THE UNIVERSITY OF MELBOURNE

COMP10001 - Sem 2 2024 - Week 7

Foundations of Computing

Daksh Agrawal

Comments

""" Docstrings """

```python
def find_ints(text):
    """This function finds the indices of the
    words that are integers in a given 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


new *
def any_palindrome(words):
    """This function checks if the list of
    words contain a palindrome"""

    # This is a comment
    for word in words:
        if word == word[::-1]:
            return True
    return False
```

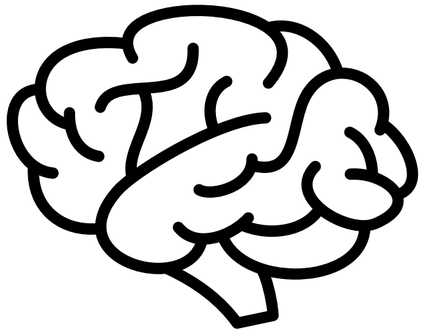
```
def any_palindrome(words):
""" this function takes in
a list of words and returns
true if any of them are
a palindrome """
...
```

Past Project Investigation

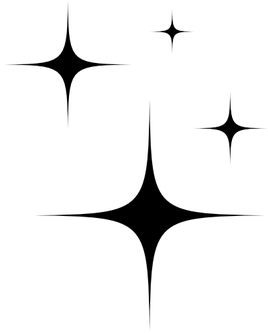
Write a function **get_species_richness()** that calculates the species richness of a habitat, based on a series of observations of various bird species. The function takes one argument: **observed_list**, a list of independent observations of bird species. The function should return a tuple consisting of:

- the species richness, calculated as the number of different species observed
- an alphabetically sorted list of the species that were observed

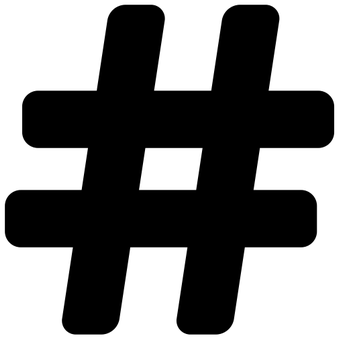
Making Rubric (Lite)



Sensible Approach?
Oversimplified?
Overcomplicated?



PEP8 Guidelines?
Good Variable Names?
“Magic Numbers”?



Comments?
Docstrings?

```python
def get_species_richness(observed_list):
    # easy approach is to convert to a set
    species_observed = set(observed_list)
    return len(species_observed), sorted(species_observed)
```

Sensible?	
Readability?	
Commenting?	

```python
def get_species_richness(observed_list):
    """ Takes a list of strings representing species observed. Returns a
    tuple containing the species richness (an int) and a sorted list
    containing names of each species. """
    observed_birds = []
    # constructs unique list of observed bird species
    for bird in observed_list:
        if bird not in observed_birds:
            observed_birds.append(bird)
    # counts number of species and sorts by unicode sort order
    return (len(observed_birds), sorted(observed_birds))
```

Sensible?	Readability?	Commenting?

```python
# returns b and sorted dictionary new *
def get_species_richness(l):
    # create a dictionary
    dict1 = {}
    b = 0
    # loop
    for i in range(1, len(l) + 5):
        # loop
        for c in l:
            if not c in dict1:
                # increment by i
                b += i
                dict1[c] = 0
        # increment by 1
        dict1[c] += 1
    """return"""
    return (b, sorted(dict1.keys()))
```

Sensible?	
Readability?	
Commenting?	

Corner Cases

Test Cases

```
7   # this only runs when pressing 'run' or 'terminal', not when 'marking'
8   if __name__ == "__main__":
9       inputs = [
10          ['cockatoo', 'magpie'],
11          # TODO: add your test case inputs here
12      ]
13      expected_outputs = [
14          (2, ['cockatoo', 'magpie']),
15          # TODO: add the expected outputs of your test cases here
16      ]
17      for test_input, expected in zip(inputs, expected_outputs):
18          print("expected:", expected)
19          print("result: ", get_species_richness(test_input))
```

Long, Good Code

```python
def favourite_animal(ballots):
    """Function to take in ballots, count votes and return the favourite animals
    tally = {}

    # create a freq count dict for animal votes
    for animal in ballots:
        if animal in tally:
            tally[animal] += 1
        else:
            tally[animal] = 1

    # create a list of the most fav animals from our tally
    most_votes = max(tally.values())
    favourites = []
    for animal, votes in tally.items():
        if votes == most_votes:
            favourites.append(animal)

    return favourites
```

