G12

المسار التكنولوجي

علوم الحاسب

الفصل الدراسي الأول 2021-2022

الوحدة الأولي : هياكل البيانات المتقدمة

1- المكدس والطابور Stack & Queue

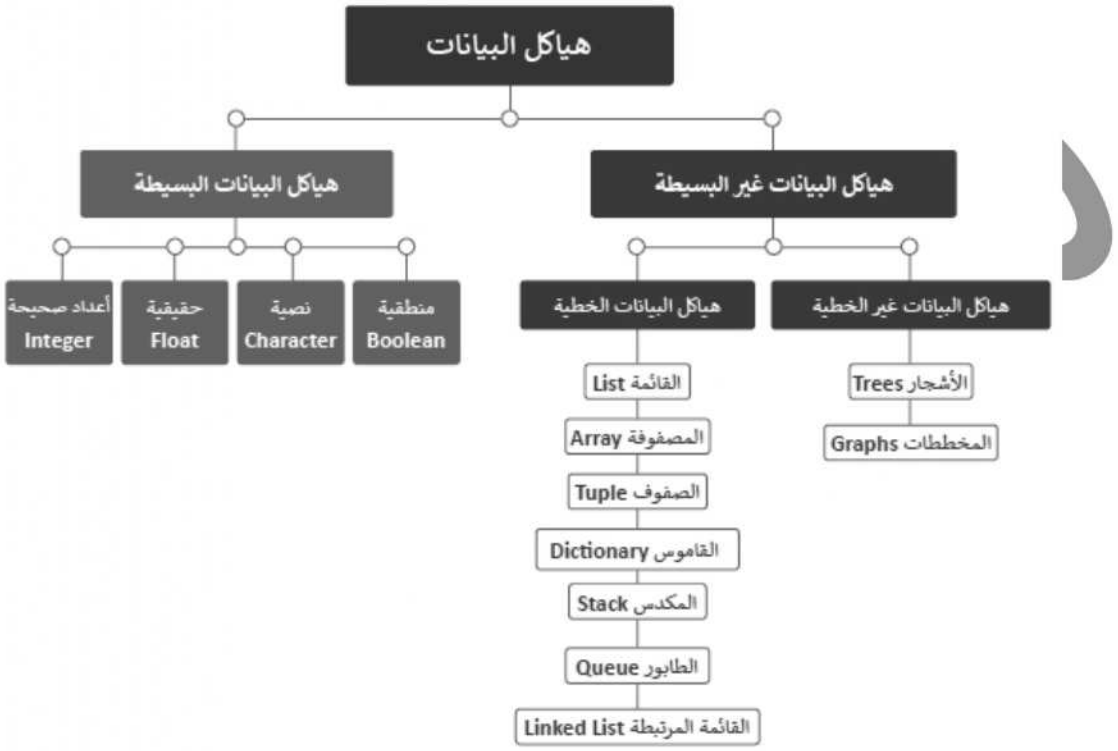
اسم الطالب : .....

اعداد M.A
77914410

الدرس الأول: المكدس والطابورStack & Queue

- هياكل البياناتData Structures :

هي تقنية لتخزين البيانات في الذاكرة وتنظيم عملية الوصول اليها لاستخدامها بكفاءة وسهولة.



- هياكل البيانات البسيطة:

تتضمن هذه الهياكل قيماً بسيطة من البيانات تخبر مترجم لغة البرمجة بنوع البيانات التي يمكن للمتغير تخزينها داخله.

- هياكل البيانات غير البسيطة:

هي هياكل تختص بتخزين مجموعة من القيم بحيث يتم إنشاؤها بواسطة المبرمج وليست معرفة ضمنياً في بايثون

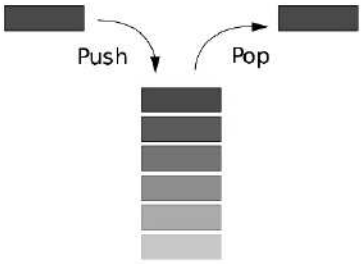
- تخزن البيانات الخطية عناصر البيانات بتسلسل معين بينما لا تحتوي هياكل البيانات غير الخطية على ارتباط تسلسلي بين عناصر البيانات.

- المُكدس:

هو مجموعة مرتبة من العناصر أطلق عليها اسم المُكدس لتكدس عناصره علي التوالي .

