النوع فهم القواميسDictionary :

النوع dictionary (القاموس) هو نوع مضمَّن في بايثون. تربط **القواميس** مفاتيح بقيم على هيئة أزواج، وهذه الأزواج مفيدة لتخزين البيانات في بايثون.

تستخدم **القواميس** عادةً لتخزين البيانات المترابطة، مثل المعلومات المرتبطة برقم تعريف، أو ملفات تعريف المستخدم، وتنشأ باستخدام الأقواس المعقوصة {}.

تبدو القواميس على الشكل التالي:

```
sammy = {'username': 'sammy-shark', 'online': True, 'followers': 987}
```

بالإضافة إلى القوسين المعقوصين، لاحظ وجود النقطتين الرأسيتين (:) في القاموس.

الكلمات الموجودة على يسار النقطتين الرأسيتين هي المفاتيح (keys) التي قد تكون أيَّ نوع بيانات غير قابل للتغيير. المفاتيح في القاموس أعلاه هي:

- username
- online
- followers

المفاتيح في المثال أعلاه عبارة عن **سلاسل نصية**.

تمثل الكلمات الموجودة على يمين النقطتين «القيم» (values). يمكن أن تتألف القيم من أي نوع من البيانات. القيم في القاموس أعلاه هي:

- sammy-shark
- True

قيم القاموس أعلاه هي إمَّا سلاسل نصية أو قيم منطقية أو أعداد صحيحة. سنطبع الآن القاموس sammy:

```
print(sammy)
```

الناتج:

```
{'username': 'sammy-shark', 'followers': 987, 'online': True}
```

نلاحظ بالنظر إلى المخرجات تغير ترتيب الأزواج قيمة مفتاح (key-value). في الإصدار بايثون 5.3 وما قبله، كانت القواميس غير مرتبة. لكن ابتداءاً من بايثون 6.3، صارت القواميس

مرتبةً. بغض النظر عما إذا كان القاموس مرتباً أُم لا، ستظل الأزواج قيمة مفتاح كما هي، وهذا

سيمكنك من الوصول إلى البيانات بناء على ترابطاتها.

1. الوصول إلى عناصر قاموس

يمكننا الوصول إلى قيم محدَّدة في القاموس بالرجوع إلى المفاتيح المرتبطة بها ويمكن ايضاً الاستعانة ببعض التوابع الجاهزة للوصول إلى القيم أُو المفاتيح أُو كليهما.

أ. الوصول إلى عناصر القاموس باستخدام المفاتيح

إذا أُردنا الحصول على اسم المستخدم في Sammy، فيمكننا ذلك عن طريق استدعاء sammy['username']. هذا مثال على ذلك:

```
print(sammy['username'])  # sammy-shark
```

تتصرف القواميس مثل قواعد البيانات، فهي بدلًا من فهرسة العناصر بأعداد صحيحة، كما ه و الحال في **القوائم**، فإنها تفهرس العناصر (أو قيم القاموس) بمفاتيح، ويمكنك عبر تلك المفاتيح الحصول على القيم المقابلة لها.

باستدعاء المفتاح username، سنحصل على القيمة المرتبطة به، وهي sammy-shark.

وبالمِثل، يمكن استدعاء القيم الأخرى في القاموس sammy باستخدام نفس الصياغة:

```
sammy['followers']  # 987

sammy['online']   # True
```

ب. استخدام التوابع للوصول إلى العناصر

بالإضافة إلى استخدام المفاتيح للوصول إلى القيم، يمكننا أيضاً استخدام بعض التوابع المضمّنة، مثل:

- dict.keys(): الحصول على المفاتيح

الحصول على القيم• dict.values():

- dict.items(): الحصول على العناصر على هيئة قائمة من أزواج (key, value) لإعادة المفاتيح، نستخدم التابع dict.keys()، كما يوضح المثال التالي:

```
print(sammy.keys())  # dict_keys(['followers',
'username', 'online'])
```

تلقينا في المخرجات كائنَ عرض تكراري (iterable view object) من الصنف dict_keys يحوي المفاتيح ثم طبِعت المفاتيح على هيئة **قائمة**.