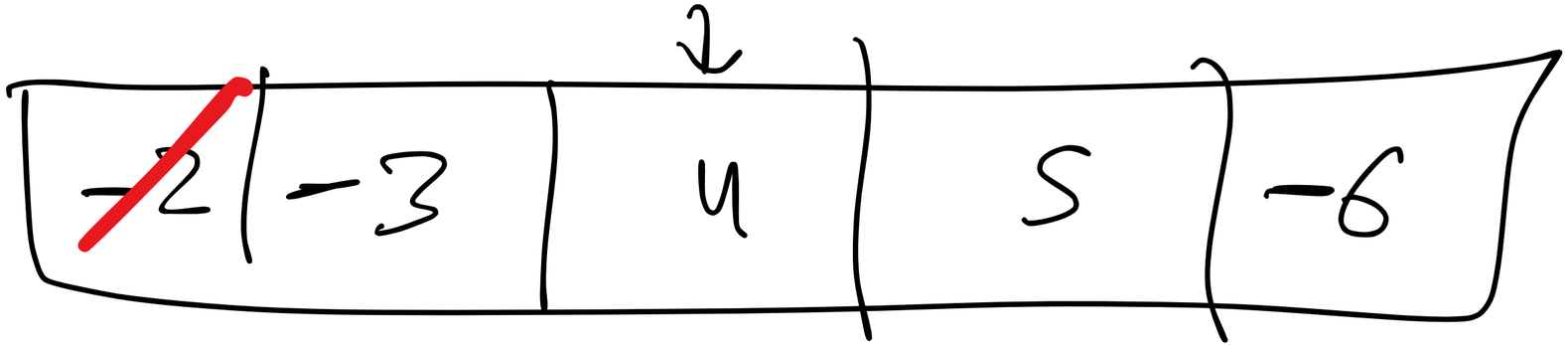
```
1  a = int(input("Enter days: "))
2  b = a * 24
3  c = b * 60
4  d = c * 60
5  print("There are", b, "hours", c, "minutes", d, "seconds in", a, "days")
```

a = num_days
b = num_hours
c = num_minutes
d = num_seconds

```
word = input("Enter text: ")
x = 0
vowels = 0
word_2 = word.split()
for word_3 in word_2:
    x += 1
    for word_4 in word_3:
        if word_4.lower() in "aeiou":
            vowels += 1
if vowels/x > 0.4:
    print("Above threshold")
```

```python
def remove_negative(nums):
    for num in nums:
        if num < 0:
            nums.remove(num)
```

```
def remove_negative(nums):
    negative_list = []
    for num in nums:
        if num<0:
            negative_list.append(num)
    for num in negative_list:
        nums.remove(num)
```



Paper Programming

Write a function **check_parens()** that takes a string text and checks that the parentheses are valid (i.e. after opening, at some point in the text the parenthesis is closed). For example, **check_parens("(())()")** should return **True** and **check_parens("())")** should return **False**.

```
def check_parens(text):
    """ Checks if text has a valid parenthesis order""""
    # checks if depth goes negative at any point
    depth=0
    for char in text:
        if char == "(":
            depth +=1
        else:
            depth -=1
        if depth <0:
            return False
    return depth ==0
```

{ }

[ ]

( )

[( )]

Write a function **gen_pascal()** that takes an integer **num_rows** (≥1) and returns the first **num_rows** rows of Pascal's triangle as a list of lists of ints. In Pascal's triangle, each number is the sum of the two numbers directly above it. For example, **gen_pascal(5)** should output **[[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1]]**

Safe and Sound

Slides

With Annotations

Question 1 (8 marks)

Give the value of each of the following expressions. If you think the expression is not legal according to the rules of Python, write "Error" in the box.

a. `16 // 4 - 2 % 3`

4 − 2 = 2 int

b. `str(2) * 3`

"2" * 3 = "222" str

c. `(6, 3, 4) + tuple("1")`

(6, 3, 4, "1")

d. `max([9, 3, 6, 5][1:3])`

max([3, 6]) = 6 int

Question 2 (3 marks)

Suppose that `nums` is a non-empty list of integers. Give a single Python assignment statement that assigns `False` to `result` if all the values in `nums` are the same, and assigns `True` to `result` if they are not all the same.

```
result = len(set(nums)) != 1
```

Question 3 (3 marks)

Suppose that `text` is a Python string whose length is an odd number, e.g., 1 or 3 or 7 etc. Give a single Python assignment statement that assigns to `middle` the character that appears at the mid-point of the string in `text`, e.g., if `text` is `"cat"` then `middle` should be `"a"`.

```
middle = text[ len( text)//2 ]
```

Question 4 (3 marks)

Suppose that `people` is a Python dictionary in which each key is a string denoting a person's name, and the value is an integer denoting that person's age. Give a single Python assignment statement that assigns to `oldest` the age of the oldest person in `people`.

```
oldest= max(people.values())
```

Question 5 (3 marks)

Suppose that `str1` is a non-empty string. Give a single Python assignment statement that assigns to `newstr` a string that contains all the characters of `str1` that appear at even-valued indexes, e.g., if `str1` is `"boat"` then `newstr` should be `"ba"`.

