Project Set: Lecture 6

Matthew Ellison
SPISE

July 12, 2019

1 Debug It!

Fix the following short programs (available at the github).

1.1 List Sort

```python
unsorted_list = [6,2,5,1,3]
sorted_list = unsorted_list.sort()
for i in range(len(unsorted_list)):
    print(sorted_list[i])
```

1.2 Printing Dots

```python
smallnumberfromuser = input('Enter a number between 0 and 20: ')
smallnumberfromuser = smallnumberfromuser * 2
for i in range(smallnumberframuser / 2)
    print('.')
```

1.3 Genotype-to-Phenotype Dictionary

```python
geno_to_pheno = {['long', 'long']: 'long', ['long', 'short']: 'long',
                 ['short', 'long']: 'long',['short','short']:'short'}
```

2   Testing...

Write a file to comprehensively test the following function (not yet implemented), then implement the function. Make sure it passes all your tests.

```python
1   def reverse_digits(number):
2       '''input is a number (int). The output should be the digits in reverse
3       order (e.g. 534 --> 435). In the case of a negative integer, the
4       negative sign should remain in the beginning of the reversed number
5       (e.g. -32 --> -23).
6       '''
```

3   Debug It! v2

Fix the following error-ridden longer program (available at the github).

```python
1   import time
2
3   difficulty=0
4   max_word_size=6
5   max_paragraph_size=10
6
7   lower_letters='abcdefghijklmnopqrstuvwxyz'
8   upper_letters='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
9   numerals='1234567890'
10  symbols='''!@#$%^&*()_+`~-=[]{}\|:;"'/?><.,'''
11  difficulty_ranking=[lower_letters,upper_letters,numerals,symbols]
12
13  def generate_letter():
14      level=random.randint(0,difficulty)
15      character=random.choice(difficulty_ranking[level])
16      return char
17
18  def generate_word():
19      word_size=random.randint(1,10)
20      output=''
21      for x in range(word_size):
22          output.append(generate_letter)
23      return output
24
25  def generate_paragraph():
26      par_size=random.randint(10,max_par_size)
27      output=''
28      for x in range(0,par_size):
29          output+=generate_word()
30          output+=' '
31      return output
32
33  def words_correct(attempt):
34      attempt_words=attempt.split(' ')
35      solution_words=solution.split(' ')
36      num_words_compare=min(len(attempt_words),len(solution_words))
37      words_correct=0
38      total_words=len(solution_words)
```

```
39        for word_idx in range(0,num_words_compare):
40            if attempt_words[word_idx]==solution_words[word_idx]:
41                words_correct+=1
42        return len(attempt_words),words_correct,total_words
43
44 def play_game():
45     playing=True
46     print('Get Ready to Type!\n')
47     time.sleep(1)
48     while playing:
49         new_par=generate_paragiph()
50         print(new_par+'\n')
51
52         start=time.time()
53         submission=input('type the above!\n')
54         end=time.time()
55
56         total_time=(end-start)/60
57         words_typed,word_score,total_words=words_correct(submission,new_par)
58         word_accuracy=round(word_score/total_words,3)
59         wpm=round(words_typed/total_time,2)
60
61         print('you typed at',wpm,'wpm with an accuracy of',word_accuracy,'!\n')
62
63         keep_going=input('keep playing? (y/n)\n')
64         if keep_going!='y':
65             playing=True
66
67 play_game()
```