```
# -*- coding: utf-8 -*-
"""
Created on Tue May  7 13:15:38 2019

@author: ebdegu01
"""

import xlsxwriter
from pathlib import Path

income_statement = xlsxwriter.Workbook("C:\Users\ebdegu01\Documents\Programming Notes\Python
Scripts\Accounting_Workbook.xlsx")
inc_worksheet = income_statement.add_worksheet("Income_Statement")

accounting_wb_file = Path("C:\Users\ebdegu01\Documents\Programming Notes\Python
Scripts\Accounting_Workbook.xlsx")

if accounting_wb_file.is_file():
    print("yes this file exists")

bold = income_statement.add_format({'bold': True})
bold.set_font_size(15)
currency = income_statement.add_format({'num_format': '$#,##0.00'})

def set_inctitle_format():
    inc_worksheet.set_column('A:D', 25)
    inc_worksheet.set_row(0,25)
set_inctitle_format()
```

```
merge_format = income_statement.add_format({
        'bold': 1,
        'font': 20,
        'border': 1,
        'align': 'center',
        'valign': 'vcenter',
        'fg_color': 'yellow'})

inc_worksheet.merge_range('A1:D1', 'Income Statement', merge_format,)


def inc_gross():
    inc_worksheet.write('A2', 'Sales', bold)
    inc_worksheet.write('A3', 'Cost of Goods Sold', bold)
    inc_worksheet.write('A4', 'Gross Profit', bold)
inc_gross()


def gross_prof():
    inc_worksheet.write('D4', '=C2-C3', currency)
gross_prof()


def op_exp():
    inc_worksheet.write('A6', 'Operating Expenses', bold)
    inc_worksheet.write('A7', 'Selling Expenses', bold)
    inc_worksheet.write('A8', 'Advertising Expenses', bold)
    inc_worksheet.write('A9', 'Commissions Expense', bold)
    inc_worksheet.write('A10', 'Administrative Expenses', bold)
    inc_worksheet.write('A11', 'Office Supples Expense', bold)
    inc_worksheet.write('A12', 'Office Equipment Expense', bold)
    inc_worksheet.write('A13', 'Total Operating Expenses', bold)
```

```
op_exp()

def op_exp_total():
    inc_worksheet.write('D13', '=SUM(C6:C13)',currency)
op_exp_total()

def non_op_exp():
    inc_worksheet.write('A15', 'Non-Operating or Other', bold)
    inc_worksheet.write('A16', 'Interest Revenues', bold)
    inc_worksheet.write('A17', 'Rent Revenue', bold)
    inc_worksheet.write('A18', 'Dividend Revenue', bold)
    inc_worksheet.write('A19', 'Gain on Sale of Investments', bold)
    inc_worksheet.write('A20', 'Interest Expense', bold)
    inc_worksheet.write('A21', 'Loss', bold)
    inc_worksheet.write('A22', 'Total Non-Operating Expense', bold)
non_op_exp()

def nonop_exp_total():
    inc_worksheet.write('D22', '=SUM(C15:C21)', currency)
nonop_exp_total()

def net_income():
    inc_worksheet.write('A24', 'Net Income', bold)
    inc_worksheet.write('D24', '=D4-D13-D22', currency)
net_income()


sales_revenue = []
cogs = []
```

```
gross_profit = []

sell_exp = []

adv_exp = []

comm_exp = []

admin_exp = []

supplies_exp = []

equip_exp = []

int_rev = []

rent_rev = []

gain = []

int_exp = []

loss = []

util_exp = []

div_rev = []


def eq_revenue():

    user_amt = raw_input('Revenue Amount: ')

    print(user_amt)

    revenue_amt = float(user_amt)

    if revenue_amt > 0:

        revenue_transaction = raw_input('''


                                'Sales',

                                'Interest',

                                'Rent',

                                'Dividend'


                            ''')

        if revenue_transaction == 'Sales' or revenue_transaction == 'sales':
            sales_revenue.append(revenue_amt)
            for a in sales_revenue:
                print(a)
            sales_total = sum(sales_revenue)
            inc_worksheet.write('C2', sales_total, currency)
        elif revenue_transaction == 'Interest' or revenue_transaction == 'interest':
            int_rev.append(revenue_amt)
            for b in int_rev:
                print(b)
            int_total = sum(int_rev)
            inc_worksheet.write('C16', int_total, currency)
        elif revenue_transaction == 'Rent' or revenue_transaction == 'rent':
            rent_rev.append(revenue_amt)
            for c in rent_rev:
                print(c)
            rent_total = sum(rent_rev)
            inc_worksheet.write('C17', rent_total, currency)
        elif revenue_transaction == 'Dividend' or revenue_transaction == 'dividend':
            div_rev.append(revenue_amt)
            for d in div_rev:
                print(d)
            div_total = sum(div_rev)
            inc_worksheet.write('C18', div_total, currency)
    income_statement_menu()