يمكن استخدام هذا التابع للاستعلام من القواميس. على سبيل المثال، يمكننا البحث عن المفاتيح المشتركة بين قاموسين:

```
sammy = {'username': 'sammy-shark', 'online': True,
'followers': 987} jesse = {'username': 'JOctopus', 'online': False,
'points':
723} for common_key in sammy.keys() & jesse.keys():

  print(sammy[common_key], jesse[common_key])
```

يحوي القاموسان sammy و jesse معلومات تعريف المستخدم. كما أنَّ لهما مفاتيح مختلفة، لأنَّ لدى Sammy ملف تعريف اجتماعي يضم مفتاحاً followers يمثل المتابعين على الشبكة الاجتماعية ،أما Jesse فلها ملف تعريف للألعاب يضم مفتاحاً points يمثل النقاط. كلا القاموسين يشتركان في المفتاحين username و online، ويمكن العثور عليهما عند تنفيذ هذا البرنامج:

```
sammy-shark  JOctopus  True
False
```

يمكننا بالتأكيد تحسين البرنامج لتسهيل قراءة المخرجات، ولكنَّ الغرض هنا هو توضيح إمكانية استخدام dict.keys() لرصد المفاتيح المشتركة بين عدَّة قواميس. هذا مفيد بشكل خاص عند العمل على القواميس الكبيرة.

و بالمثل، يمكننا استخدام التابع dict.values() للاستعلام عن القيم الموجودة في القاموس sammy على النحو التالي:

```
sammy = {'username': 'sammy-shark', 'online': True,
'followers': 987}

print(sammy.values())  #  dict_values([True, 'sammy-shark', )]789
```

يعيد كلا التابعين values() و keys() قوائم غير مرتبة تضم مفاتيح وقيم القاموسsammy على هيئة كائِن يعرض من الصنف dict_values و dict_keys على التوالي.

إن أُردت الحصول على الأزواج الموجودة في القاموس، فاستخدم التابع items():

```
print(sammy.items())
```

المخرجات ستكون:

```
dict_items([('online', True), ('username', 'sammy-shark'),
('followers', 987)])
```

س تكون النتيجة المعادة على هيئة قائمة مكونة من أُزواج (key, value) من dict_items.الصنف

يمكننا التكرار (iterate)على القائمة المعادة باستخدام الحلقة for. على سبيل المثال، يمكننا طباعة جميع مفاتيح وقيم القاموس المحدد، ثم جعلها أُكثر مقروئية عبر إضافة سلسلة نصية توضيحية:

```
for key, value in sammy.items():
print(key, 'is the key for the value', value)
```

وسينتج لنا:

```
online is the key for the value True followers is the
key for the value 987 username is the key for the
value sammy-shark
```

كررت الحلقة for على العناصر الموجودة في القاموس sammy، وطبعت المفاتيح والقيم سطراً سطراً، مع إضافة معلومات توضيحية.

2. تعديل القواميس

القواميس هي هياكل بيانات قابلة للتغيير (mutable)، أي يمكن تعديلها. في هذا القسم، سنتعلم كيفية إضافة عناصر إلى قاموس، وكيفية حذفها.

أ. إضافة وتغيير عناصر القاموس

يمكنك إضافة أزواج قيمة مفتاح إلى قاموس دون استخدام توابع أو دوال باستخدام الصياغة التالية:

```
dict[key] = value
```

في المثال التالي، سنضيف زوجاً مفتاح قيمة إلى قاموس يسمى usernames:

```
usernames = {'Sammy': 'sammy-shark', 'Jamie': 'mantisshrimp54'}

usernames['Drew'] = 'squidly'

print(usernames)      # {'Drew': 'squidly', 'Sammy': 'sammyshark',
'Jamie': 'mantisshrimp54'}
```

لاحظ أنَّ القاموس قد تم تحديثه بالزوج 'Drew': 'squidly'.

