testing.py

*"""  
CP1404/CP5632 Practical  
Testing demo using assert and doctest  
"""***import** doctest  
**from** car **import** Car  
  
  
**def** repeat\_string(s, n):  
 *"""Repeat string s, n times, with spaces in between."""* **if** n == 1:  
 **return** s  
 **else**:  
 **return** repeat\_string(s, n - 1) + **' '** + s  
  
  
**def** is\_long\_word(word, length=5):  
 *"""  
 Determine if the word is as long or longer than the length passed in  
 >>> is\_long\_word("not")  
 False  
 >>> is\_long\_word("supercalifrag")  
 True  
 >>> is\_long\_word("Python", 6)  
 True  
 """* **return** len(word) >= length  
  
  
**def** run\_tests():  
 *"""Run the tests on the functions."""  
 # assert test with no message - used to see if the function works properly* **assert** repeat\_string(**"Python"**, 1) == **"Python"** *# the test below should fail* **assert** repeat\_string(**"hi"**, 2) == **"hi hi"** *#* ***TODO: 1. fix the repeat\_string function above so that it passes the failing test*** *# Hint: "-".join(["yo", "yo"] -> "yo-yo"  
  
 # assert test with custom message,  
 # used to see if Car's init method sets the odometer correctly  
 # this should pass (no output)* test\_car = Car()  
 **assert** test\_car.odometer == 0, **"Car does not set odometer correctly"  
 assert** test\_car.fuel == 0, **"Car does not set fuel correctly"** *#* ***TODO: 2. write assert statements to show if Car sets the fuel correctly*** *# Note that Car's \_\_init\_\_ function sets the fuel in one of two ways:  
 # using the value passed in or the default  
 # You should test both of these* test\_car = Car(fuel=10)  
  
  
**def** phrase\_to\_sentence(phrase):  
 *"""  
 Modify phrase to sentence, capitalize the first letter and ending with'.'  
 >>> phrase\_to\_sentence('hello')  
 'Hello.'  
 >>> phrase\_to\_sentence("that's insane")  
 "That's insane."  
  
 """* sentence = phrase.capitalize()  
 **if** sentence[-1] != **'.'**:  
 sentence += **'.'  
 return** sentence  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 run\_tests()  
  
*#* ***TODO: 3. Uncomment the following line and run the doctests****# (PyCharm may see your >>> doctest comments and run doctests anyway.)*doctest.testmod()  
  
*#* ***TODO: 4. Fix the failing is\_long\_word function****# (don't change the tests, change the function!)  
  
#* ***TODO: 5. Write and test a function to format a phrase as a sentence,****# starting with a capital and ending with a single full stop.  
# Important: start with a function header and just use pass as the body  
# then add doctests for 3 tests:  
# 'hello' -> 'Hello.'  
# 'It is an ex parrot.' -> 'It is an ex parrot.'  
# and one more you decide (one that is valid!)  
# test this and watch the tests fail  
# then write the body of the function so that the tests pass*

wiki.py

**import** wikipedia  
  
  
**def** main():  
 **while True**:  
 search\_phrase = input(**">>>Enter a phrase: "**)  
 **if** search\_phrase != **''**:  
 **try**:  
 search\_title = wikipedia.search(search\_phrase)[0]  
 print(search\_title)  
 print(wikipedia.summary(search\_title))  
 search\_page = wikipedia.page(search\_title)  
 print(search\_page.url)  
 **except** IndexError:  
 print(**'No result'**)  
 **except** wikipedia.exceptions.DisambiguationError **as** e:  
 print(e.options)  
  
 **else**:  
 **break  
  
  
if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

app.py:

**from** flask **import** Flask  
  
app = Flask(\_\_name\_\_)  
  
  
@app.route(**'/'**)  
**def** hello\_world():  
 **return '<h1>Hello World :)</h1>'**@app.route(**'/greet'**)  
@app.route(**'/greet/<name>'**)  
**def** greet(name=**""**):  
 **return "Hello {}"**.format(name)  
  
  
**def** cel\_to\_fa(cel):  
 *"""convert Celsius value to Fahrenheit float"""* cel = float(cel)  
 **return** cel \* 9 / 5 + 32  
  
  
@app.route(**'/f'**)  
@app.route(**'/f/<c>'**)  
**def** f(c):  
 fa = str(cel\_to\_fa(c))  
 **return** fa  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 app.run()

flaskdemo.py

**from** flask **import** Flask, render\_template, request, redirect, url\_for, session  
**import** wikipedia  
  
app = Flask(\_\_name\_\_)  
*# Set the secret key. Keep this really secret:*app.secret\_key = **'IT@JCUA0Zr98j/3yXa R~XHH!jmN]LWX/,?RT'**@app.route(**'/'**)  
**def** home():  
 **return** render\_template(**"home.html"**)  
  
  
@app.route(**'/about'**)  
**def** about():  
 **return** render\_template(**"about.html"**)  
  
  
@app.route(**'/search'**, methods=[**'POST'**, **'GET'**])  
**def** search():  
 **if** request.method == **'POST'**:  
 session[**'search\_term'**] = request.form[**'search'**]  
 **return** redirect(url\_for(**'results'**))  
 **return** render\_template(**"search.html"**)  
  
  
@app.route(**'/results'**)  
**def** results():  
 search\_term = session[**'search\_term'**]  
 page = get\_page(search\_term)  
 **return** render\_template(**"results.html"**, page=page)  
  
  
**def** get\_page(search\_term):  
 **try**:  
 page = wikipedia.page(search\_term)  
 **except** wikipedia.exceptions.PageError:  
 *# no such page, return a random one* page = wikipedia.page(wikipedia.random())  
 **except** wikipedia.exceptions.DisambiguationError:  
 *# this is a disambiguation page, get the first real page (close enough)* page\_titles = wikipedia.search(search\_term)  
 *# sometimes the next page has the same name (different caps), so don't try the same again* **if** page\_titles[1].lower() == page\_titles[0].lower():  
 title = page\_titles[2]  
 **else**:  
 title = page\_titles[1]  
 page = get\_page(wikipedia.page(title))  
 **return** page  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 app.run()

Picture for flask run:















