THE UNIVERSITY OF MELBOURNE

COMP10001 - Sem 2 2024 - Week 6

Foundations of Computing

Daksh Agrawal

Checkpoint 1

Project 1 Out

GET STARTED NOW

Grok Tips

Assume that seq is a list. Write one line of code to:

Sort seq in-place

seq.sort()

Assign the sorted version of seq to a new variable new_seq. Do not change the original list.

new_seq = sorted(seq)

Add the string "hi" to the end of seq.

seq.append("hi")

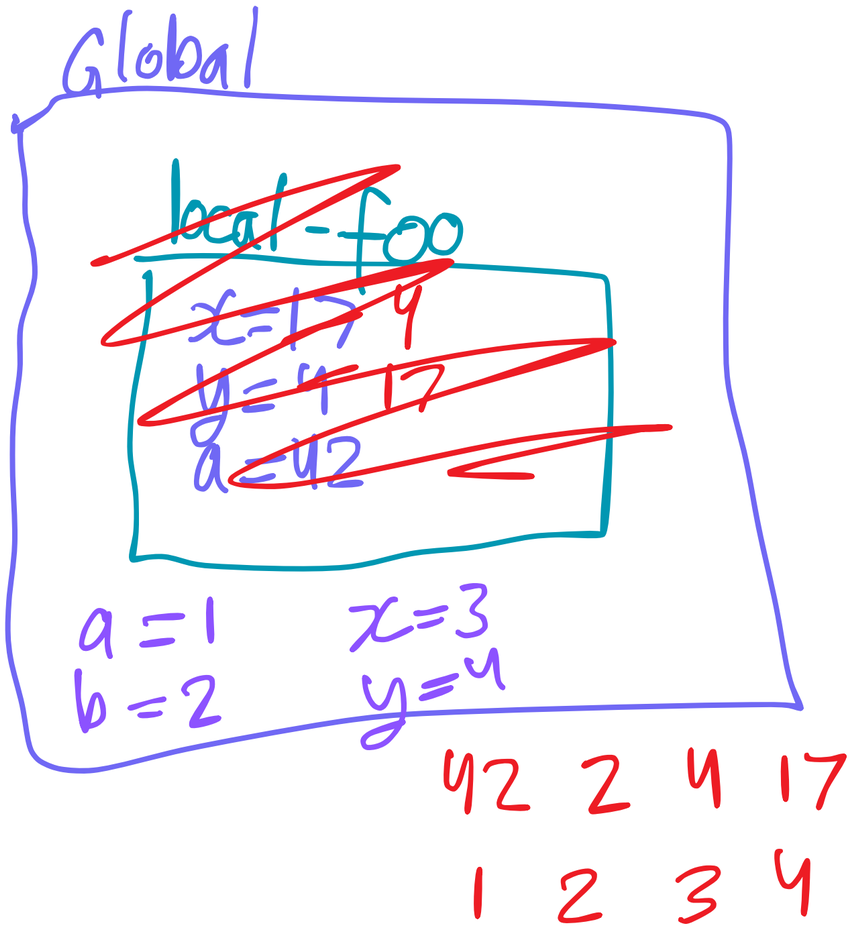
new_seq = seq.copy()
new_seq.append("hi")

new_seq = seq + ["hi"]

What is the output of the following code? Classify the variables by which namespace they belong in.

```
def foo(x, y):
    a = 42
    x, y = y, x
    print(a, b, x, y)

a, b, x, y = 1, 2, 3, 4
foo(17, 4)
print(a, b, x, y)
```



What is the output of this code? Why?

```
def mystery(x):
    x.append(5)
    x[0] += 1
    print("mid-mystery:", x)


my_list = [1,2]
print(my_list)
mystery(my_list)
print(my_list)
mystery(my_list.copy())
print(my_list)
```

my_list → [2,2,5]

x → [3,2,5,5]

[1,2]

mid-mystery [2,2,5]

[2,2,5]

mid-mystery [3,2,5]

[2,2,5]

variable → value

lists, dict → reference

Compare the two functions below. Are they equivalent? Why would we prefer one over the other?

```
def noletter_1(words, letter='z'):
    for word in words:
        if letter in word:
            return False
    return True


1 usage  new *
def noletter_2(words, letter='z'):
    no_z = True
    for word in words:
        if letter in word:
            no_z = False
    return no_z


wordlist = ['zizzer'] + ['aardvark'] * 10_000_000
print(noletter_1(wordlist))
print(noletter_2(wordlist))
```

Write a Python function `find_ints(text)` that takes a (possibly empty) Python string `text` and returns a (possibly empty) list of the word locations at which integers occur. A “word” is defined as a consecutive sequence of non-whitespace characters, and an “integer” is defined as a word that is either completely made up of digits, or is a single + or - sign, and then nothing but digits. Word positions within text are counted from one.