- العنصر الذي تم إضافته مؤخراً (آخر عنصر) يتم الوصول إليه أولاً وهذا المبدأ يسمى

آخر من يدخل أول من يخرج Last-In-First-Out LIFO



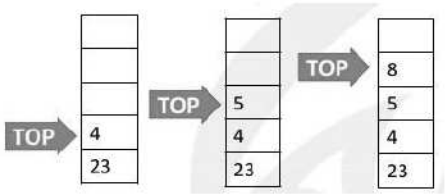
- هناك عمليتان رئيسيتان خاصتان بالمُكدس:

Push : تستخدم هذه العملية للإدخال عنصر إلي أعلى المُكدس.

Pop: تستخدم هذه الدالة لحذف العنصر أعلى (قيمة) المُكدس.

- عملية الإضافة Push:

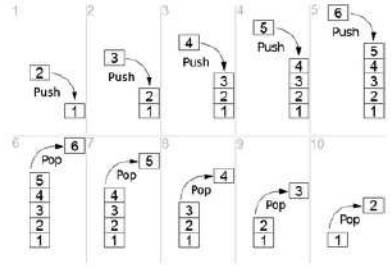
يستخدم المُكدس مؤشراً يسمى Top ويشير إلى العنصر أعلى المكدس.



عند إضافة عنصر جديد إلى المُكدس :

- **تزيد قيمة المؤشر بمقدار 1.**
- **تتم إضافة العنصر الجديد إلى أعلى المُكدس.**

- يجب التحقق من امتلاء المكدس قبل إضافة أي عنصر حيث أن للمكدس سعة محددة تعتمد على ذاكرة الحاسوب.



- عملية الإزالة Pop:

عند إضافة عنصر جديد إلى المُكدس:

- **تنقص قيمة المؤشر بمقدار 1.**
- **تتم إزالة العنصر الجديد إلى أعلى المُكدس.**

- إذا أردنا إزالة عنصر من المكدس يجب التحقق من احتواء المُكدس على عنصر واحد على الأقل.

- تمثيل المُكدسات بلغة Python:

يتم تمثيل المُكدسات بلغة Python باستخدام القائمة List والتي بدورها تقدم بعض العمليات الجاهزة لاستخدامها مع المكدسات.

- عمليات المُكدسات:

العملية	الوصف
listName.append(x)	إضافة العنصر x إلى نهاية القائمة التي تمثل المكدس.
listName.pop()	إزالة أخر عنصر من القائمة التي تمثل المكدس.

- مثال 1:

1- إنشاء مُكدس لحفظ مجموعة من الأعداد

(كتابة مقطع برمجي بلغة Python لحل المثال السابق بحيث يتم إنشاء المكدس القائمة myStack=[1,21,32,45] ثم استخدام عملية الـ Pop مرتين لحذف آخر عنصرين ثم إضافة عنصر جديد إلى المُكدس وليكن 78)

```
myStack=[1,21,32,45]
print("Initial stack:", myStack)
print(myStack.pop())
print(myStack.pop())
print("The new stack after pop:", myStack)
myStack.append(78)
print("The new stack after push:", myStack)
```

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 201
4)] on win32
Type "copyright", "credits" or "license()"
>>>
 RESTART:

Initial stack: [1, 21, 32, 45]
45
32
The new stack after pop: [1, 21]
The new stack after push: [1, 21, 78]
>>>
```

- مثال 2:

تعديل المقطع البرمجي السابق دون استخدام جمل الطباعة Print لعملية الإزالة Pop

```
File Edit Format Run Options Window Help
myStack=[1,21,32,45]
print("Initial stack:", myStack)
myStack.pop()
myStack.pop()
print("The new stack after pop:", myStack)
myStack.append(78)
print("The new stack after push:", myStack)
```

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 20
4)] on win32
Type "copyright", "credits" or "license()"
>>>
 RESTART:

Initial stack: [1, 21, 32, 45]
The new stack after pop: [1, 21]
The new stack after push: [1, 21, 78]
```

- مثال 3:

كتابة برنامج بحيث يتم إنشاء المكدس (القائمة) myStack =[1,21,32,45] ثم الوصول للقيمة 21 في المكس وطباعتها باستخدام العملية Pop

```
myStack=[1,21,32,45]
print("Initial stack:",
myStack.pop()
myStack.pop()
print(myStack.pop())
```

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.0 (v3.7.0:1bf9cc509
4)] on win32
Type "copyright", "credits" or
>>>
 RESTART: D:

Initial stack: [1, 21, 32, 45]
21
```

- مثال 4:

سيقوم المستخدم بإنشاء مكدس ثم يقوم بإضافة عنصر ثم إزالته وسيعرض البرنامج قائمة تسأل المستخدم عن الإجراء الذي يود القيام بهفي كل مرة.

يوجد نوعين من الدوال :

1- الطباعة مباشرة 2- إعادة قيمة في الدالة واستخدامها في البرنامج

```
def push(stack,item):
    stack.append(item)
def pop(stack):
    return stack.pop()
def isEmpty(stack):
    return len(stack)==0
def createStack():
    return []

newStack=createStack()
while True:
    print("The stack so far is: ",newStack)
    print("------------------------------")
    print ("Choose 1 for push")
    print ("Choose 2 for pop")
    print ("Choose 3 for end")
    print("------------------------------")
    choice=int(input("Enter your choice:"))
    while choice!=1 and choice!=2 and choice!=3:
        print ("Error")
        choice=int(input("Enter your choice: "))
    if choice==1:
        x=int(input("Enter item for push: "))
        push(newStack,x)
    elif choice==2:
        if not isEmpty(newStack):
            print ("The pop item is:",pop(newStack))
        else:
            print ("The stack is empty")
    else:
        print ("End of program")
        break;
```

```
The stack so far is:  []
------------------------------
Choose 1 for push
Choose 2 for pop
Choose 3 for end
------------------------------
Enter your choise:1
Enter item for push: 26
The stack so far is:  [26]
------------------------------
Choose 1 for push
Choose 2 for pop
Choose 3 for end
------------------------------
Enter your choise:1
Enter item for push: 18
The stack so far is:  [26, 18]
------------------------------
Choose 1 for push
Choose 2 for pop
Choose 3 for end
------------------------------
Enter your choise:1
Enter item for push: 23
The stack so far is:  [26, 18, 23]
------------------------------
Choose 1 for push
Choose 2 for pop
Choose 3 for end
------------------------------
Enter your choice:
```