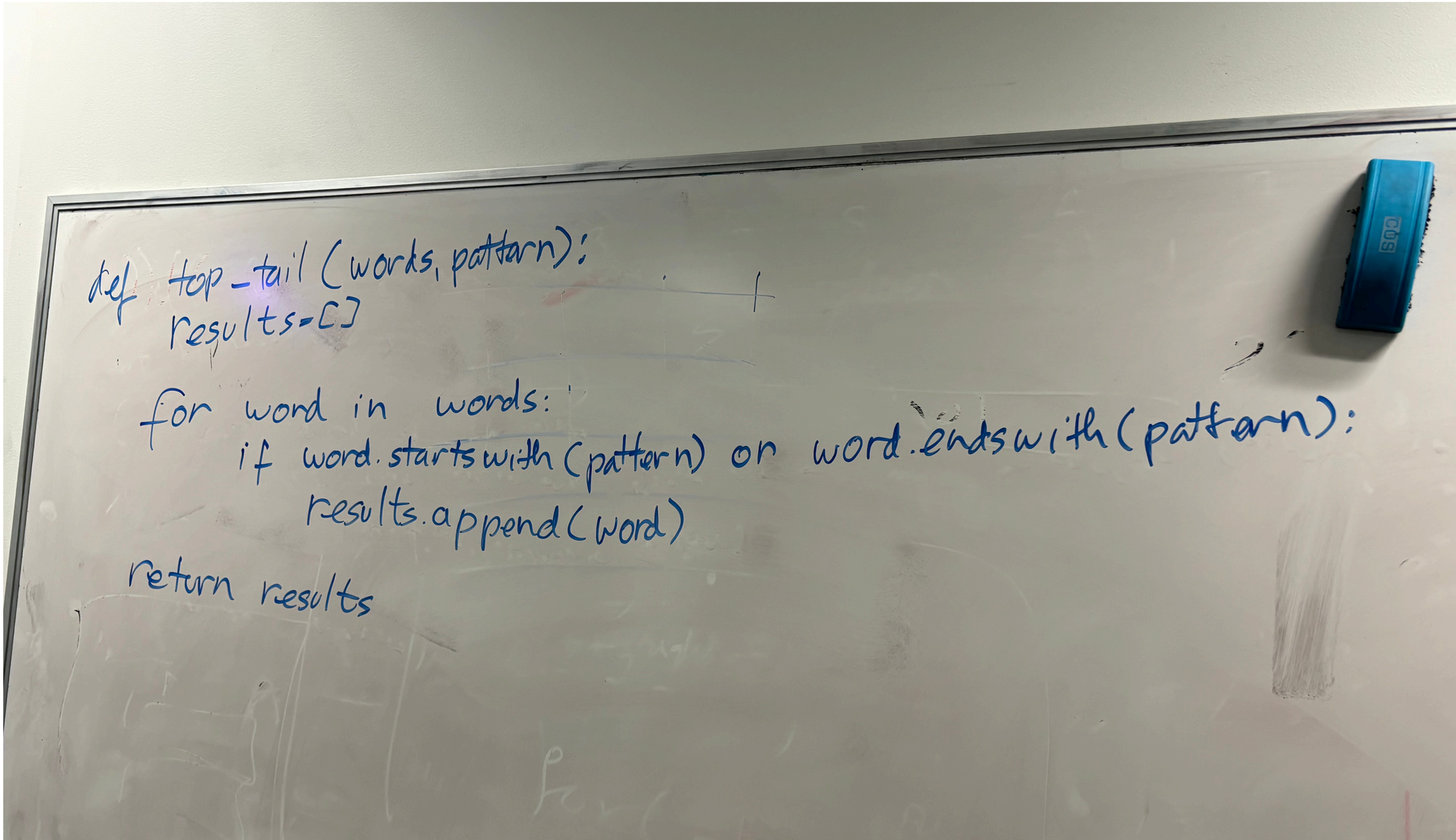
```
newstr = str1[::2]
```

Question 6 (20 marks)

Write a Python function `top_tail(words, pattern)` that takes a (possibly empty) list of Python strings `words` and returns a (possibly empty) list of the strings in `words` that either start or end with the string in `pattern`. The strings in the returned list should be in the same relative order that they appeared in `words`. The length of each string in `words` is greater than or equal to the length of the string in `pattern`.

Here are three possible execution sequences:

```
>>> results = top_tail(["in", "vein"', "he", "inked", "the", "tin"], "in")
>>> print(results)
["in", "vein", "inked", "tin"]
>>> results = top_tail(["ink", "vein", "her", "inked", "the", "tin"], "ing")
>>> print(results)
[]
>>> results = top_tail(["oops", "cooked", "boo", "hoo"], "oo")
>>> print(results)
["oops", "boo", "hoo"]
```


def top_tail(words, pattern):
results=[]
for word in words:
if word.startswith(pattern) or word.endswith(pattern):
results.append(word)
return results