```
import numpy as np
from sklearn_extra.cluster import KMedoids
import networkx as nx
import pandas as pd
import matplotlib.pyplot as plt
import time
import xlsxwriter
from collections import defaultdict

sheets_dict = pd.read_excel(r"C:\\Users\\Asus\\Downloads\\
studentcourses.xlsx", sheet_name=None)

# Access a specific sheet by name
sheetScience= sheets_dict['علوم طبية ']
sheetEng= sheets_dict['الهندسة']
sheetAdmin= sheets_dict['العلوم الادارية']
sheetPharma= sheets_dict['صيدلة وطب اسنان']
sheetComp= sheets_dict['حاسب الي']

columns = [
    "م",
    "رقم الطالب",
    "اسم الطالب",
    "التخصص",
    "رقم المقرر",
    "اسم المقرر",
    "رمز المقرر" ,
     "رقم الشعبة"
]
sheetScience.columns= columns
sheetEng.columns= columns
sheetAdmin.columns= columns
sheetPharma.columns= columns
sheetComp.columns= columns

merged_df = pd.concat(sheets_dict.values(), ignore_index=True)
merged_df = merged_df.drop(columns=['م'])

def get_most_common_name(names):
    return names.mode()[0]

merged_df['اسم المقرر'] = merged_df.groupby('رمز المقرر')[' اسم
المقرر'].transform(get_most_common_name)
merged_df
```

	التخصص	اسم الطالب	رقم الطالب \
0	1105312002	السجلات الطبية-2	عبدالمجيد علي عبدالرحمن الشاوي
1	1105311013	السجلات الطبية-2	سالم سليمان دهش الشمري

```
2      1105312004          احمد عبدالله صالح الحربي  السجلات الطبية-2
3      1105321021       يوسف عبدالله سليمان المهوس  السجلات الطبية-2
4      1105312011           سطام بندر جايز المطيري  السجلات الطبية-2
...           ...                                ...                 ...
14786  1501311010     أحمد عبدالعزيز محمد العبيدان      هندسة الحاسب
14787  1401321075            صالح محمد صالح الرميح       علوم الحاسب
14788  1502301005          يزيد محمد صالح الهجرس       علوم حاسب-1
14789  1502312010       محمد عبدالله محيميد العريني       علوم الحاسب
14790  1502312003            صالح أحمد صالح المعتاز       علوم الحاسب

       رقم الشعبة رمز المقرر      اسم المقرر رقم المقرر
0           110501  ARAB 101  المهارات اللغوية     211637
1           110501  ARAB 101  المهارات اللغوية     211637
2           110511  ARAB 102    التحرير العربي      111647
3           110511  ARAB 102    التحرير العربي     211647