نظراً لأنَّ القواميس غير مرتبة، فيمكن أن يظهر الزوج المضاف في أ ي مكان في مخرجات القاموس. إذا استخدمنا القاموس usernames لاحقاً، فسيظهر فيه الزوج المضاف حديثًا.

يمكن استخدام هذه الصياغة لتعديل القيمة المرتبطة بمفتاح معيَّن. في هذه الحالة، سنشير إلى مفتاح موجود سلفاً، ونمرر قيمة مختلفة إليه.

سنعرف في المثال التالي قاموساً باسم drew يمثل البيانات الخاصة بأحد المستخدمين على بعض الشبكات الاجتماعية. حصل هذا المستخدم على عدد من المتابعين الإضافيين اليوم، لذلك سنحدّث القيمة المرتبطة بالمفتاح followers ثم نستخدم التابع print() للتحقق من أنّ القاموس قد عدل.

```
# تعريف القاموس الأصلي
usernames = {'Sammy': 'sammy-shark', 'Jamie': 'mantisshrimp54'}
# while إعداد الحلقة التكرارية  while True:  # اطلب من المستخدم
إدخال اسم  print('Enter a name:')

تعيين
#المدخلات إلى المتغير
name ()  name =
input

# تحقق مما إذا كان الاسم موجودا في القاموس ثم اطبع الرد
if name in usernames: print(usernames[name] + ' is the username
of ' + name)

# إذا لم يكن الاسم في القاموس  :
else

#  اطبع الرد

print('I don\'t have ' + name + '\'s username, what is it?')

#  خذ اسم مستخدم جديد لربطه بذلك الاسم
username = input()

عين قيمة اسم المستخدم إلى المفتاح
```

```
        # name

    usernames[name] = username

                                    # اطبع رداً يبيّن أنَّ البيانات قد حدثت

( print)'Data updated.'
drew = {'username': 'squidly', 'online': True, 'followers':
305} drew['followers'] = 342
print(drew)
#  {'username': 'squidly', 'followers': 342, 'online': True}
```

في المخرجات نرى أنّ عدد المتابعين قد قفز من 305 إلى 342.

يمكننا استخدام هذه الطريقة لإضافة أزواج قيمة مفتاح إلى القواميس عبر مدخلات المستخدم. سنكتب برنامجاً سريعاً usernames.py، يعمل من سطر الأوامر ويسمح للمستخدم بإضافة الأسماء وأسماء المستخدمين المرتبطة بها:

سننفذ البرنامج من **سطر الأوامر**:

```
python usernames.py
```

عندما ننفذ البرنامج، سنحصل على مخرجات مشابهة لما يلي:

```
Enter a name: Sammy sammy-shark
is the username of Sammy
Enter a name:
Jesse I don't have Jesse's username, what
is it?
JOctopus
Data
updated.
Enter a name:
```

عند الانتهاء من اختبار البرنامج، اضغط على CTRL + C للخروج من البرنامج. يمكن كتخصيص حرف لإنهاء البرنامج (مثل الحرف q)، وجعل البرنامج ينصت له عبر التعليمات الشرطية.

يوضح هذا المثال كيف يمكنك تعديل القواميس بشكل تفاعلي. في هذا البرنامج، بمجرد خروجك باستخدام CTRL + C، ستفقد جميع بياناتك، إلا إن **خزّنَت البيانات في ملف**.

يمكننا ايضاً إضافة عناصر إلى القواميس وتعديلها باستخدام التابع dict.update(). هذا التابع مختلف عن التابع append() الذي يستخدم مع القوائم.