For example:

- **`find_ints("Ints -34 there and here +551 but not here88")`** should return **`[2, 6]`**
- **`find_ints("No integers here99 88-77 or there")`** should return **`[]`**
- **`find_ints("+18 and -777 and 666 are all 3 integers")`** should return **`[1, 3, 5, 8]`**

Hint: if you get stuck, try seperating the first character from the rest for each word.

```python
def find_ints(text):
    words = text.split()
    result = []

    for i, word in enumerate(words):
        if word[0] == "+" or word[0] == "-":
            word = word[1:]
        if word.isdigit():
            result.append(i + 1)

    return result
```

Write a function which takes two lists as input and returns a list containing the numbers which they both have in common.
in_common([1, 2, 4], [3, 4, 5]) should return **[4]**.

```
def in_common(lst_1, lst_2):
    set_1 = set(lst_1)
    set_2 = set(lst_2)
    inter = set_1 & set_2        set_1.intersection(set_2)
    list_inter = list(inter)
    return list_inter
```

```
def in_common(lst_1, lst_2):
    return list(set(lst_1) & set(lst_2))
```

Write a function which takes a dictionary and returns a sorted list containing the unique values in that dictionary.
unique_values({'a': 1, 'b': 0, 'c': 0}) should return **[0, 1]**.

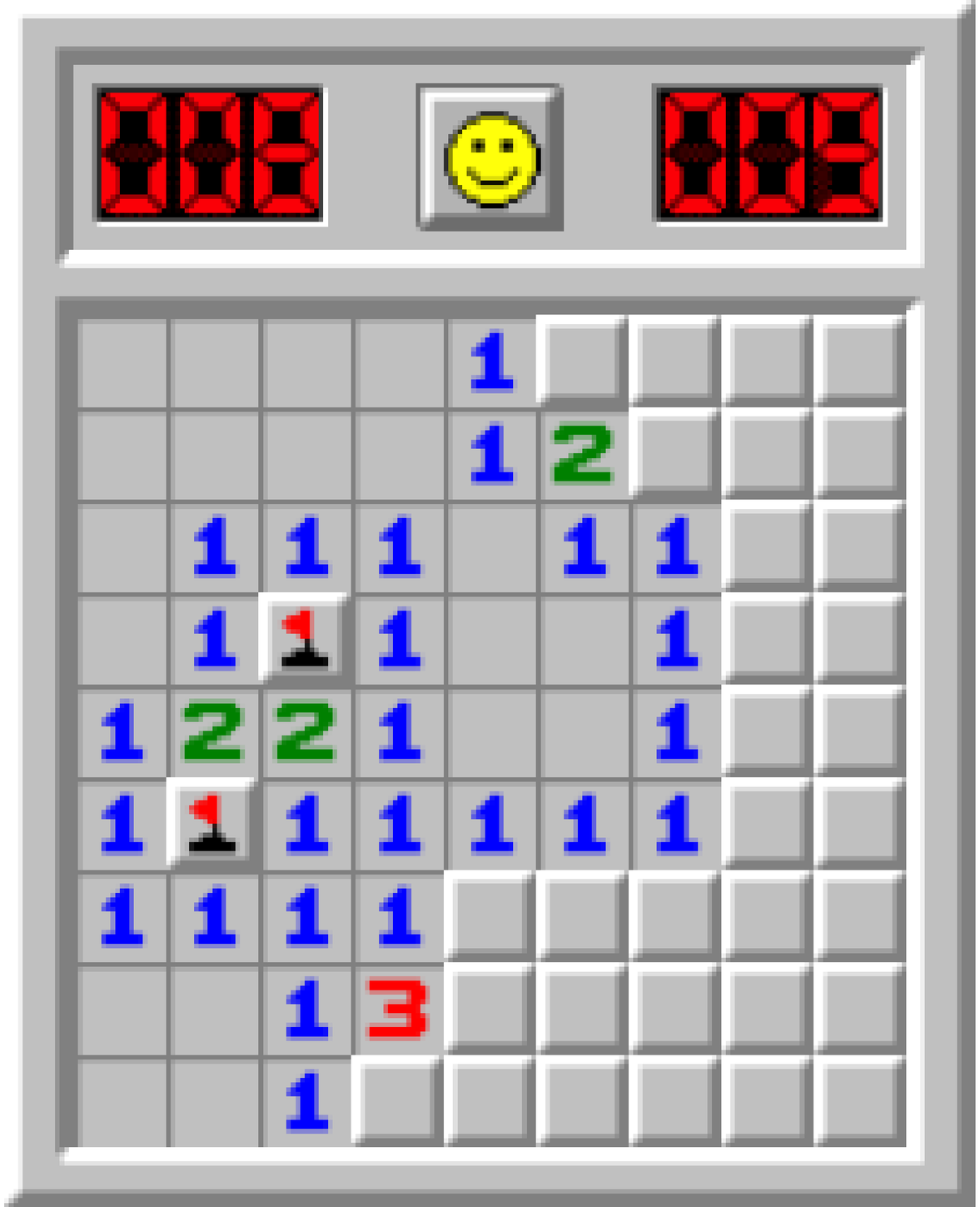
```
def unique_values(d):
    vals = d.values()
    unique_vals = list(set(vals))
    unique_vals.sort()
    return unique_vals
```

