الجامعة الافتراضية السورية

ماجستير : **علوم الحاسوب MCS**
المقرر : **PRB**
المدرس : **الدكتور طارق الساطي**

الفصل الدراسي : **F22**

اسم الطالب	الرقم الجامعي	الصف
كرم دالي	**259063**	**C2**

السؤال الأول:

استخدام اساسيات اللغة والبنى الشرطية:

تم انجاز المطلوب ضمن هذا السؤال بتعريف توابع خاصة لكل طلب وذلك لتسهيل تصحيح البرنامج وجعله أكثر وضوحا.

حيث تم تعريف التوابع التالية:

`def setValueAge():`	تابع يطلب من المستخدم ادخال عمره ويدقق القيمة المدخلة
`def setValueSex():`	تابع يطلب من المستخدم ادخال جنسه ويدقق القيمة المدخلة
`def setValueEmp():`	تابع يطلب من المستخدم تحديد اذا ماكان مظف أم لا
`def identityCard():`	تابع يطلب من المستخدم تحديد فيما اذا يحمل بطاقة شخصية أم لا
`def reqPapers(age,sex,emp,iden):`	تابع يأخذ القيم المحددة من قبل المستخدم وعلى أساسها يحدد الأوراق المطلوبة لاستصدار جواز السفر

ضمن كل تابع يوجد حلقة لانهائية تطلب من المستخدم ادخال البيانات المطلوبة وفي حال عدم مطابقتها فانها تقوم بإعادة طلبها حتى تتطابق مع ماهو مطلوب من المستخدم وتكسر الحلقة ويعيد كل تابع القيمة المدخلة من المستخدم لحفظها ضمن متغير.
ماعدا التابع :def reqPapers(age,sex,emp,iden) حيث أنه يأخذ القيم المدخلة من قبل المستخدم ولايعيد أي قيمة فقط يظهر المستندات المطلوبة لاستصدار جواز السفر.

تم ضمن التابع الأول :`def setValueAge()` حيث تم استخدام `try , except` للتحقق فيما اذا كانت القيمة المدخلة من المستخدم هي رقمية. لتجنب حدوث أخطاء ضمن البرنامج.

- استدعاء التوابع لاظهار ماهي المستندات المطلوبة:

```
-   age = setValueAge() #ask the for the user's age
-   if age >= 18 : #if the user age is equal or older than 18 ask if
    he has his identity card and his employment status
-       iden = identityCard()
-       emp = setValueEmp()
-   else: #if the user is younger than 18 we don't neet to ask fro his
    identity card and his employment status
-       iden = "not required"
-       emp = "not required"
-   sex = setValueSex()
-
-   #show the required document according to the entered data.
-   reqPapers(age,sex,emp,iden)
```

تم في البداية الطلب من المستخدم ادخال عمره **في حال كان العمر أقل من ١٨ عام** فان البرنامج لن يطلب من المستخدم بيان وضع بطاقته أو عمله.

وبعدها تمرير قيم المتغيرات الى التابع `reqPapers(age,sex,emp,iden)` لاظهار المستندات المطلوبة.

السؤال الثاني:

استخدام البنى التكرارية والأرقام العشوائية:

لانشاء لعبة التخمين حيث أنه على المستخدم تخمين الرقم المولد بشكل عشوائي من قبل الحاسوب ضمن خمس محاولات وفي حال التخمين الخاطء ضمن هذه المحاولات فان الحاسوب سيخبر المستخدم اذا كان توقعه أكبر أو اصغر من القيمة المدخلة.

في البداية تم استدعاء مكتبة random	`import random`
اظهار رسالة ترحيبية للمستخدم وشرح لقواعد اللعبة	`print("I will guess a numb……..etc`
توليد رقم عشوائي بين ١ و ٢٠ واسناده الى المتغير target	`target = random.randint(1,20)`
رسالة توضح بدء اللعبة	`print("The game is started\n")`
تم تعريف متغير من النوع list يحتوي على خمس عناصر مساوية لعدد التكرارات مع ترتيبها لاظهارها للمستخدم واعلامع برقم المحاولة	`chances = ["first","second","thrird","fourth","fifth"]`
حلقة تكرارية على عدد القيم ضمن المتحول chances	`for i in chances:`
الطلب من المستخدم ادخال قيمة عددية	`guessedNumber = input("\nEnter your {} geuss : ".format(i))`

التحقق من أن القيمة المدخلة من قبل المستخدم هي قيمة رقمية. في حال لم تكن كذلك سيتم اظهار رسالة الى المستخدم وينتقل الى المحاولة التالية.	`try:`
	`  guessedNumber = int(guessedNumber)`
	`except:`
	`  print("you didn't enter a number!")`
	`  continue`

