5.

Projects / Life_Project

LP Sprint 2

Sprint 4 goals!

DS Label

TO DO 5 ISSUES	IN PROGRESS	DONE ✓
Sprint 4 - Investigate/plan subprocess implementation in LC 22FEB21 Sprint4 LP-18		
Sprint 4 - Implement and test functionality with Josh's CG 23FEB21 Sprint4 LP-19		
Sprint 4 - Evaluate code for smells 25FEB21 Sprint4 LP-20		
Sprint 4 - Refactor ALL code smells 26FEB21 Sprint4 LP-21		
Sprint 4 - Create video outlining communication between microservices 27FEB21 Sprint4 LP-22		

LC = life generator CG = Content Generator

6.

a.

https://github.com/asonawalla/FBEraser/blob/master/FBEraser.py

```python

if __name__ == '__main__':
    """
    Main section of script
    """
    # set up the command line argument parser
    parser = ArgumentParser(description='Delete your Facebook activity.  Requires Firefox')
    parser.add_argument('--wait', type=float, default=1, help='Explicit wait time between page loads (default 1 second)')
    args = parser.parse_args()

    # execute the script
    email = raw_input("Please enter Facebook login email: ")
    password = getpass.getpass()
    eraser = Eraser(email=email, password=password, wait=args.wait)
    eraser.login()
    eraser.go_to_activity_page()
```

```python
def execute_script():
    email = raw_input("Please enter Facebook login email: ")
    password = getpass.getpass()
    eraser = Eraser(email=email, password=password, wait=args.wait)
    eraser.login()
    eraser.go_to_activity_page()


def main():
    """
    Main section of script
    """
    # set up the command line argument parser
    parser = ArgumentParser(description='Delete your Facebook activity.  Requires Firefox')
    parser.add_argument('--wait', type=float, default=1, help='Explicit wait time between page loads (default 1 second)')
    args = parser.parse_args()

    execute_script()

if __name__ == '__main__':
    main()
```

Name: Long function – The main function is written into the if__name == '__main__': as well as the "function" currently is a Long Function of greater than 5 lines.

Refactor:

Broke the main function into its' own function and added the first 5 lines then created a second function to deal with the other 5 lines.

b.

https://github.com/flatbean/helloworld/blob/flatbean-patch-3/ipnew.py

```python

    str=[]
    str=response.json()
    print("****************************************")
    print("国家: %s"%(str[0]))
    print("省份: %s"%(str[1]))
    print("城市: %s"%(str[2]))
    print("区域: %s"%(str[3]))
    print("运营商: %s"%(str[4]))
    print("数据来源<www.ipip.net免费查询接口>")
    print("****************************************")
```

```python
    json_string=response.json()

    symbols = ['国家', '省份', '城市', '区域', '运营商']
    print("****************************************")
    counter = 0
    for symbol in symbols:
        print(symbol + ' ' + str.format(json_string[counter]))
        counter += 1
    print("数据来源<www.ipip.net免费查询接口>")
    print("****************************************")
```

Name: Duplicate code – the function has 8 print lines, 5 of which can be solved with 1 loop.

Refactor – Added a loop and through the symbols into an array to iterate over. Also removed the %s format as str.format is the newer convention.

c.

https://github.com/udacity/ud330/blob/master/InitialProject/project.py

```python

@app.route('/restaurant/new/', methods=['GET', 'POST'])
def newRestaurant():
    if request.method == 'POST':
        newRestaurant = Restaurant(name=request.form['name'])
        session.add(newRestaurant)
        flash('New Restaurant %s Successfully Created' % newRestaurant.name)
        session.commit()
        return redirect(url_for('showRestaurants'))
    else:
        return render_template('newRestaurant.html')

# Edit a restaurant


@app.route('/restaurant/<int:restaurant_id>/edit/', methods=['GET', 'POST'])
def editRestaurant(restaurant_id):
    editedRestaurant = session.query(
        Restaurant).filter_by(id=restaurant_id).one()
    if request.method == 'POST':
        if request.form['name']:
            editedRestaurant.name = request.form['name']
            flash('Restaurant Successfully Edited %s' % editedRestaurant.name)
            return redirect(url_for('showRestaurants'))
    else:
        return render_template('editRestaurant.html', restaurant=editedRestaurant)