```
------------------------------
Choose 1 for push
Choose 2 for pop
Choose 3 for end
Enter your choice:2
The pop item is: 23
The stack so far is:  [26, 18]
------------------------------
Choose 1 for push
Choose 2 for pop
Choose 3 for end
------------------------------
Enter your choice:2
The pop item is: 18
The stack so far is:  [26]
------------------------------
Choose 1 for push
Choose 2 for pop
Choose 3 for end
------------------------------
Enter your choice:3
End of program
>>>
```

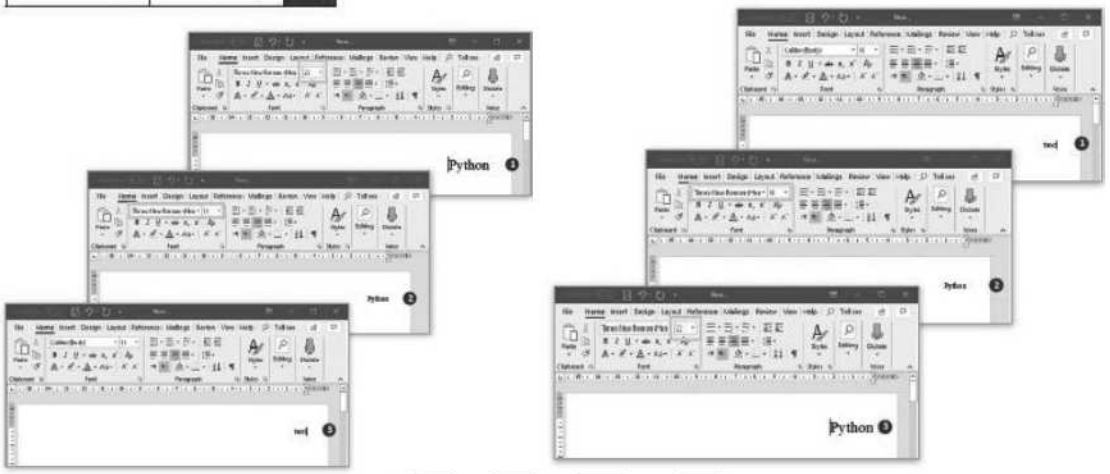
- استخدام الأمر break للخروج من البرنامج ومقاطعة التكرار While عند اختيار رقم 3

- تطبيقات المكدسات في الحاسوب:

من الأمثلة على المُكدس أمر التراجع Undo من خلال إضافة الحدث باستخدام Push والتراجع باستخدام Pop

	الحدث	العملية
3	تغيير الحجم	push
2	تغيير نوع الخط	push
1	كتابة النص	push

مكدس التراجع		
	الحدث	العملية
1	تغيير حجم الخط	pop
2	تغيير نوع الخط	pop
3	كتابة النص	pop



ملاحظة: من مظاهر الاختلاف بين المكدس والطابور

المكدس : الإضافة والحذف من نفس الطرف

الطابور : الإضافة من طرف والحذف من طرف آخر

- الطابور Queue:

هو هيكل بيانات يتبع قاعدة First In First Out (FIFO) أي أن أول من يدخل هو أول من يخرج بمعني أن كل عنصر في قائمة الطابور يتم التعامل معه حسب ترتيب وصوله.

- هناك عمليتان رئيسيتان خاصتان بالطابور:

Enqueue: وهي عملية الإضافة حيث يتم إضافة عنصر إلى آخر الطابور

Dequeue: وهي عملية الإزالة حيث يتم إزالة عنصر من مقدمة الطابور

المؤشر Pointer : هو متغير يشير إلى عنوان عنصر من عناصر الطابور

للطابور مؤشرين :

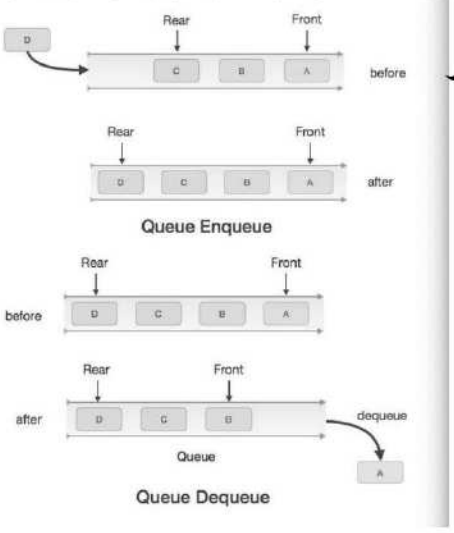
Front:يحمل أصغر قيمة وهي عنوان العنصر الأول

Rear:يحمل أكبر قيمة وهي عنوان العنصر الأخير

لا يمكننا إضافة أو إزالة عنصر من منتصف الطابور

يجب التحقق من وجود مساحة كافية في الطابور لإضافة عنصر جديد

يجب التأكد من وجود عنصر واحد على الأقل ليتم إزالته

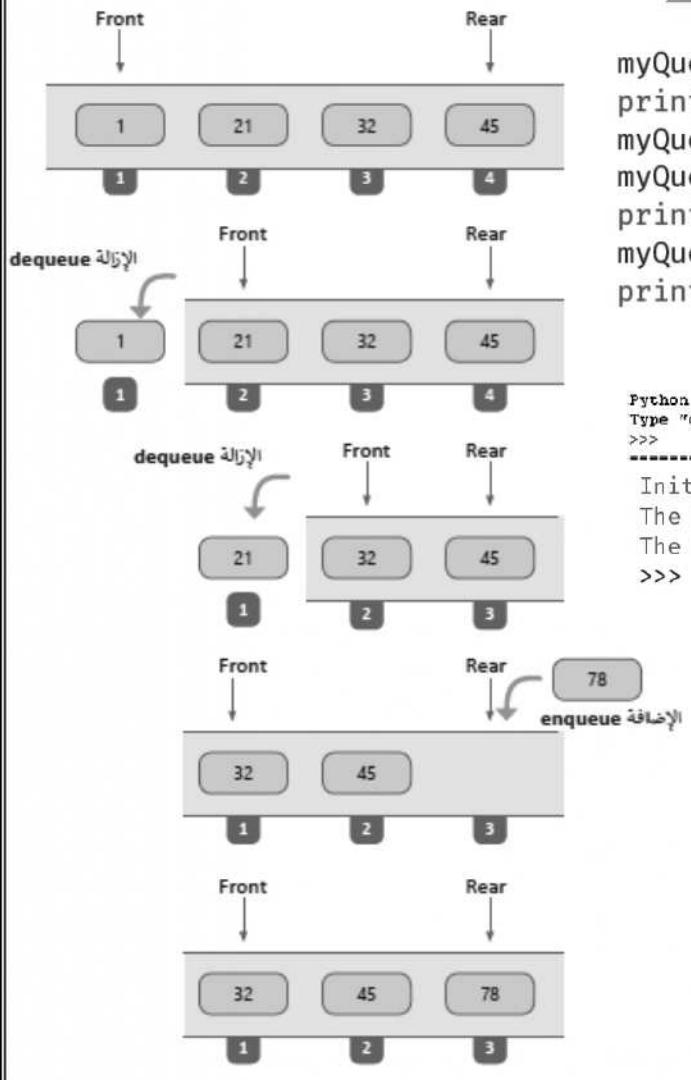


- تمثيل الطابور بلغة Python:

عمليات الطابور	
العملية	الوصف
listName.append(x)	إضافة العنصر x إلى نهاية القائمة التي تمثل الطابور.
listName.pop(0)	إزالة أول عنصر من القائمة التي تمثل الطابور، حيث يشير (0) إلى أول عنصر.

مقارنة عملية list.pop() وعملية list.pop(0)	
العملية	الوصف
listName.pop()	إذا كان وسيط الدالة فارغًا، ستتم إزالة العنصر من نهاية القائمة التي تمثل المكدس.
listName.pop(0)	إذا كان وسيط الدالة صفرًا، ستتم إزالة أول عنصر من عناصر القائمة التي تمثل الطابور.

مثال كيفية تمثيل الطابور بلغة البايثون:




```
myQueue=[1,21,32,45]
print("Initial queue:", myQueue)
myQueue.pop(0)
myQueue.pop(0)
print("The new queue after pop:", myQueue)
myQueue.append(78)
print("The new queue after push:", myQueue)
```

التطبيق:

```
Python 3.7.0 (v3.7.0:1bf9cc5093 [MSC v.1914 32 bit (Intel)] on win3
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/python/examples.py ==========
Initial queue: [1, 21, 32, 45]
The new queue after pop: [32, 45]
The new queue after push: [32, 45, 78]
>>>
```

ما ذا سيحدث إذا حاولنا إزالة عنصر من طابور فارغ
سوف يظهر خطأ وذلك لأننا قمنا بحذف عنصر من طابور فارغ

```
myQueue=[1,21,32,45]
print("Initial queue:", myQueue)
a=len(myQueue)
print("size of queue",a)
#empty the queue
for i in range(a):
    myQueue.pop(0)
print(myQueue)
myQueue.pop(0)
```

```
>>>
========== RESTART: C:/python/examples.py ==========
Initial queue: [1, 21, 32, 45]
size of queue 4
[]
Traceback (most recent call last):
  File "C:G12_CS_m1/python/example.py", line 9, in <module>
    myQueue.pop(0)
IndexError: pop from empty list
>>>
```

- أحد الأمثلة الشائعة على الطابور في علوم الحاسب هو طابور الطابعة

مثال : شركة تحتوي على 20 جهاز حاسوب متصلين بطابعة واحدة تتم معالجة المهام بطريقة

First In First Out

- تمثيل المكدس والطابور باستخدام Queue Library:

تتضمن مكتبة Queue القياسية بعض الدوال الجاهزة للاستخدام التي يمكن استخدامها مع المكدس أو الطابور

العمليات في مكتبة Queue القياسية	
العملية	الوصف
queueName=queueName.Queue()	انشاء طابور جديد تحت مسمى queueName.
queueName.put(x)	إضافة العنصر x إلى الطابور.
queueName.qsize()	إرجاع حجم الطابور.
queueName.get()	جلب وإزالة أول عنصر من الطابور وآخر عنصر من المكدس.
queueName.full()	ترجع True إذا كانت المكدس ممتلئة وFalse إذا كانت المكدس فارغة ويمكن تطبيقها على الطابور أيضا.
queueName.empty()	ترجع True إذا كانت المكدس فارغة وFalse إذا كانت المكدس ممتلئة ويمكن تطبيقها على الطابور أيضا.

مثال 1:لنستخدم مكتبة Queue القياسية لإنشاء طابور سنقوم بـ

- استدعاء المكتبة queue
- إنشاء طابور فارغ my Queue
- إضافة العناصر a,b,c,d للطابور my Queue
- طباعة عناصر الطابور

```
import queue

myQueue = queue.Queue()

myQueue.put("a")
myQueue.put("b")
myQueue.put("c")
myQueue.put("d")
myQueue.put("e")

#print the items of the queue
for item in list(myQueue.queue):
    print(item)
```

```
Python 3.7.0 (v3.7.0:1bf9c
Type "copyright", "credits
>>>
========== RE:
a
b
c
d
e
>>>
```

مثال 2 : إنشاء طابوراً يتم إدخال 5 قيم من قبل المستخدم أثناء تنفيذ البرنامج ثم طباعة هذه القيم وطباعة حجم الطابور في النهاية.