في حال كان تخمين المستخدم صحيح سيتم اظهار رسالة تخبه بفوزه وتنكسر الحلقة بتعليمة break	`if guessedNumber == target:`
اذا وصل المستخدم الى التكرار الأخير ولم يخمن الرقم بشكل صحيح سيتم اعلامه بخسارته وانهاء الحلقة	`if i == "fifth"`
في حال كانت القيمة المدخلة أكبر من الرقم العشوائي فسوف تظهر رسالة للبرنامج تعلمه بهذا	`if guessedNumber > target:`
في حال كانت القيمة المدخلة أصغر من الرقم العشوائي فسوف تظهر رسالة للبرنامج تعلمه بهذا	`elif guessedNumber < target:`

السؤال الثالث:

استخدام بنى المعطيات:

الطلب الأول:

تم تضمين البيانات ضمن متحول من النوع `dictionary` وتم تسميته `data` وجعل أسماء الأشهر مفاتيح له كما يلي:

```
data = {
    "jan": [("tea",3000), ("cofee",5000), ("sugar",2300)],
    "feb": [("tea",2800), ("cofee",5500), ("sugar",2300)],
    "mar": [("tea",1900), ("cofee",4700), ("sugar",4000)],
    "apr": [("tea",3300), ("cofee",6000), ("sugar",3600)],
    "may": [("tea",3500), ("cofee",6100), ("sugar",3200)],
    "jun": [("tea",1900), ("cofee",4700), ("sugar",4000)],
    "jul": [("tea",2000), ("cofee",2300), ("sugar",1900)],
    "aug": [("tea",3500), ("cofee",3200), ("sugar",3600)],
    "sep": [("tea",6000), ("cofee",4000), ("sugar",4500)],
    "oct": [("tea",1200), ("cofee",2100), ("sugar",2000)],
    "nov": [("tea",3500), ("cofee",3200), ("sugar",3600)],
    "dec": [("tea",4000), ("cofee",8000), ("sugar",4400)]
    }
```

تم اسناد قيمة لكل مفتاح من النوع `list` تحوي على `tuples` اسم المنتج وقيمة البيع الاجمالية منه.

الطلب الثاني:

حساب وطباعة متوسط المبيعات لكل شهر من الأشهر.

```
for month, products in data.items() :
    sum = 0
    i = 0
    for product in products:
        sum = sum + product[1]
        i = i+1
    mean = sum/i #calculate the mean
    print("The mean of selling products in {} is
{:.2f}".format(month,mean))
```

لتحقيق المطلوب تم انشاء حلقة تكرارية وتم الاستعانة بتعليمة `.items()` للحصول على أسماء المفاتيح والقيم الخاصة بها. ثم تم انشاء حلقة مضمنة ضمن الحلقة الأولى لحساب مجموع المبيعات كل مادة في كل شهر مع متغير لحساب عدد المنتجات. وبنهاية كل تكرار من الحلقة الثانية يتم طباعة المتوسط الحسابي الخاص بكل شهر.

الطلب الثاث:

حساب وطباعة متوسط المبيعات لكل منتج من المنتجات.

```
tea = 0
cofee = 0
sugar = 0
for month, products in data.items():
    product = dict(products)
    tea = tea + product["tea"]
    cofee = cofee + product["cofee"]
    sugar = sugar + product["sugar ]
print("the mean of the total sales of TEA during the year is
{:.2f}\n".format(tea/12))
print("the mean of the total sales of COFEE during the year is
{:.2f}\n".format(cofee/12))
print("the mean of the total sales of SUGAR during the year is
{:.2f}\n".format(tea/12))
```

لتحقيق المطلوب تم تحديد ثلاث متحولات كل منها لتخزين مجموع مبيعات كل منتج خلال العام. تم انشاء حلقة تكرارية للمرور على كل قيم المفاتيح الخاصة بمتحول لـdata بعدها تم تحول هذه القيم من النوع list الى النوع dictionary باستخدام تعليمة dict(products) . وتم جمع القيم الخاصة بكل منتج وفي نهاية الحلقة تم طباعة المتوسط الحسابي لمبيعات كل منتج بعد تقسيم المجاميع الناتجة عن الحلقة التكرارية على ١٢ عدد الأشهر.