```

```python
@app.route('/restaurant/new/', methods=['GET', 'POST'])
def newRestaurant():
    if request.method == 'POST':
        newRestaurant = Restaurant(name=request.form['name'])
        session.add(newRestaurant)
        flash('New Restaurant %s Successfully Created' % newRestaurant.name)
        session.commit()
        return redirect(url_for('showRestaurants'))
    else:
        return render_template('newRestaurant.html')


@app.route('/restaurant/<int:restaurant_id>/edit/', methods=['GET', 'POST'])
def editRestaurant(restaurant_id):
    editedRestaurant = session.query(
        Restaurant).filter_by(id=restaurant_id).one()
    if request.method == 'POST':
        if request.form['name']:
            editedRestaurant.name = request.form['name']
            flash('Restaurant Successfully Edited %s' % editedRestaurant.name)
            return redirect(url_for('showRestaurants'))
    else:
        return render_template('editRestaurant.html', restaurant=editedRestaurant)
```

Name: Obsolete comment – the comments before the function are named the same as the function

Refactor: Removed the comments as the functions are already named with concise names.

d. https://github.com/dangrover/sf-transit-inequality/blob/master/code/grab_routes.py

```python
# Helper to call an FCC API to grab the correct census tract for a given lat/lon.
def get_fips(latitude, longitude):
        r = requests.get("http://data.fcc.gov/api/block/find?format=json&latitude=%f&longitude=%f&showall=true" % (latitude, longitude))
        return r.json()
```

```python
1 # Helper to call an FCC API to grab the correct census tract for a given lat/lon.
2 def get_fips(latitude, longitude):
3     census_tract = requests.get("http://data.fcc.gov/api/block/find?format=json&latitude=%f&longitude=%f&showall=true" % (latitude, longitude))
4     return census_tract.json()
```

Name: Vague naming – r could easily be replaced with a concise name, but instead it is left vague. Further in the code a separate r is used for iterating through an array, furthering the confusion.

Refactor: Renamed r to ‘census_tract’, I’m not sure what exactly a census tract is, however, it seems to follow the logic set-out by the comment in line 1.

e. https://github.com/aiti-ghana-2012/Lab_Python_03/blob/master/solutions/Lab03_1.py

```python
#program to get the first 50 primes

#print the first 50 primes
n = 50

print "the first 50 primes:"

#initialize the counter that keeps
#track of how many primes we have found
prime_count = 0

#possible_prime is the number that
#we are going to check to see if it's prime
#2 is the first prime number, so we start there
possible_prime = 2

#we want to keep looking for primes as long
#as we have found less than the number for which
#we are looking (which is 50 in this case)
while prime_count < n:

    #initialize a counter that will keep track of
    #the number of divisors that possible_prime will have
    divisor_count = 0

    #we want to loop over every number from
    #1 to possible_prime, checking if it is
    #a divisor of possible_prime
    for i in range(1,possible_prime+1):

        #if i is a divisor of possible_prime...
        if possible_prime % i == 0:
            #increment the divisor count by 1
            divisor_count += 1
```

```python

"""
Lab_Python_03
Solution for Question 1
"""

#program to get the first 50 primes

def main()
    n = 50

    print "the first 50 primes:"


    prime_count = 0
    possible_prime = 2

    while prime_count < n:

        divisor_count = 0
        for i in range(1,possible_prime+1):
            if possible_prime % i == 0:
                divisor_count += 1

        if divisor_count == 2:

            #(Comma print WITHOUT a newline - prevent the newline)
            print possible_prime,
            prime_count += 1

            #if mult of 10 print new line
            if prime_count % 10 == 0:
                print

    possible_prime += 1

if __name__ == '__main__':
    main()
```

Name: Long Comment – The comments are typically 3 in-line codes long and are overly descriptive for the program. The function names are explicit enough to provide the description of the program.

Refactor: Removed most of the comment blocks and combined 3 line comments into 1. Also added a main function.

f. https://github.com/jpf/wikigifs/blob/master/parse.py

```python
def process(row):
    if len(row) < 8:
        return False
    if not row[7] == "'image'":
        return False
    if not row[8] == "'gif'":
        return False
    # print [row[0],row[7],row[8]]
    # print row
    return True
```

smell6.py > ...

```python
def process(row):
    if len(row) < 8:
        return False
    if not row[7] == "'image'":
        return False
    if not row[8] == "'gif'":
        return False
    return True
```

Name: Commented out code – print functions are left in the code, most likely from testing

Refactor: Removed the print functions