4           110511  ARAB 102    التحرير العربي     211647
...            ...       ...               ...        ...
14786       150147    215153    342عال  هندسة البرمجيات
14787       150147    115152    342عال  هندسة البرمجيات
14788       150147    115155    342عال  هندسة البرمجيات
14789       150147    115152    342عال  هندسة البرمجيات
14790       150147    115152    342عال  هندسة البرمجيات

[14791 rows x 7 columns]
```

```
merged_df = merged_df.drop_duplicates(subset=["رقم المقرر","رقم الطالب"])

merged_df.dropna(subset=["رقم المقرر","رقم الطالب"], inplace=True)
print(merged_df.isnull().sum())
merged_df
```

```
0  رقم الطالب
0  اسم الطالب
0     التخصص
0  رقم المقرر
0  اسم المقرر
0  رمز المقرر
0  رقم الشعبة
dtype: int64

C:\Users\Asus\AppData\Local\Temp\ipykernel_21536\1811259916.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  merged_df.dropna(subset=["رقم المقرر","رقم الطالب"], inplace=True)
```

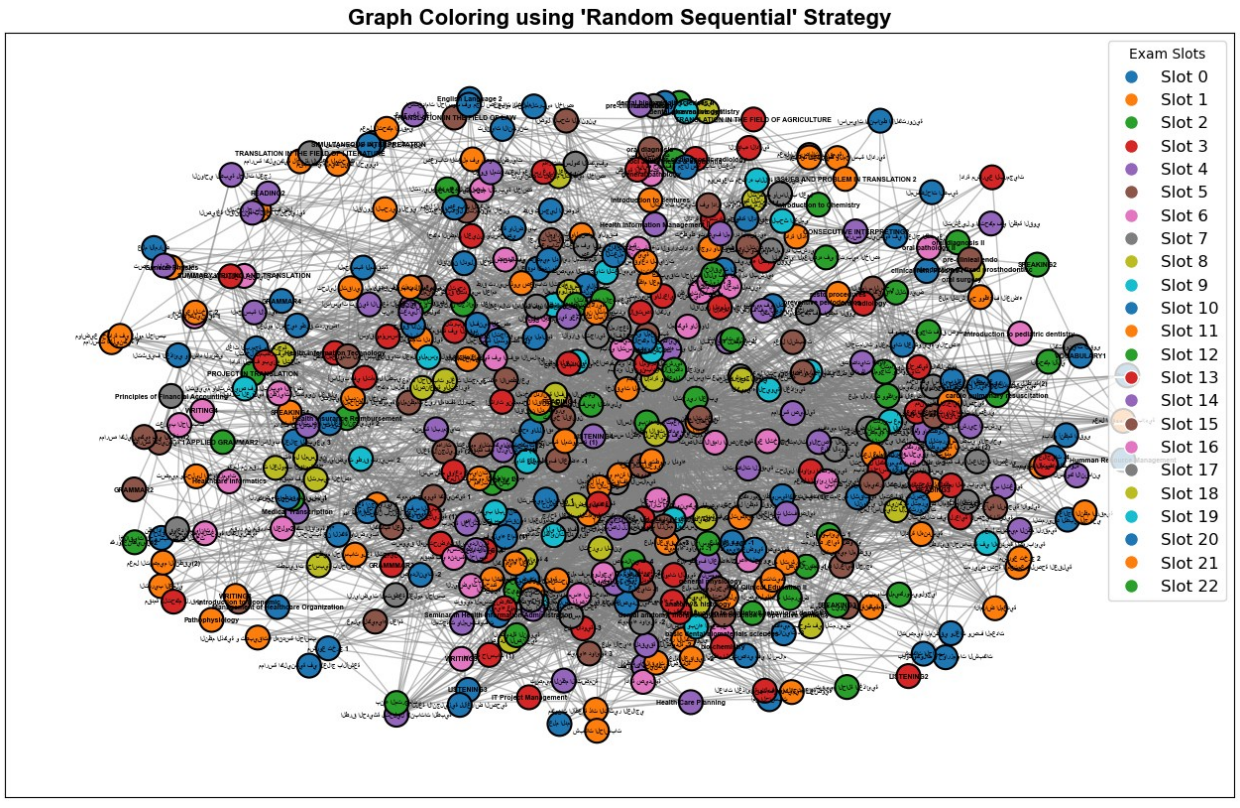
```
        التخصص       اسم الطالب                  رقم الطالب  \
0      1105312002  عبدالمجيد علي عبدالرحمن الشاوي  السجلات الطبية-2
1      1105311013          سالم سليمان دهش الشمري  السجلات الطبية-2
2      1105312004          احمد عبدالله صالح الحربي  السجلات الطبية-2
3      1105321021       يوسف عبدالله سليمان المهوس  السجلات الطبية-2
4      1105312011          سطام بندر جايز المطيري  السجلات الطبية-2
...           ...                             ...                 ...
14786  1501311010    أحمد عبدالعزيز محمد العبيدان      هندسة الحاسب
14787  1401321075          صالح محمد صالح الرميح       علوم الحاسب
14788  1502301005       يزيد محمد صالح الهجرس       علوم حاسب-1
14789  1502312010      محمد عبدالله محيميد العريني       علوم الحاسب
14790  1502312003          صالح أحمد صالح المعتاز       علوم الحاسب

      رقم الشعبة رمز المقرر      اسم المقرر رقم المقرر
0          110501  ARAB 101  المهارات اللغوية     211637
1          110501  ARAB 101  المهارات اللغوية     211637
2          110511  ARAB 102    التحرير العربي      111647
3          110511  ARAB 102    التحرير العربي     211647
4          110511  ARAB 102    التحرير العربي     211647
...           ...       ...               ...        ...
