```python

##### Question 1:

### a.    GitHub Workflow: Commit the edits --------> Push the commited edits --------> Create a pull request to the LSD. [2 marks]

### b.    The snipet is re-written thus:

if co2_comp > 0.12 or n2_comp > 0.03 or h2s_comp > 0:
    gas_gravity = some_computations
    print('The corrected gas gravity is', gas_gravity)
else:
    print('Correction not needed')

### c.    The Function 'volumetrics' is thus created:

def volumetrics(area, thickness, porosity):
    BV = area*thickness
    PV = BV*porosity
    return (BV, PV)

# Sample call
bulk_vol, pore_vol = volumetrics(100, 15, 0.2)

### d.

## i.    1 is added to ny in order to generate a sequence of intergers starting from 1 and ending at ny. The function 'range' generates sequences that ends with a value 1 less than the second argument. In order to make up for this short fall, 1 is added to ny so that the sequence ends with ny.

## ii.   The bug: Python would not concatenate 'Block' (a string) with block_n_order (an integer)
#       The fix: convert block_n_order to a string using function str thus:
block_label = 'Block'+str(block_n_order)

## iii.  1 is subtracted from block_n_order so that the intended element in poro_list is correctly indexed. Python lists indices starts at zero, so that the first element is indexed 0; second element is indexed 1; and so on. Consequently, the nth element is indexed n-1.

## iv.   Methods 'extend' or 'insert' could have been used in place of Method 'append'.
stoiip_list.extend(block_stoiip)
stoiip_list.insert(i - 1, block_stoiip)

## v.    The new line is thus:
stoiip_tuple + (block_stoiip,)



##### Question 2:

### a.    GitHub Workflow: Accept pull request --------> Fetch changes --------> Merge changes. [2 marks]

### b.    The purpose of Line 76 is to initialize a list (as a place-holder) into which the STOIIP value to be computed in the 'for' loop would be stored


### c.    The Function 'darcy_rate' is thus created:

def darcy_rate(perm, area, del_p, visc, interval, cf = 0.001127):
    q = (cf*perm*area*del_p)/(visc*interval)
    return round(q, 2)


### d.  The Function 'mat_bal' is thus created:

def mat_bal (nx, ny, N, boi, pi, pb, bob, co, ce, pnow_list):
    total_np = 0
    np_list =[]
    for j in range(1,ny+1):
        for i in range(1,nx+1):
            block_n_order = (nx*(j-1))+i
            pnow = pnow_list[(block_n_order - 1)]
            bo = bob*(1 - (co*(pnow - pb)))
            block_np = (N*boi*ce*(pi - pnow))/bo
            np_list.append(block_np)
            total_np = total_np + block_np
    return np_list, total_np




##### Question 3:

### a.    The object stoiip_list should have been initialized before the given 'for' loop

stoiip_list = []

### b.    rs is NOT optional, in the context of the function's workings.
# If a user did not specify value for rs, function fvf itself computes its value internally by calling functions sol_gor
# The internally-computed value of rs then overides the default values (None) and get used in computation.


### c.    The 'while' loop is thus:

while average_pressure > bubble_pressure:
    np = (N*boi*ce*(pi - pnow))/bo
print('Bubble-point attained')



### d.  The Function 'block_classifier' is thus created:

def block_classifier (perm_list, cut_off, nx, ny):
    classification_dict = {}
    for j in range(1,ny+1):
        for i in range(1,nx+1):
            block_n_order = (nx*(j-1))+i
            block_ID = 'Block'+str(block_n_order)
            if perm_list[block_n_order - 1] < cut_off:
                classification_dict[block_ID] = 'inactive'
            else:
                classification_dict[block_ID] = 'active'
    return classification_dict
```

```
107:
108:
109:
110: ##### Question 4:
111:
112: ### a.    The logical statement is thus:
113:
114: current_pressure < bubble_pressure or flow_rate < economic_limit
115:
116: ### b.    From the function header in Line 7, pressure is positionally the second
     argument. However, this call positionally passed the intended temperature value as
     the second argument. The sript would erroneously assign this temperature value to
     pressure instead.
117: # To fix this, the value should be passed, still as the second argument, but
     keyworded. With the keyword, the value is rightly assigned to temperature.
118:
119: gas_density(0.786, temperature = 600)
120:
121:
122: ### c.    The 'for' loop is thus:
123:
124: for blockID in stoiip_dict:
125:   print(blockID + ': ' + str(stoiip_dict[blockID]))
126:
127:
128:
129: ### d.  The Function 'appar_molweight' is thus created:
130:
131: def appar_molweight(molfrac_list, molweight_list):
132:     product_sum = 0
133:     for i in range(len(molfrac_list)):
134:         product = molfrac_list[i] * molweight_list[i]
135:         product_sum = product_sum + product
136:     return round(product_sum, 4)
137:
138: # The function gas_density is thus created:
139:
140: def gas_density(P, T, R = 10.73, molfrac_list = None, molweight_list = None, Mapp =
     None):
141:     if Mapp is None:
142:         Mapp = appar_molweight(molfrac_list, molweight_list)
143:     mix_density = (P*Mapp)/(R*T)
144:     return round(mix_density, 4)
145:
```