THE UNIVERSITY OF MELBOURNE

COMP10001 - Sem 2 2024 - Week 7

Foundations of Computing

Daksh Agrawal

Comments

""" Docstrings """

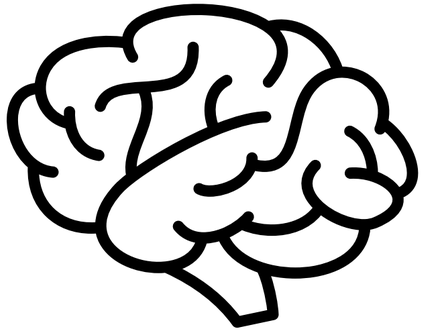
```python
def find_ints(text):
    """This function finds the indices of the
    words that are integers in a given text"""

    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
```

Past Project Investigation

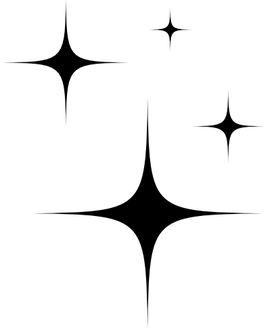
Write a function **get_species_richness()** that calculates the species richness of a habitat, based on a series of observations of various bird species. The function takes one argument: **observed_list**, a list of independent observations of bird species. The function should return a tuple consisting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

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

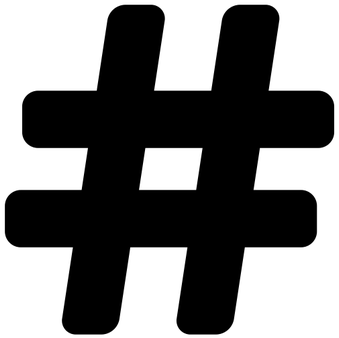
Making Rubric (Lite)



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

```python
# this only runs when pressing 'run' or 'terminal', not when 'marking'
if __name__ == "__main__":
    inputs = [
        ['cockatoo', 'magpie'],
        # TODO: add your test case inputs here
    ]
    expected_outputs = [
        (2, ['cockatoo', 'magpie']),
        # TODO: add the expected outputs of your test cases here
    ]
    for test_input, expected in zip(inputs, expected_outputs):
        print("expected:", expected)
        print("result: ", get_species_richness(test_input))
```

Long, Good Code

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

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

    # find the max num of votes and animals that achieved it
    most_votes = max(tally.values())
    favourites = []
    for animal, votes in tally.items():
        if votes == most_votes:
            favourites.append(animal)

    return favourites
```

```python
a = float(input("Enter days: "))
b = a * 24
c = b * 60
d = c * 60
print("There are", b, "hours", c, "minutes", d, "seconds in", a, "days")
```

```python
HOURS_IN_DAY=24
MINUTES_IN_HOUR=60
SECONDS_IN_MINUTE=60
num_days = float(input("Enterdays:"))
num_hrs = num_days * HOURS_IN_DAY
```

```python
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)
```

~~-3~~	-4	5	-1	3

?

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

    for i in neg_nums:
        nums.remove(i)
```

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
```

{ }

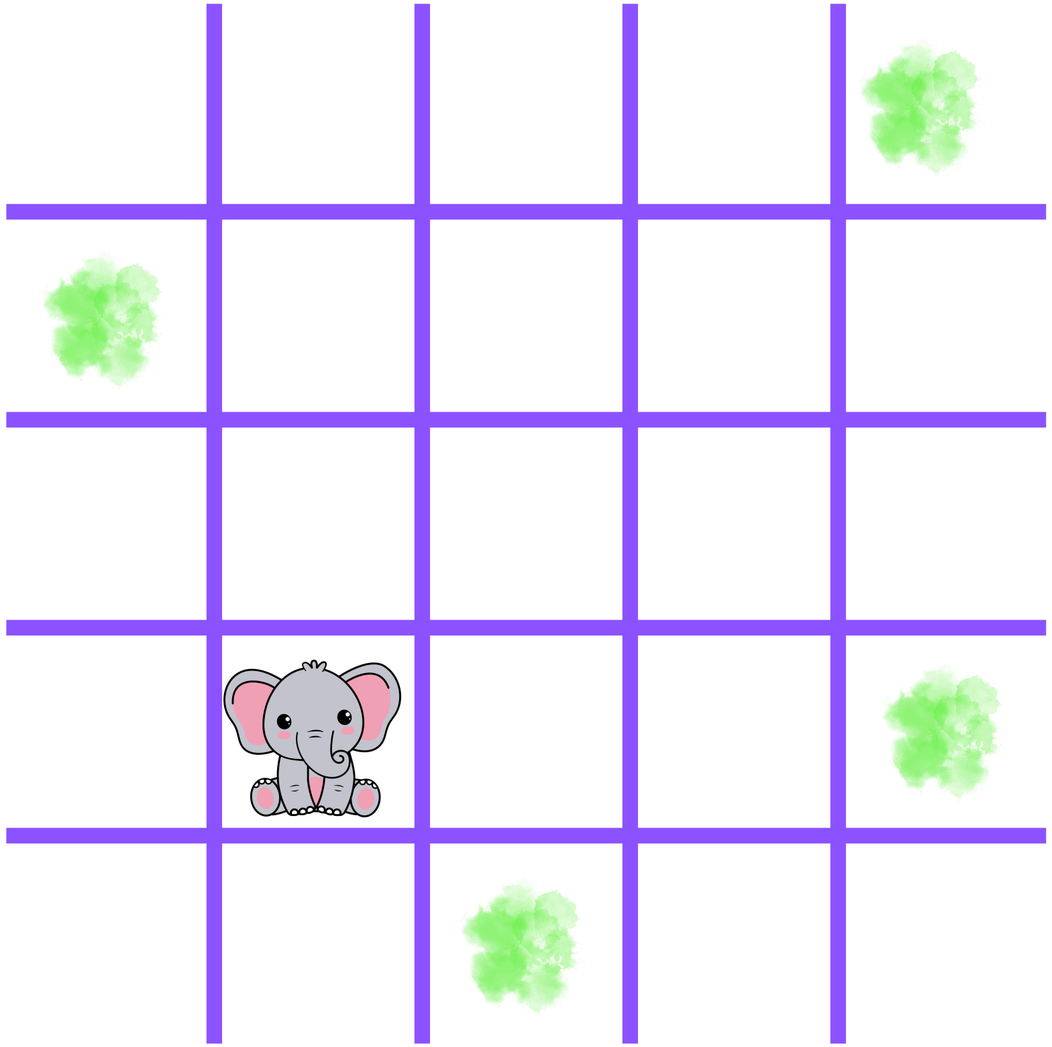
[ ]

( )

[( )]

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