14786      150147    215153    342عال  هندسة البرمجيات
14787      150147    115152    342عال  هندسة البرمجيات
14788      150147    115155    342عال  هندسة البرمجيات
14789      150147    115152    342عال  هندسة البرمجيات
14790      150147    115152    342عال  هندسة البرمجيات

[11104 rows x 7 columns]
```

```python
from itertools import combinations
# Create a graph where nodes are courses and edges connect conflicting courses (shared students)
G = nx.Graph()

# Group students by courses
student_courses = defaultdict(set)  #automatically initializes missing keys with an empty set.
for _, row in merged_df.iterrows():
    student_courses[row["اسم المقرر"]].add(row["رقم الطالب"])  #Adds each student'sID to the corresponding course name

# Add edges between courses taken by the same student (without self-loops)
for c1, c2 in combinations(student_courses.keys(), 2):  # Only unique pairs
    inters = student_courses[c1].intersection(student_courses[c2])
#set containing the ids which are shared bet c1 and c2
    if inters:  # If there's a conflict
```

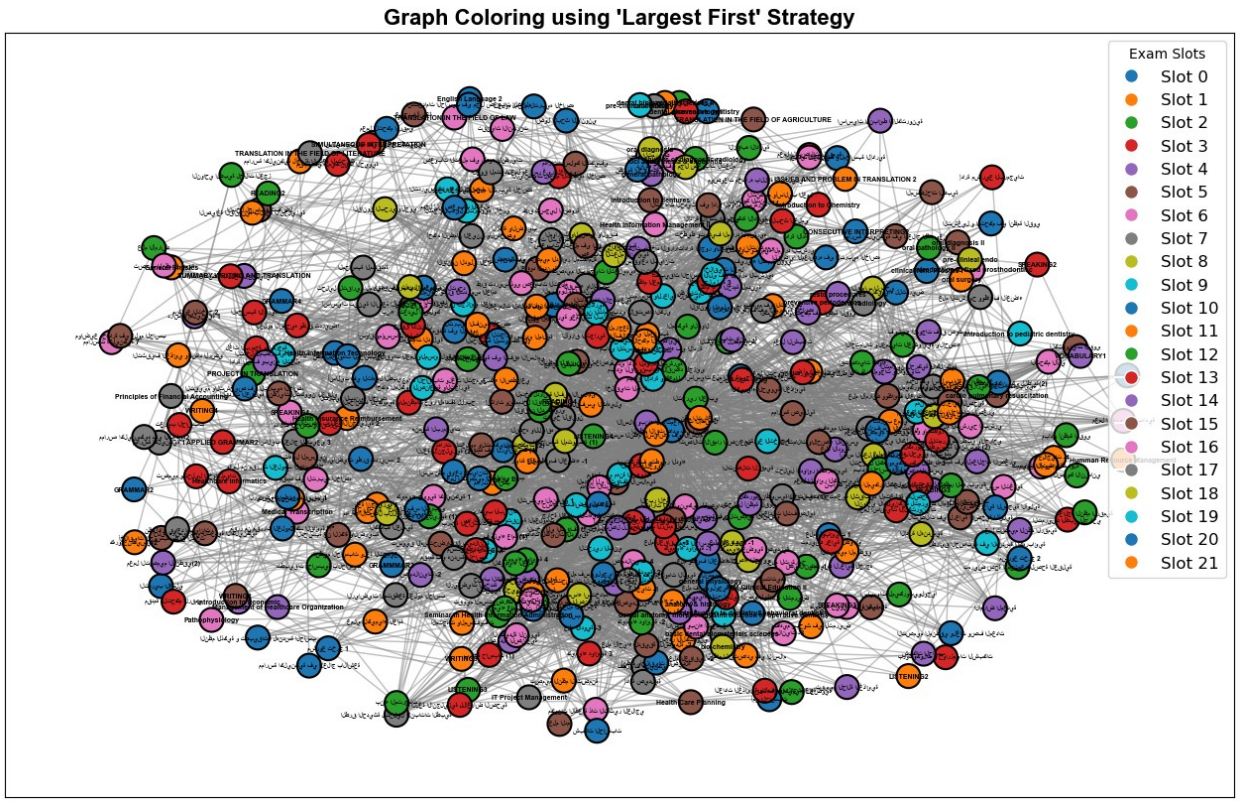
```
        G.add_edge(c1, c2, weight=len(inters))  # Weighted by shared
students

import matplotlib.pyplot as plt
import networkx as nx

# Define coloring strategies
strategies = {
    "Random Sequential": "random_sequential",
    "Largest First": "largest_first"
}

# Get a color palette
unique_colors = list(plt.cm.get_cmap("tab10").colors)

# Loop through each strategy
for title, strategy in strategies.items():
    coloring = nx.coloring.greedy_color(G, strategy=strategy)

    # Generate node colors based on coloring
    node_colors = [unique_colors[coloring[node] % len(unique_colors)]
for node in G.nodes()]

    # Arabic labels for courses
    labels = {node: attrs.get("اسم المقرر", node) for node, attrs in