```
import queue

myQueue = queue.Queue()

#user enters the items of a queue
for i in range(5):
    item=input("enter queue item: ")
    myQueue.put(item)

#print the items of the queue
for item in list(myQueue.queue):
    print(item)

print ("Queue size is :",myQueue.qsize())
```

```
Python 3.7.0 (v3.7.0:1bf9cc5093 [M
Type "copyright", "credits" or "li
>>>
========== RESTART: C
enter queue item: 5
enter queue item: f
enter queue item: 12
enter queue item: b
enter queue item: 23
5
f
12
b
23
Queue size is :5
>>>
```

مثال 3 : إنشاء مقطع برمجي لنتحقق ما إذا كان الطابور فارغاً أو ممتلئاً

```
import queue

myQueue = queue.Queue()

myQueue.put("a")
myQueue.put("b")
myQueue.put("c")
myQueue.put("d")
myQueue.put("e")

checkFull=myQueue.full()
print("Is the queue full?", checkFull)
checkEmpty= myQueue.empty()
print("Is the queue empty?", checkEmpty)
```

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window
Python 3.7.0 (v3.7.0:1bf9cc5093
Type "copyright", "credits" or
>>>
========== RESTART: C:/p
Is the queue full? False
Is the queue empty? False
>>>
```

- تتضمن مكتبة Queue القياسية على بعض الدوال الجاهزة للاستخدام التي يمكن استخدامها مع المكدس أو الطابور

العمليات في مكتبة Queue القياسية	
العملية	الوصف
stackName=stackName.LifoQueue()	إنشاء مكدس جديدة تحت مسمى stackName.
stackName.get()	ازالة اخرعنصر من المكدسة stackName

استخدام مكتبة queue القياسية لإنشاء مكدس فارغ:

```
import queue

myStack = queue.LifoQueue()

myStack.put("a")
myStack.put("b")
myStack.put("c")
myStack.put("d")
myStack.put("e")

for i in range(5):
    k=myStack.get()
    print(k)

checkEmpty= myStack.empty()
print("Is the stack empty?", checkEmpty)
```

```
File Edit Shell Debug Options Window
Python 3.7.0 (v3.7.0:1bf9cc5093
Type "copyright", "credits" or
>>>
========== RESTART:
e
d
c
b
a
Is the stack empty? True
>>>
```

التدريبات

- وضح المقصود بكلاً من:

1- هياكل البيانات:

.....

.....

2- المُكدس Stack:

.....

.....

3- عملية الإضافة للمكدس Push:

.....

.....

4- عملية الإزالة للمكدس Pop:

.....

.....

5- الطابور:

.....

.....

6- عملية الإضافة للطابورEnqueue:

.....

.....

7- عملية الإزالة للطابورDequeue:

.....

.....

9- المؤشر Pointer:

.....

.....

- وضح الوصف للعمليات التالية:

العملية	الوصف
queueName=queueName.Queue()	
queueName.put(x)	
queueName.qsize()	
queueName.get()	
queueName.full()	
queueName.empty()	

- أكمل الجمل التالية:

1- يتبع المكدس القاعدة ..... بينما يتبع الطابور قاعدة .....

2- يدل مؤشر Top إلى ..... بينما يدل مؤشر Rearإلى .....

3- لإضافة عنصر جديد في الطابور نستخدم .....

4- .....هي تقنية لتخزين البيانات في الذاكرة وتنظيم عملية الوصول اليها لاستخدامها بكفاءة وسهولة.

5- ..... هي هياكل تختص بتخزين مجموعة من القيم بحيث يتم إنشاؤها بواسطة المبرمج وليست معرفة ضمنياً في بايثون

6- ..... هو مجموعة مرتبة من العناصر أطلق عليها اسم المُكدس لتكدس عناصره على التوالي.

7- .....هو هيكل بيانات يتبع قاعدة First In First Out (FIFO) أي أن أول من يدخل هو أول من يخرج بمعني أن كل عنصر في قائمة الطابور يتم التعامل معه حسب ترتيب وصوله.

- أكمل الجدول التالي:

العمليات في مكتبة Queue القياسية	
العملية	الوصف
queueName=queueName.Queue()	
queueName.put(x)	
queueName.qsize()	
queueName.get()	
queueName.full()	
queueName.empty()	

مع تمنياتي بدوام النجاح والتفوق