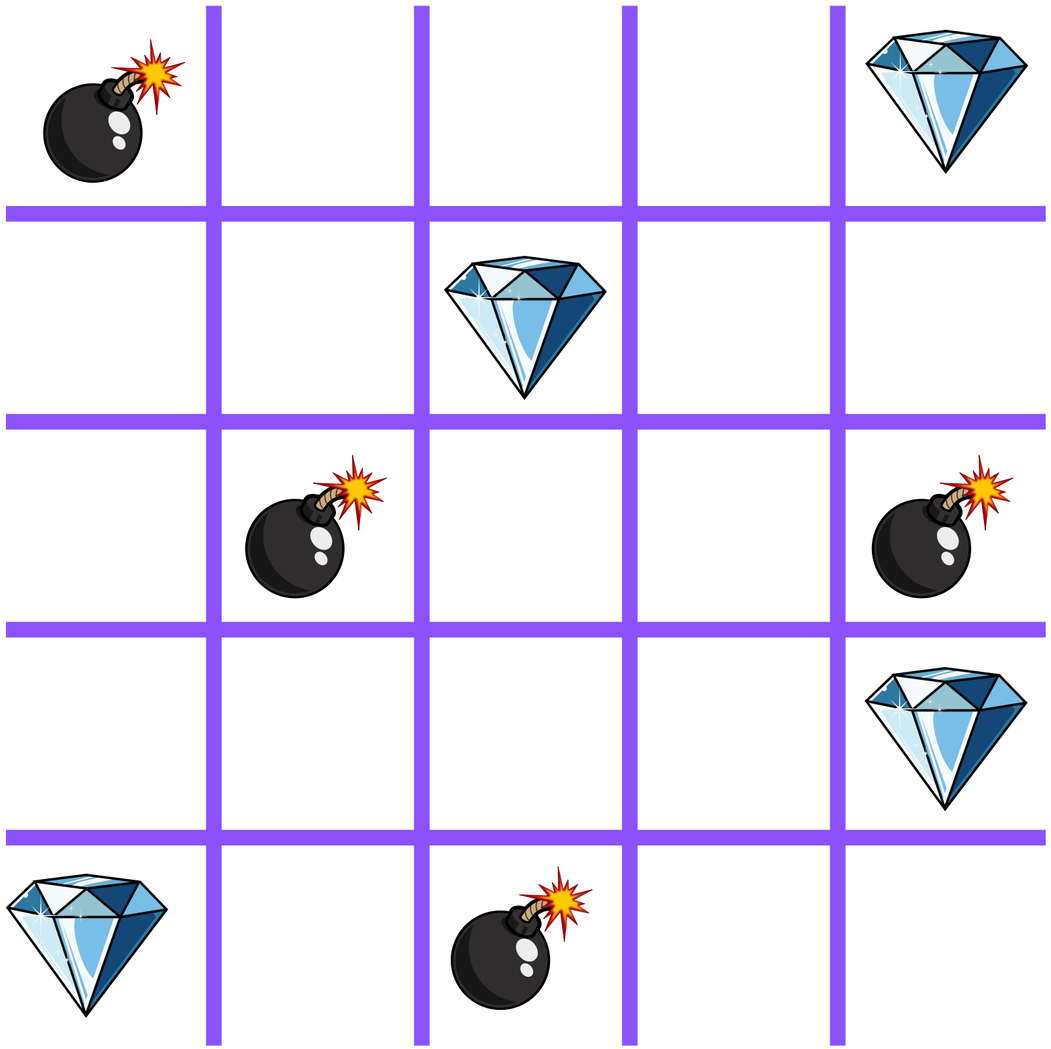
Write a function which takes a list of words and checks whether any of those words are palindromes (spelled the same way backwards as forwards, like “kayak”). It should return True if there are any palindromes and False if there are none. Use a timely return to save some time!

any_palindrome(['kayak', 'motorbike', 'scooter']) should return **True**.