G.nodes(data=True)}

    # Set up plot
    plt.figure(figsize=(16, 10))
    pos = nx.spring_layout(G, seed=42, k=0.9)

    # Draw nodes
    nx.draw_networkx_nodes(G, pos, node_color=node_colors,
node_size=300, edgecolors="black", linewidths=1.5)

    # Draw edges
    nx.draw_networkx_edges(G, pos, edge_color="gray", width=1,
alpha=0.5)

    # Draw labels
    nx.draw_networkx_labels(G, pos, labels, font_size=5,
font_weight="bold", font_family="Arial")

    # Create legend
    color_map = {slot: unique_colors[slot % len(unique_colors)] for
slot in set(coloring.values())}
    legend_labels = [
        plt.Line2D([0], [0], marker='o', color='w',
markerfacecolor=color, markersize=10, label=f"Slot {slot}")
        for slot, color in color_map.items()
```

```
    ]
    plt.legend(handles=legend_labels, title="Exam Slots", loc="upper
right", fontsize=12)

    # Set title and show
    plt.title(f"Graph Coloring using '{title}' Strategy",
fontname="Arial", fontsize=16, fontweight="bold")
    plt.show()

    # Print number of slots used
    print(f"{title} strategy used {len(set(coloring.values()))}
slots.")

# Optionally: print number of courses
print(f"Total number of courses (nodes): {G.number_of_nodes()}")
G.number_of_edges()
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_21536\462961462.py:11:
MatplotlibDeprecationWarning: The get_cmap function was deprecated in
Matplotlib 3.7 and will be removed in 3.11. Use
``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap()``
or ``pyplot.get_cmap()`` instead.
  unique_colors = list(plt.cm.get_cmap("tab10").colors)
```




```
Random Sequential strategy used 23 slots.
```




```
Largest First strategy used 22 slots.
Total number of courses (nodes): 455

5206
```

Fewer slots = fewer exam days = more efficient schedule.