الطلب الرابع:

طباعة الشهر الذي حقق أعلى مبيعات.

```
totalSales={}
for month, products in data.items() :
    sum = 0
    for product in products:
        sum = sum + product[1]
    totalSales[month] = sum
month = list(totalSales.keys())
sumOfSales = list(totalSales.values())
print("the month with the highest sales is {} with \
total sales equal to\
 {}".format(month[sumOfSales.index(max(sumOfSales))],max(sumOfSales)))
```

لتحقيق المطلوب تم في البداية تعريف متغير من النوع dictionary باسم totalSales سيتم حفظ مجموع مبيعات كل شهر ضمن مفتاح باسم الشهر المعني. تم انشاء حلقة تكرارية باستخدام الأداة .items() وذلك ليتم المرور بقيم جميع المفاتيح بسهولة. بعدها تم انشاء حلقة للمرور على جميع قيم المبيعات ضمن كل شهر وجمعها. واسناد القيمة تحت مفتاح اسم الشه المعني بالمتحول totalSales . تم بعدها تحويل مفاتيح وقيم ذلك المتحول الى متغيرات من النوع list وبعدها تم طباعة اسم الشهر ذو أعلى مبيعات عبر التعليمة

month[sumOfSales.index(max(sumOfSales))]

الطلب الخامس:

طباعة اسم المنتج الذي حقق أعلى مبيعات خلال السنة.

```
tea = 0
cofee = 0
sugar = 0

sumSales ={"tea":0,"cofee":0,"sugar":0}
for month, products in data.items():
    product = dict(products)
    tea = tea + product["tea"]
    cofee = cofee + product["cofee"]
    sugar = sugar + product["sugar"]
sumSales["tea"]=tea
sumSales["cofee"]=cofee
sumSales["sugar"]=sugar
productName = list(sumSales.keys())
sumOfSales = list(sumSales.values())
print("the product with the highest sales is {} with \
total sales equal to\
 {}".format(productName[sumOfSales.index(max(sumOfSales))],max(sumOfSale
s)))
```

لتحقيق هذا الطلب فق تم تعريف ثلاث متغيرات . وتم أيضا كما في الأسئلة السابقة انشاء حلقة تكرارية لقرئة المفاتيح الوقيم المرتبطة بها تم تحويل كل قيمة الى `dict` وتم جمع القيم مع بعضها ليصار الى الحصول على المجموع الكلي لمبيعات كل منتج تم حفظ كل قيمة ضمن متغير اسمه `sumSales` من النوع `dictionary` كل قيمة مرتبطة باسم مفتاح يحمل اسم المنتج. تم تحويل المفاتيح و القيم لهذا المتحول الى متغيرين من النوع `list` وبالتالي الحصول على المنتج صاحب أكبر قيمة مبيعات من خلال التعليمة.
productName[sumOfSales.index(max(sumOfSales))]

السؤال الرابع:

التوابع والملفات والوحدات البرمجية:

الطلب الأول:

تم انشاء وحدة برمجية `module` وتسميتها بـ `unit` تحتوي على ٦ توابع التوابع الأربعة الأولى منها مشابهة تقريبا لطريقة عمل البرنامج في الطلب السابق وهي :

تابع يطلب يأخذ مدخلات هي من النوع `dict` واسم الشهر ليقوم بحساب المتوسط الحسابي لمبيعات ذلك الشهر	`def monthMean(data,month):`

`def productMean(data,product):`	تابع يقوم بأخذ مدخلات من النوع dict واسم المنتج ليقوم بحساب المتوسط الحسابي لمبيعات ذلك المنتج خلال العام
`def hightestMonth(data):`	تابع يأخذ متحول من النوع data ويخرج أسم الشهر الذي تمت به أكبر قيمة بالمبيعات
`def highestProduct(data):`	تابع يأخذ متحول من النوع data ويخرج اسم المنتج الأكثر مبيعا خلال العام

أما فيما يتعلق بالتابعين الأخيرين فسيتم شرح كل منهما:

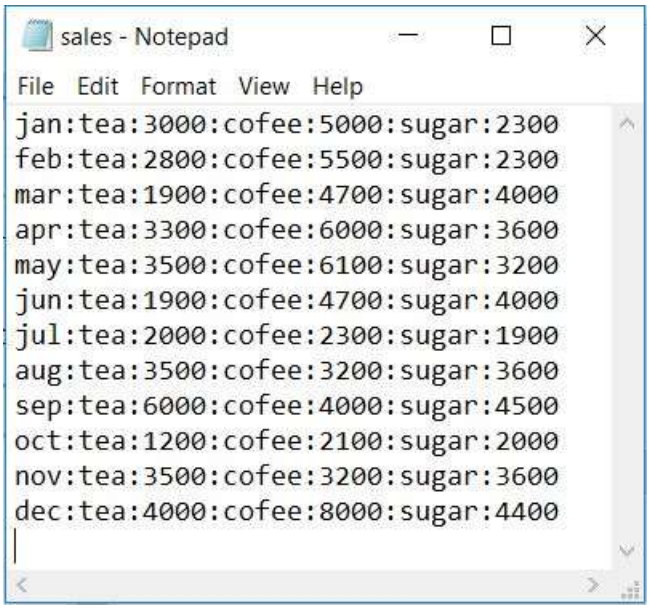
التابع الخاص بكتابة قيمة المتحول من النوع dictionary الى ملف يحمل الصيغة .txt

```
def DictToTxt(data, fileName):
    handl = open(fileName, "w")
    for month,value in data.items():
        handl.write(month + ":" + str(value[0][0])+":"+str(value[0][1]) +":"+ str(value[1][0]) +":"+\
                    str(value[1][1])+":"+ str(value[2][0]) +":"+ str(value[2][1])+"\n")
    handl.close()
```

