You have a captain's log due before 2024-04-21 (in 1 day)! Log it now!
(/captain_logs/5596018/edit)

Evaluation quiz correction

Evaluation Quiz: Evaluation #3

Date: 2023-10-25

Status: Done

Duration: 11 minutes

Score: 94.12%

 # "I don't know": 0

 # Success: 16

 # Fail: 1

Responses

0. Which symbol should I use to redirect the error output to the standard output?

Score: 1.0

- [x] **2>&1**
- [ ] 1>&2
- [ ] 2>
- [ ] I don't know

1. What do these lines print?

```
>>> a = { 'id': 89, 'name': "John", 'projects': [1, 2, 3, 4], 'friends': [ { 'id': 82, 'name':
"Bob" }, { 'id': 83, 'name': "Amy" } ] }
>>> a.get('friends')[-1].get("name")
```

Score: 1.0

- [ ] 89
- [ ] [{'id':82, 'name':"Bob"}, {'id':83, 'name': "Amy"}]
- [x] **'Amy'**
- [ ] 'Bob'
- [ ] Nothing
- [ ] I don't know

2. What do these lines print?

```
a = 12
if a > 2:
    if a % 2 == 0:
        print("Tech")
    else:
        print("C is fun")
else:
    print("School")
```

Score: 1.0

- [x] **Tech**
- [ ] C is fun
- [ ] School
- [ ] I don't know

3. What do these lines print?

```
>>> def my_function(counter=89):
>>>     print("Counter: {}".format(counter))
>>>
>>> my_function(12)
```

Score: 1.0

(/)

- [x] ~~**Counter: 12**~~
- [ ] Counter: 89
- [ ] Counter: 101
- [ ] I don't know

4. What do these lines print?

```
>>> def my_function(counter=89):
>>>     return counter + 1
>>>
>>> print(my_function())
```

Score: 1.0

- [ ] 1
- [ ] 89
- [x] **90**
- [ ] 891
- [ ] I don't know

5. What do these lines print?

```
>>> a = [1, 2, 3, 4]
>>> a[2] = 10
>>> a
```

Score: 1.0

- [ ] [1, 2, 3, 4]
- [ ] [1, 10, 3, 4]
- [x] **[1, 2, 10, 4]**
- [ ] [1, 2, 10, 10]
- [ ] I don't know

6. What is a circular import in Python?

(/)

Score: 1.0

- [x] **When two or more modules are dependant on each other.**
- [ ] When you import a module for calculating dimensions for circles.
- [ ] When one module imports multiple other modules.
- [ ] I don't know

7. How many bytes will this statement allocate on a 64 bit machine?

```
malloc(sizeof(char) * 4)
```

Score: 1.0

- [x] **4**
- [ ] 8
- [ ] 12
- [ ] 16
- [ ] I don't know

8. In a doubly linked list, what's the "head" of a linked list?

Score: 1.0

- [ ] It's the node with the pointer to the next node equals to `NULL`
- [x] **It's the node with the pointer to the previous node equals to `NULL`**
- [ ] I don't know

9. What does this print?

```
>>> print("{:d} Mission street, {}".format(972, "San Francisco"))
```

Score: 1.0

- [ ] "972 Mission street, San Francisco"
- [ ] 72 Mission street, San

- [x] **972 Mission street, San Francisco**
- [ ] San Francisco Mission street, 972
- [ ] I don't know

10. What do these lines print?

```
for i in range(2, 10, 2):
    print(i, end=" ")
```

Score: 1.0

- [ ] 2 3 4 5 6 7 8 9 10
- [ ] 2 3 4 5 6 7 8 9
- [ ] 4 6 8 10 12 14 16 18
- [x] **2 4 6 8**
- [ ] I don't know

11. In a doubly linked list, what are possible directions to traverse it?

(select all possible answers)

Score: 1.0

- [x] **Forward**
- [x] **Backward**
- [ ] I don't know

12. What is the `unistd` symbolic constant for the standard output?

Score: 1.0

- [ ] STDIN_FILENO
- [x] **STDOUT_FILENO**
- [ ] STDERR_FIELNO
- [ ] I don't know

13. Which line of code will create a list of every other number from 0 to 10 in reverse in Python?

Score: 0.0

- [x] **list(range(10, 0, -2))**
- [x] **array(range(10, 0, -2))**
- [ ] list(range(0, 10, -2))
- [ ] array(10, 0, 2))
- [ ] I don't know

14. You're standing in line at a grocery store, which data type best represents this situation?

Score: 1.0

- [x] **Queue**
- [ ] Array
- [ ] Dictionary
- [ ] Stack
- [ ] I don't know

15. What's wrong with the following C code to get the nth node of a linked list?

Select all correct answers.

(/)

```
#include "lists.h"
/**
 * get_nodeint_at_index - finds nth node of a listint_t list
 * @head: list to evaluate
 * @index: index of node to find
 *
 * Return: node found at index (SUCCESS), NULL if node does not exist
 **/

listint_t *get_nodeint_at_index(listint_t *head, unsigned int index)
{
        unsigned int i;
        listint_t *ptr;

        if (head == NULL)
                return (NULL);

        ptr = head;
        i = 0;

        while (i < index)
        {
                ptr = ptr->next;
                i++;
        }

        return (ptr);
}
```

Score: 1.0

- [x] **There is no check for if `ptr->next` is `NULL` before moving `ptr`**
- [ ] The function should not return `NULL` if `head` is not found.
- [x] **If `index` is out of range, the program should return `NULL`**
- [ ] Nothing is wrong
- [ ] I don't know

16. What does this print?

```
>>> a = "Python is cool"
>>> print(a[7:-5])
```

Score: 1.0

- [ ] on

- [ ] nohtyP
- [ ] Python
- [ ] si
- [x] **is**
- [ ] I don't know