so largest first strategy is better and more efficient for making the exam timetable as each slot represents an exam day

```
"""
import pandas as pd
from datetime import datetime, timedelta

# Assume this coloring is from the best strategy
coloring = nx.coloring.greedy_color(G, strategy="largest_first")

# Reverse mapping: slot -> list of courses
slot_to_courses = {}
for course, slot in coloring.items():
    slot_to_courses.setdefault(slot, []).append(course)

# Define schedule parameters
start_date = datetime(2024, 5, 19)  # Example start date (Sunday)
periods_per_day = 2
period_times = ["9:00 AM", "1:00 PM"]
```

```
# Generate a mapping of slot -> (date, time)
slot_schedule = {}
for i, slot in enumerate(sorted(slot_to_courses.keys())):
    day_offset = i // periods_per_day
    period = i % periods_per_day

    exam_date = start_date + timedelta(days=day_offset)
    exam_time = period_times[period]

    slot_schedule[slot] = {
        "date": exam_date.strftime('%Y-%m-%d'),
        "time": exam_time
    }

# Build timetable DataFrame
rows = []
for slot, courses in slot_to_courses.items():
    for course in courses:
        rows.append({
            "Course Code": G.nodes[course].get("رمز المقرر", ""),
            "Course Name": G.nodes[course].get("اسم المقرر", course),
            "Exam Date": slot_schedule[slot]["date"],
            "Exam Time": slot_schedule[slot]["time"]
        })

timetable_df = pd.DataFrame(rows).sort_values(by=["Exam Date", "Exam
Time"])

# Show the final timetable
print(timetable_df.head(20))

# Optional: Save to Excel
output_path = "D:/exam_timetable.xlsx"
timetable_df.to_excel(output_path, index=False)
"""

'\nimport pandas as pd\nfrom datetime import datetime, timedelta\n\n#
Assume this coloring is from the best strategy\ncoloring =
nx.coloring.greedy_color(G, strategy="largest_first")\n\n# Reverse
mapping: slot -> list of courses\nslot_to_courses = {}\nfor course,
slot in coloring.items():\n    slot_to_courses.setdefault(slot,
[]).append(course)\n\n# Define schedule parameters\nstart_date =
datetime(2024, 5, 19)  # Example start date (Sunday)\nperiods_per_day
= 2\nperiod_times = ["9:00 AM", "1:00 PM"]\n\n# Generate a mapping of
slot -> (date, time)\nslot_schedule = {}\nfor i, slot in
enumerate(sorted(slot_to_courses.keys())):\n    day_offset = i //
periods_per_day\n    period = i % periods_per_day\n\n    exam_date =
start_date + timedelta(days=day_offset)\n    exam_time =
period_times[period]\n\n    slot_schedule[slot] = {\n        "date":
```

```
exam_date.strftime(\'%Y-%m-%d\'),\n        "time": exam_time\n    }\n\
n# Build timetable DataFrame\nrows = []\nfor slot, courses in
slot_to_courses.items():\n    for course in courses:\n
rows.append({\n            "Course Code": G.nodes[course].get(" رمز
المقرر", ""),\n            "Course Name": G.nodes[course].get(" اسم
المقرر", course),\n            "Exam Date": slot_schedule[slot]
["date"],\n            "Exam Time": slot_schedule[slot]["time"]\n
})\n\ntimetable_df = pd.DataFrame(rows).sort_values(by=["Exam Date",
"Exam Time"])\n\n# Show the final timetable\
nprint(timetable_df.head(20))\n\n# Optional: Save to Excel\
noutput_path = "D:/exam_timetable.xlsx"\
ntimetable_df.to_excel(output_path, index=False)\n'

import matplotlib.colors as mcolors

# Step 1: Map each course to its assigned exam slot
merged_df["exam_slot"] = merged_df["اسم المقرر"].map(coloring)

# Step 2: Count unique students per exam slot
students_per_day = (
    merged_df.groupby("exam_slot")["رقم الطالب"]
    .nunique()
    .reset_index()
    .rename(columns={"رقم الطالب": "unique_students"})
)

# Step 3: Identify the busiest day (slot with most students)
max_students_day =
students_per_day.loc[students_per_day["unique_students"].idxmax()]
busiest_slot = int(max_students_day["exam_slot"])
max_students = int(max_students_day["unique_students"])

print("Maximum number of students scheduled on a single exam day:")
print(f"Slot (busiest day): {busiest_slot}")
print(f"Students: {max_students}")
merged_df
```

```
Maximum number of students scheduled on a single exam day:
Slot (busiest day): 6
Students: 978
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_21536\2034545319.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  merged_df["exam_slot"] = merged_df["اسم المقرر"].map(coloring)
```

```
        التخصص       اسم الطالب                  رقم الطالب  \
0      1105312002  السجلات الطبية-2  عبدالمجيد علي عبدالرحمن الشاوي
1      1105311013  السجلات الطبية-2  سالم سليمان دهش الشمري
2      1105312004  السجلات الطبية-2  احمد عبدالله صالح الحربي
3      1105321021  السجلات الطبية-2  يوسف عبدالله سليمان المهوس
4      1105312011  السجلات الطبية-2  سطام بندر جايز المطيري
...           ...                ...                     ...