سنضيف المفتاح followers في القاموس jesse أُدناه، ونَمنحه قيمة عددية صحيحة بواسطة التابع jesse.update(). بعد ذلك، سنطبع القاموس المحدَّث.

```
jesse = {'username': 'JOctopus', 'online': False, 'points': 723}
jesse.update({'followers': 481})

print(jesse)  # {'followers': 481, 'username': 'JOctopus', 'points': 723, 'online': False}
```

نتبيَّن من المخرجات أُنَّنا نجحنا في إضافة الزوج followers: 481 إلى القاموس jesse.

يمكننا ايضاً استخدام التابع dict.update() لتعديل زوج قيمة مفتاح موجود سلفاً عن طريق استبدال قيمة مفتاح معيَّن.

سنغير القيمة المرتبطة بالمفتاح online في القاموس Sammy من True إلى False:

```
sammy = {'username': 'sammy-shark', 'online': True, 'followers': 987} sammy.update({'online': False})
print(sammy)  # {'username': 'sammy-shark', 'followers': 987, 'online': False}
```

يغيّر السطر sammy.update({'online': False}) القيمة المرتبطة بالمفتاح 'online' من True إلى False. عند استدعاء التابع print() على القاموس، يمكنك أُن ترى في المخرجات أُنَّ التحديث قد تمّ.

لإضافة عناصر إلى القواميس أُو تعديل القيم، يمكن مّا استخدام الصياغة

dict[key] = value ، أو التابع dict.update().

3. حذف عناصر من القاموس

كما يمكنك إضافة أزواج قيمة مفتاح إلى القاموس ،أو تغيير قيمه، يمكنك أيضاً حذف العناصر الموجودة في القاموس.

لتزيل زوج - قيمة مفتاح من القاموس، استخدم الصياغة التالية:

```
del dict[key]
```

لنأخذ القاموس jesse الذي يمثل أحد المستخدمين، ولنفترض أنَّ jesse لم تعد تستخدم المنصة لأجل ممارسة الألعاب، ل ذلك سنزيل العنصر المرتبط بالمفتاح points. بعد ذلك، سنطبع القاموس لتأكيد حذف العنصر:

```
jesse = {'username': 'JOctopus', 'online': False, 'points': 723, 'followers': 481} del jesse['points']
print(jesse)
# {'online': False, 'username': 'JOctopus', 'followers': 481}
```

يزيل السطر del jesse ['points'] الزوج 'points': 723 من القاموس jesse.إذا أُردت محو جميع عناصر القاموس، فيمكنك ذلك باستخدام التابع dict.clear().

سيَبقى هذا القاموس في الذاكرة، وهذا مفيد في حال احتجنا إلى استخدامه لاحقاً في البرنامج، بيْد أنه سيفرغ جميع العناصر من القاموس.

دعنا نزيل كل عناصر القاموس jesse:

```
jesse = {'username': 'JOctopus', 'online': False, 'points':
723, 'followers': 481}
jesse.clear() print(jesse)
# {}
```

تظهِر المخرجات أنَّ القاموس صار فارغاً الآن.

إذا لم تعد بحاجة إلى القاموس، فاستخدم del للتخلص منه بالكامل:

```
del jesse

print(jesse)
```

إذا نفّذت الأمر print() بعد حذف القاموس jesse، سوف تتلقى الخطأ التالي:

```
NameError: name 'jesse' is not defined
```

4. خلاصة الفصل

أُلقينا في هذا الفصل نظرة على النوع dictionary (القواميس) في بايثون. تتألف القواميس (dictionaries) من أُزواج قيمة مفتاح، وتوفر حلاً ممتازاً لتخزين البيانات دون الحاجة إلى فهرستها. يتيح لنا ذلك استرداد القيم بناءً على معانيها وعلاقتها بأنواع البيانات الأخرى.

إن لم تطلع على فصل **فهم أُنواع البيانات**، فيمكنك الرجوع إليه للتعرف على أُنواع البيانات الأخرى الموجودة في بايثون.