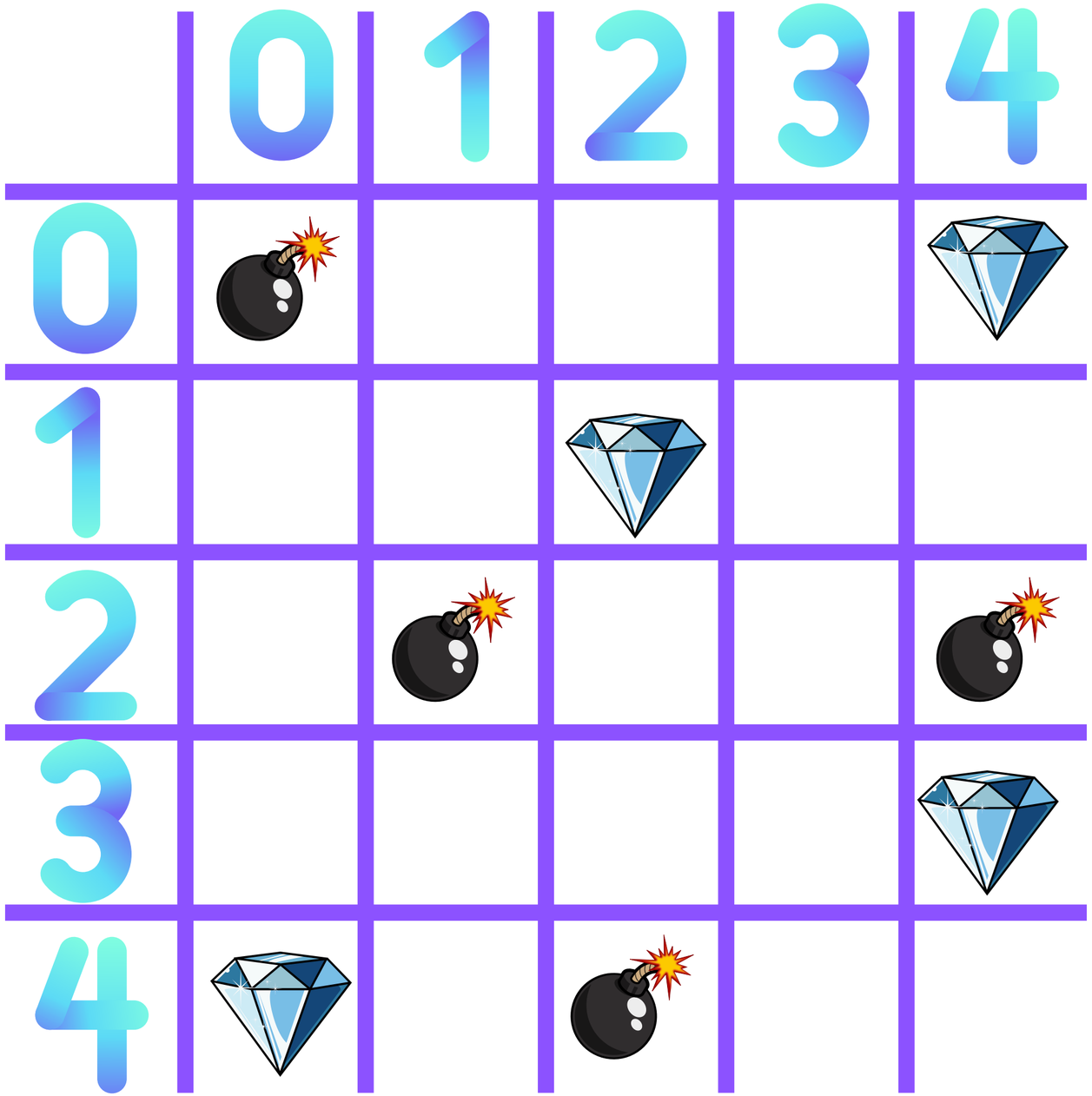
```
14  def any_palindrome(words):
15      """This function checks if the list of
16      words contain a palindrome"""
17
18      # This is a comment
19      for word in words:
20          if word == word[::-1]:
21              return True
22      return False
```


MAKE You MINE
DARLIN!!





```
[
  ["b", "e", "e", "e", "d"],
  ["e", "e", "d", "e", "e"],
  ["e", "b", "e", "e", "b"],
  ["e", "e", "e", "e", "d"],
  ["d", "e", "b", "e", "e"]
]
```



0 2

3 2

4 0

3 1

2 1

One diamond cannot be mined twice

```python
def contains_diamonds(board):
    for row in board:
        if "d" in row:
            return True
    return False
```

```python
def make_you_mine(board):
    """
    Game Minesweeper Simplified
    """
    num_diamonds = 0
    while True:
        u = input("Where mine?")
        coords = u.split()
        x, y = coords
        x = int(x)
        y = int(y)

        item = board[x][y]
        if item == "e":
            continue
        elif item == "b":
            print("GAME OVER")
            break
        else:
            num_diamonds += 1
            print("You mined", num_diamonds, "diamonds")
            board[x][y] = "e"
            if not contains_diamonds(board):
                print("YOU WIN", num_diamonds)
                break
```

Anonymus Feedback