14786  1501311010     هندسة الحاسب  أحمد عبدالعزيز محمد العبيدان
14787  1401321075       علوم الحاسب  صالح محمد صالح الرميح
14788  1502301005       علوم حاسب-1  يزيد محمد صالح الهجرس
14789  1502312010       علوم الحاسب  محمد عبدالله محيميد العريني
14790  1502312003       علوم الحاسب  صالح أحمد صالح المعتاز

       رقم الشعبة رمز المقرر      اسم المقرر  رقم المقرر  exam_slot
0           110501  ARAB 101  المهارات اللغوية      211637        3.0
1           110501  ARAB 101  المهارات اللغوية      211637        3.0
2           110511  ARAB 102    التحرير العربي      111647        2.0
3           110511  ARAB 102    التحرير العربي      211647        2.0
4           110511  ARAB 102    التحرير العربي      211647        2.0
...            ...       ...               ...         ...        ...
14786       150147       4.0            215153    عال342  هندسة البرمجيات
14787       150147       4.0            115152    عال342  هندسة البرمجيات
14788       150147       4.0            115155    عال342  هندسة البرمجيات
14789       150147       4.0            115152    عال342  هندسة البرمجيات
14790       150147       4.0            115152    عال342  هندسة البرمجيات

[11104 rows x 8 columns]
```

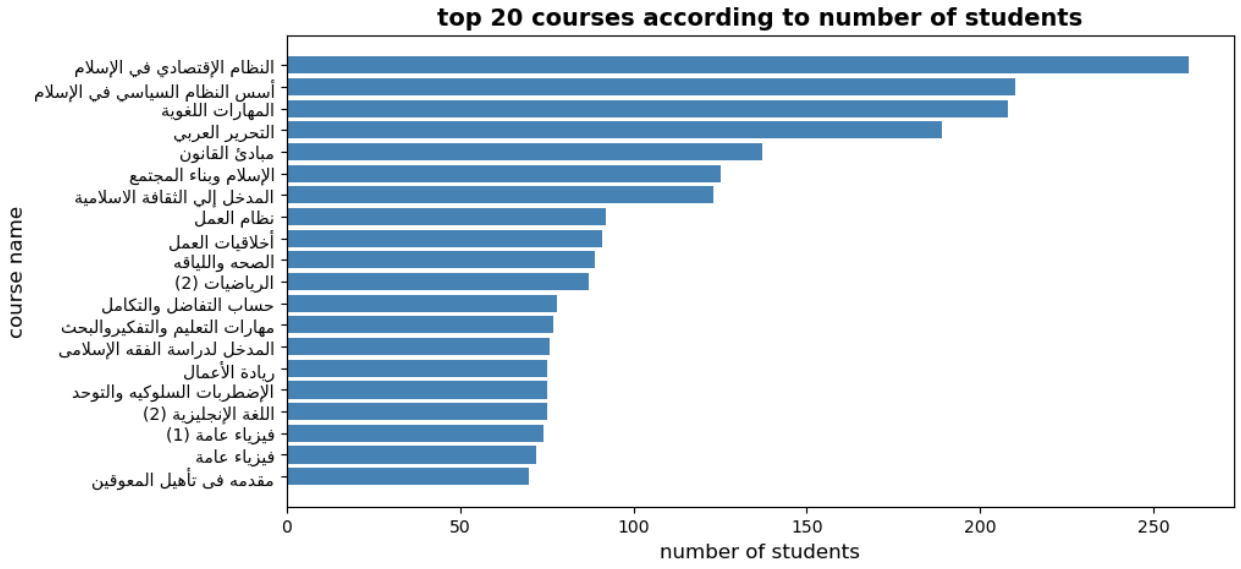
```
students_in_courses = merged_df.groupby(["اسم المقرر"])[" رقم
الطالب"].nunique()
# Print the results
print("Number of Students in Each Course:")
print(students_in_courses)

# Convert to DataFrame for easy export
students_in_courses_df = students_in_courses.reset_index()
students_in_courses_df.columns = ['اسم المقرر','عدد الطلاب']

# Save the DataFrame to an Excel file
output_path = "D:/students_in_courses.xlsx"
students_in_courses_df.to_excel(output_path, index=False)
```

```
Number of Students in Each Course:
اسم المقرر
APPLIED GRAMMAR2              6
Biology                       1
CONSECUTIVE INTERPRETING2     2
English Language 2            3
```

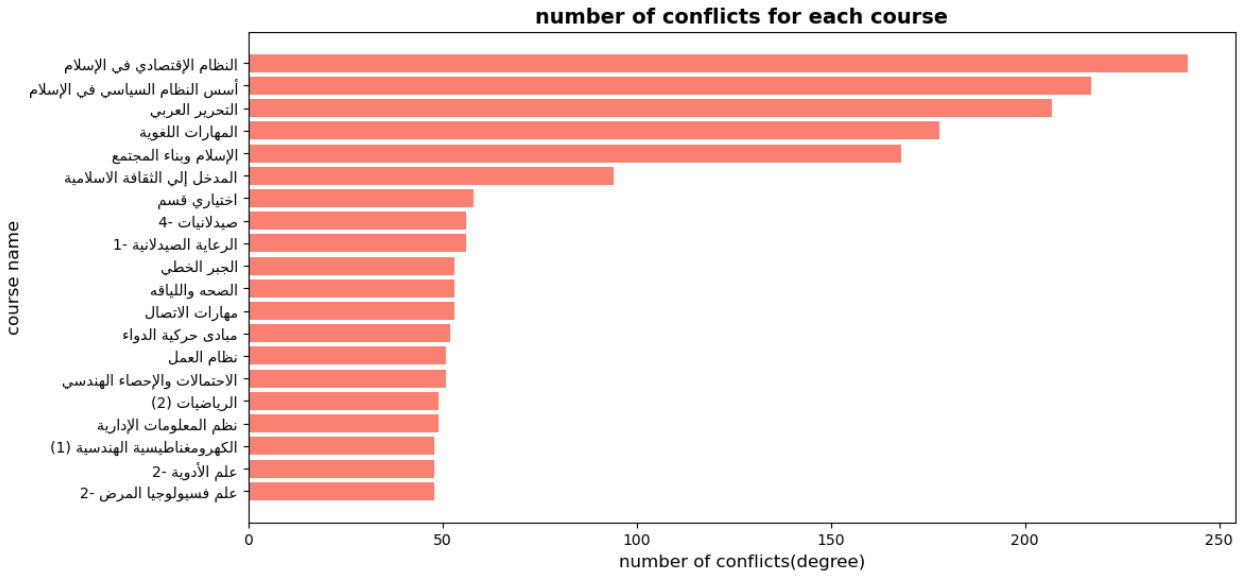
```
GRAMMAR2                          1
                                 ..