def eq_exp():
    user_amt = raw_input('Expense Amoount: ')
    print(user_amt)
    exp_amt = float(user_amt)
    if exp_amt > 0:
        exp_transaction = raw_input('''


                                'Selling',
                                'Advertising',
                                'Commissions',
                                'Administrative',
                                'Supplies',
                                'Equipment',
                                'Interest',
                                'Cost of Goods Sold'


                                ''')
        if exp_transaction == 'Cost of Goods Sold' or exp_transaction == 'cost of goods sold' or
exp_transaction == 'cogs' or exp_transaction == 'COGS':
            cogs.append(exp_amt)
            for l in cogs:
                print(l)
            cogs_total = sum(cogs)
            inc_worksheet.write('C3', cogs_total, currency)
        if exp_transaction == 'Selling' or exp_transaction == 'selling':
            sell_exp.append(exp_amt)
            for e in sell_exp:
                print(e)
            sell_total = sum(sell_exp)
            inc_worksheet.write('C7', sell_total, currency)
        elif exp_transaction == 'Advertising' or exp_transaction == 'advertising':
            adv_exp.append(exp_amt)
            for f in adv_exp:
```

```
        print(f)
    adv_total = sum(adv_exp)
    inc_worksheet.write('C8', adv_total, currency)
elif exp_transaction == 'Commissions' or exp_transaction == 'commissions':
    comm_exp.append(exp_amt)
    for g in comm_exp:
        print(g)
    comm_total = sum(comm_exp)
    inc_worksheet.write('C9', comm_total, currency)
elif exp_transaction == 'Administrative' or exp_transaction == 'administrative':
    admin_exp.append(exp_amt)
    for h in admin_exp:
        print(h)
    admin_total = sum(admin_exp)
    inc_worksheet.write('C10', admin_total, currency)
elif exp_transaction == 'Supplies' or exp_transaction == 'supplies':
    supplies_exp.append(exp_amt)
    for i in supplies_exp:
        print(i)
    supplies_total = sum(supplies_exp)
    inc_worksheet.write('C11', supplies_total, currency)
elif exp_transaction == 'Equipment' or exp_transaction == 'equipment':
    equip_exp.append(exp_amt)
    for j in equip_exp:
        print(j)
    equip_total = sum(equip_exp)
    inc_worksheet.write('C12', equip_total, currency)
elif exp_transaction == 'Interest' or exp_transaction == 'interest':
    int_exp.append(exp_amt)
```

```
            for k in int_exp:
                print(k)
            int_total =sum(int_exp)
            inc_worksheet.write('C20', int_total, currency)
    income_statement_menu()

re_worksheet = income_statement.add_worksheet('Retained_Earnings')
re_worksheet.merge_range('A1:D1', 'Retained Earnings', merge_format,)



def set_retitle_format():
    re_worksheet.set_column('A:D', 25)
    re_worksheet.set_row(0,25)
set_retitle_format()

def re_col1():
    re_worksheet.write('A2', 'Retained Earnings at the beginning of the period: ', bold)
    re_worksheet.write('A3', 'Add: Net Income (or less: net loss)', bold)
    re_worksheet.write('A4', 'Less: Dividends', bold)
    re_worksheet.write('A6', 'Retained Earnings at the end of the period: ', bold)
re_col1()

def re_col2():
    beg_re = 0
    re_worksheet.write('C2', beg_re, currency)
    re_worksheet.write('C3', '=Income_Statement!D24', currency)
    re_worksheet.write('C4', 0, currency)
    re_worksheet.write('C6', '=C2+C3-C4', currency)
re_col2()
```

```
bs_worksheet = income_statement.add_worksheet('Balance_Sheet')

bs_worksheet.merge_range('A1:D1', 'Balance Sheet', merge_format)

def set_bstitle_format():

  bs_worksheet.set_column('A:D',25)

  bs_worksheet.set_row(0,25)

set_bstitle_format()


cash = []

def asset_cash():

  cash_amt = raw_input('Cash: ')

  if cash_amt > 0:

    cash.append(cash_amt)

acc_rev = []

def asset_ar():

  ar_amt = raw_input('Accounts Receivable: ')

  if ar_amt > 0:

    acc_rev.append(ar_amt)

inv = []

def asset_inv():

  inv_amt = raw_input('Inventory: ')

  if inv_amt > 0:

    inv.append(inv_amt)

supplies = []

def asset_supplies():

  supplies_amt = raw_input('Supplies: ')

  if supplies_amt > 0 :

    supplies.append(supplies_amt)

pre_exp = []
```

```
def asset_prepaid():
    prepaid_amt = raw_input('Prepaid Expenses: ')
    if prepaid_amt > 0:
        pre_exp.append(prepaid_amt)
allowance = []
def asset_allow():
    allowance_amt = raw_input('Allowance for Doubtful Accounts: ')