تم فتح الملف بوضع الكتابة . وتم انشاء حلقة تكرارية على مفاتيح وقيم المتحول المدخل الى التابع والذي هو من النوع dictionary وضمن كل تكرار يتم كتابة سطر جديد الى الملف كما يلي حيث يتم الفصل بين القيم بـ": "

key	First value of the first tuple	Second value of the first tuple	First value of the second tuple	Second value of the second tuple	First value of the third tuple	Second value of the third tuple	Newline “\n”

الملف الناتج عن التابع السابق يكون كما هو موضح بالصورة التالية:



التابع الأخر وهو التابع الذي يأخذ اسم ملف ويأقرؤه ويعيد متحول من النوع`dictionary`

```
def txtToDict(fileName):
    data={}
    handl = open(fileName, "r")
    for line in handl:
        values = line.split(":")
        data[values[0]] =
[tuple([values[1],int(values[2])]),tuple([values[3],int(values[4])]),tup
le([values[5],int(values[6])])]
    return data
```

يتم تعريف متغير من النوع `dictionary` باسم `data` لحفظ القيم التي ستمرر له ضمن كل تكرار من الحلقة التالية. يتم فتح الملف للقراءة فقط يتم انشاء حلقة تكرارية لقرائة كل سطرضمن الملف. يتم فصل اسطر عبر التعليمة

`split(":")` لتعطينا متحول من النوع `list` أول قيمة فيه هي المفتاح اسم الشهر القيمة الثانية هي اسم المنتج الأول القيمة الثالثة اجمالي مبيعات المنتج الأول القيمة الرابعة اسم المنتج الثاني ... وهكذا . يتم جمعهم بالطريقة المناسبة وحفظها ضمن المتغير `data` .

الطلب الثاني :

فيما يلي كود البرنامج المستخدم في استدعاء الوحدة البرمجية التي تم تشكيلها من التوابع السابقة واستخدامها ضمن البرنامج عبر تعليمة `from unit import *`

```
#Qeustion 4
#selling data analysier using modules
```

```python
#Damascus on 04/20/2023
#import the needed module for the functions used in this code
from unit import *

#Resolve the question three using the defined functions.
data = {
    "jan": [("tea",3000), ("cofee",5000), ("sugar",2300)],
    "feb": [("tea",2800), ("cofee",5500), ("sugar",2300)],
    "mar": [("tea",1900), ("cofee",4700), ("sugar",4000)],
    "apr": [("tea",3300), ("cofee",6000), ("sugar",3600)],
    "may": [("tea",3500), ("cofee",6100), ("sugar",3200)],
    "jun": [("tea",1900), ("cofee",4700), ("sugar",4000)],
    "jul": [("tea",2000), ("cofee",2300), ("sugar",1900)],
    "aug": [("tea",3500), ("cofee",3200), ("sugar",3600)],
    "sep": [("tea",6000), ("cofee",4000), ("sugar",4500)],
    "oct": [("tea",1200), ("cofee",2100), ("sugar",2000)],
    "nov": [("tea",3500), ("cofee",3200), ("sugar",3600)],
    "dec": [("tea",4000), ("cofee",8000), ("sugar",4400)]
    }

DictToTxt(data,"salesData") #write the dictionary data to a file named
"salesData".
newData = txtToDict("salesData") #read the data and assign it to a new
dictionary.

#prith the mean of each month.
for month in newData.keys():
    print("The mean of selling products in {} is
{:.2f}".format(month,monthMean(newData,month)))

#print the mean of each product
for product in ["tea","cofee","sugar"]:
    print("the mean of the total sales of {} during the year is
{:.2f}\n".format(product,productMean(newData,product)))

#print the month with the highest sales
print("the month with the highest sales is
{}".format(hightestMonth(newData)))

#print the product with the highest sales.
print("the product with the highest sales is
{}".format(highestProduct(newData)))
```