```python
#!/usr/bin/python3

import pickle

# Chargement des données.
#
# Il faudra vraisemblablement adapter le chemin vers le fichier wooldridge.pkl.

fid = open('C:/Users/claire.loupias/Desktop/wooldridge23.pkl', 'rb')
alldatasets = pickle.load(fid)
fid.close()

# base de données de l'exercice 3 du TD4
dataset = alldatasets['mlb1']
#
# Question 1.
#
import numpy as np
dataset['logged_salary'] = np.log(dataset['salary'])

# Estimation du modèle avec rbisyr
import statsmodels.formula.api as smf
results = smf.ols('logged_salary ~ years + gamesyr + bavg + hrunsyr + rbisyr', data=dataset).fit
print(results.summary())

# Estimation du modèle sans rbisyr
results = smf.ols('logged_salary ~ years + gamesyr + bavg + hrunsyr', data=dataset).fit()
print(results.summary())

# On cherche une explication à la l'augmentation du coefficient de hrunsyr
# et au fait qu'il apparaisse soudainement significatif.
from scipy.stats import pearsonr

corr,_ = pearsonr(dataset.hrunsyr, dataset.rbisyr)
print("La corrélation entre hrunsyr et rbisyr est : ", corr)

# Vu le résultat (corr=0,89), ce n'était peut-être pas une très bonne idée d'enlever rbisyr
# La p-valeur de rbisyr n'était pas si basse (0,134)
# Maintenant, on a rendu hrunsyr significatif, alors qu'il ne l'était pas auparavant,
# en multipliant sont coef par plus que deux, ce qui laisse penser à un biais de variable omise.
# C'est peut-être hrunsyr qu'il aurait fallu enlever plutôt que rbisyr, sa p-valeur était beauco
# plus élevée.

#
# Question 2.
#

# Estimation du modèle complet (non contraint _nc)
results_nc = smf.ols('logged_salary ~ years + gamesyr + bavg + hrunsyr + runsyr + fldperc + sbas
print(results_nc.summary())

#
# Question 3.
#
# Estimation du modèle sans bavg, fldperc et sbaseyr (modèle contraint _c)
results_c = smf.ols('logged_salary ~ years + gamesyr + hrunsyr + runsyr', data=dataset).fit()
print(results_c.summary())

#On importe tout le module de stats de scipy (et pas une seule fonction comme dans un exercice p
from scipy import stats
```

```
# Calcul de la statistique de Fisher
# ssr = sum of squared residuals, autrement dit, la somme des carrés des résidus.
# df_resid = degree of freedom, autrement dit, les degrés de liberté des résidus
# df_resid= n-k-1, c'est-à-dire le nombre d'observations moins le nombre de variables explicati
# cdf = cumulative distribution function, autrement dit, la fonction de répartition
# f.cdf = fonction de répartition pour la loi de Ficher (f)

F=(results_c.ssr-results_nc.ssr)/3/(results_nc.ssr/results_nc.df_resid)
# rajouter le calcul de la valeur critique ici
pval = 1-stats.f.cdf(F, 3, results_nc.df_resid)

print('La statistique de Fisher est ', F)
print("La valeur d'un Fisher à 3 et",results_nc.df_resid,'degrés de libertés est ', '2,60')
print('La p-valeur associée au test de Fisher ci-dessus est ', pval)

# Calcul préprogrammé dans Python du test de Fisher (f)
# Il donne directement la statistique de Fisher, la p-valeur critique et les degrés de liberté
print(results_nc.f_test("bavg=fldperc=sbasesyr=0"))
```