3"`

B. `"g1g2"`

C. `["g1", "g2"]`

D. `[["g1"], ["g2"]]`

E. `{"genres": [["g1"], ["g2"]]}`

29. Which of the following will `result` possibly refer to?

```
result = []
for x in movies[-1]:
    result.append(x)
```

A. `["g", "e", "n", "r", "e", "s"]`

B. `["genres", "style", "year", "title"]`

C. `["g1", "g2", "g3"]`

D. `["D", 19, "long", "g1", "g2", "g3"]`

E. `["D", 19, "long", ["g1", "g2", "g3"]]`

30. What will be printed? Be careful!

```
buckets = {}
bucket = []
for m in movies:
    key = m["year"]
    if not key in buckets:
        buckets[key] = bucket
    buckets[key].append(m)
print(len(buckets[18]), len(buckets[19]))
```

A. 0 0 B. 2 0 C. 2 2 D. 0 4 E. 4 4

31. What will `counts` contain? Be careful!

```
counts = {}
for m in movies:
    for genre in m["genres"]:
        if not genre in counts:
            counts[genre] = 1
        counts[genre] += 1
```

A. {}

B. {"g1":1, "g2":1, "g3":1}

C. {"g1":3, "g2":3, "g3":3}

D. {"g1":3}

E. {"g1":6}

32. What would the following code print?

```
movies.append(movies.pop(0))
print(movies[1].get("title", None)]
```

A. A B. B C. C D. D E. None

33. Assuming the following succeeds, what must be the type of `mov`?

```
print(mov.title, mov.year)
```

A. str B. dict C. list D. tuple E. namedtuple

34. How many lines will be printed to the screen?

```
for i in range(10):
    try:
        m = movies[i]
        print(m["title"])
        year = m["DATE"]
        print(year)
    except:
        pass
```

A. 0

B. 4

C. 8 (not really correct, but we're accepting it)

D. 10

E. 20

35. What will the following code print?

```
import copy
x = [{"movie":"Wonder Woman", "stars":4.5},
    {"movie":"Captain Marvel", "stars":5},
    {"movie":"Ghost Busters", "stars":4}]
y = copy.deepcopy(x)
x[0]["movie"] = "Akira"
print(y[0]["movie"])
```

A. `Wonder` B. `Wonder Woman` C. `Akira` D. `A` E. `4.5`

(Blank Page)