7                نظم التشغيل
53       نظم المعلومات الإدارية
12      نظم المعلومات المحاسبية
23            هندسة البرمجيات
8           هندسة الجهد العالي
Name: رقم الطالب, Length: 456, dtype: int64
```

```
import arabic_reshaper
from bidi.algorithm import get_display

students_in_courses_sorted = students_in_courses_df.sort_values(by="
عدد الطلاب", ascending=False)
top_20_courses=students_in_courses_sorted.head(20)

# Reshape Arabic text for correct display
top_20_courses["اسم المقرر"] = top_20_courses["اسم المقرر"].apply(lambda
x: get_display(arabic_reshaper.reshape(x)))

# Create the plot
plt.figure(figsize=(10, 5))
plt.barh(top_20_courses["اسم المقرر"], top_20_courses["عدد الطلاب"],
color="steelblue")
plt.ylabel("course name", fontsize=12)
plt.xlabel("number of students", fontsize=12)
plt.title("top 20 courses according to number of students",
fontsize=14, fontweight="bold")

# Rotate x-axis labels for better readability

plt.gca().invert_yaxis()
# Show the plot
plt.show()
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_21536\1907420875.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  top_20_courses["اسم المقرر"] = top_20_courses["اسم
المقرر"].apply(lambda x: get_display(arabic_reshaper.reshape(x)))
```




```
course_degrees = dict(G.degree())  # Get degree for each course
(number of conflicts (shared students))
# Convert to DataFrame for sorting and visualization
degree_df = pd.DataFrame(course_degrees.items(), columns=[" ,"اسم المقرر
درجة المقرر""])
degree_df = degree_df.sort_values(by="درجة المقرر", ascending=False)

# Reshape Arabic text for correct display
degree_df["اسم المقرر"] = degree_df["اسم المقرر"].apply(lambda x:
get_display(arabic_reshaper.reshape(x)))
degree_df= degree_df.head(20)

# Plotting
plt.figure(figsize=(12, 6))
plt.barh(degree_df["اسم المقرر"], degree_df["درجة المقرر"],
color="salmon")
plt.xlabel("number of conflicts(degree)", fontsize=12)
plt.ylabel("course name", fontsize=12)
plt.title("number of conflicts for each course", fontsize=14,
fontweight="bold")
plt.gca().invert_yaxis()  # Highest first
plt.show()
```




```python
import random
# Define the periods and their corresponding section numbers
periods_ordered = [
    ("فترة أولى صباحية","رقم 1"),
    ("فترة ثانية صباحية","رقم 2"),
    ("فترة أولى مسائية","رقم 3"),
    ("فترة ثانية مسائية","رقم 4"),
]

# Create a dictionary to store the number of students for each course
course_student_count = {}

# Iterate through each row in the DataFrame
for _, row in merged_df.iterrows():
    course_id = row['رمز المقرر']  # Get the course code
    course_name = row['اسم المقرر']  # Get the course name
    student_id = row['رقم الطالب']  # Get the student ID

    # If the course is not in the dictionary, initialize it
    if course_id not in course_student_count:
        course_student_count[course_id] = {
            'name': course_name,
            'students': set(),
            'period': random.choice([p[1] for p in periods_ordered])
# Random period
        }

    # Add the student ID to the set for the course
    course_student_count[course_id]['students'].add(student_id)

# Prepare structured data for Excel
```

```
excel_data = []

for section_number, period in periods_ordered:
    # Add section title
    excel_data.append([f"اليوم {section_number}", ""])
    excel_data.append([f"الفترة الصباحية:", ""])
    excel_data.append(["مقررات " + period, ""])
    excel_data.append(["", ""])
    excel_data.append(["العدد","أسم المقرر","رمز المقرر"])  # Table headers

    # Add courses belonging to this period
    for course, details in course_student_count.items():
        if details["period"] == period:
            excel_data.append([course, details["name"],
len(details["students"])])

    # Add an empty row after each period
    excel_data.append(["", "", ""])

# Convert list to DataFrame
df_courses = pd.DataFrame(excel_data)

# Save to Excel file
file_path = r"D:/output.xlsx"
df_courses.to_excel(file_path, index=False, header=False)  # No
default headers

print(f"Excel file '{file_path}' has been created successfully!")

df_courses
```

```
Excel file 'D:/output.xlsx' has been created successfully!
```

	0	1	2
0	اليوم رقم 1		None
1	الفترة الصباحية:		None
2	مقررات فترة أولى صباحية	None	
3			None
4	العدد	أسم المقرر	رمز المقرر
..	...	...	...
499	6	معمل من مقررات العلوم – القائمة "أ"	معمل2
500	4156	مقدمة في تصميم الدوائر المتكاملة	هال
501	4551	مقدمة للتحكم الرقمي	هال
502	4931	مواضيع مختارة في علوم الحاسب	عال
503			

```
[504 rows x 3 columns]
```