    if allowance_amt > 0:
        allowance.append(allowance_amt)
land = []
def asset_land():
    land_amt = raw_input('Land: ')
    if land_amt > 0:
        land.append(land_amt)
equip = []
def asset_equip():
    equip_amt = raw_input('Equipment: ')
    if equip_amt > 0:
        equip.append(equip_amt)
accrued_rev = []
def asset_accrued():
    accrued_amt = raw_input('Accrued Revenues: ')
    if accrued_amt > 0:
        accrued_rev.append(accrued_amt)
st_inv = []
def asset_stinv():
    sti_amt = raw_input('Shprt-term Investment; ')
    if sti_amt > 0:
        st_inv.append(sti_amt)
```

```
asset = [
        'Cash',
        'Accounts Receivable',
        'Inventory',
        'Supplies',
        'Prepaid Expenses',
        'Allowance for Doubtful Accounts',
        'Land',
        'Equipment',
        'Accrued Revenues',
        'Short-Term Investments'
        ]

st_loans = []
def li_stl():
    stl_amt = raw_input('Short-term Loans Payable: ')
    if stl_amt > 0:
        st_loans.append(stl_amt)
lt_debt = []
def li_ltdebt():
    ltdebt_amt = raw_input('Long-term Debt: ')
    if ltdebt_amt > 0:
        lt_debt.append(ltdebt_amt)
acc_payable = []
def li_accpay():
    accpay_amt = raw_input('Accounts Payable: ')
    if accpay_amt > 0:
        acc_payable.appned(accpay_amt)
```

```
def_rev = []
def li_defrev():
   defrev_amt = raw_input('Deferred Revenue: ')
   if defrev_amt > 0:
      def_rev.append(defrev_amt)
unearned_rev = []
def unearnedrev():
   unearnedrev_amt = raw_input('Unearned Revenue: ')
   if unearnedrev_amt > 0:
      unearned_rev.append(unearnedrev_amt)
inst_loans_pay = []
def li_installments():
   install_amt = raw_input('Installment Loans Payable; ')
   if install_amt > 0:
      inst_loans_pay.append(install_amt)
mortgage_loans = []
def li_mortgage():
   mortgage_amt = raw_input('Mortgage Loans Payable: ')
   if mortgage_amt > 0:
      mortgage_loans.append(mortgage_amt)

liability = [
          'Short-term Loans Payable',
          'Long-Term Debt',
          'Accounts Payable',
          'Accrued Expenses',
          'Deferred Revenues',
          'Unearned Revenue',
          'Installment Loans Payable',
            'Mortgage Loans Payable'
            ]
pref_stock = []
def eq_pref():
    prefstock_amt = raw_input('Preferred Stock: ')
    if prefstock_amt > 0:
        pref_stock.append(prefstock_amt)
common_stock = []
def eq_cs():
    cs_amt = raw_input('Common Stock: ')
    if cs_amt > 0:
        common_stock.append(cs_amt)
pic_pref = []
def eq_picpref():
    picpref_amt = raw_input('PIC in excess of par value - preferred stock:')
    if picpref_amt > 0:
        pic_pref.append(picpref_amt)
pic_common = []
def eq_piccomm():
    piccomm_amt = raw_input('PIC in excess of par value - common stock:')
    if piccomm_amt > 0:
        pic_common.append(piccomm_amt)
pic_treasury = []
def eq_treasury():
    treasuryst_amt = raw_input('PIC in excess of par value - treasury stock; ')
    if treasuryst_amt > 0:
        pic_treasury.append(treasuryst_amt)


equity = [
          'Preferred Stock',
          'common stock',
          'PIC in excess of par value - preferred stock',
          'PIC in excess of par vaule - common stock',
          'PIC from treasury stock',
          'retained earnings',
          'revenues',
          'expenses',
          'accumulated other comprehensive income',
          'treasury stock'
          ]

def bs_assetlist():
    bs_worksheet.write('A2', 'Asset: ', bold)
    column = 2
    row = 1
    for l in asset:
        bs_worksheet.write(column,row,l,bold)
        column += 1
    bs_worksheet.write('D2','=SUM(D3:D12)', currency)
bs_assetlist()

def bs_lilist():
    bs_worksheet.write('A14', 'Liabilities: ', bold)
    column = 14
    row = 1
    for j in liability:
        bs_worksheet.write(column,row,j,bold)
        column += 1
    bs_worksheet.write('D14', '=SUM(D15:D22)', currency)
bs_lilist()


def bs_eqlist():
    bs_worksheet.write('A24', 'Equities: ',bold)
    column = 24
    row = 1
    for k in equity:
        bs_worksheet.write(column,row,k,bold)
        column += 1
    bs_worksheet.write('D24', '=SUM(D25:D34)', currency)
bs_eqlist()


def income_statement_menu():
    usr_choice = raw_input('Choose an option: (1 - Revenue), (2 - Expense), (3 - Close)')
    if usr_choice == '1':
        eq_revenue()
    elif usr_choice == '2':
        eq_exp()
    elif usr_choice == '3':
        quit()

income_statement_menu()
```