3ème Année

Cours 04Bionformatique Ensembles et dictionnaires

Dr. Mohammed Mehdi Bouchene

Ensembles

- Les ensembles contiennent des éléments uniques, c'est-à-dire qu'aucune répétition n'est autorisée.
- Les éléments d'un ensemble n'ont pas d'ordre
- Les ensembles ne peuvent pas contenir d'éléments pouvant être modifiés en interne (par exemple, des listes et des dictionnaires)

```
l = [1, 2, 3, 2, 3] # list of 5 values
s = set(l) # set of 3 unique values
print(s)
e = set() # empty set
print(e)
```

Les ensembles sont très similaires aux listes et aux tuples et vous pouvez utiliser plusieurs des mêmes opérateurs et fonctions, à l'exception du fait qu'ils ne sont pas ordonnés de manière inhérente. Ils ne possèdent donc pas d'index et ne peuvent contenir que des valeurs uniques. Vous devez donc ajouter une valeur déjà dans l'ensemble n'aura aucun effet.

```
s = set([1, 2, 3, 2, 3])
print(s)
print("number in set:", len(s))
s.add(4)
print(s)
s.add(3)
```

Vous pouvez supprimer des éléments spécifiques de l'ensemble.

```
s = set([1, 2, 3, 2, 3])
print(s)
s.remove(3)
print(s)
```

Vous pouvez effectuer toutes les opérations logiques attendues sur les ensembles, telles que l'union ou l'intersection de 2 ensembles avec les opérateurs **|** et **&** respectivement.

Exercice 1

Compte tenu de la séquence protéique "MPISEPTFFEIF", divisez la séquence en ses codes d'acides aminés constitutifs et utilisez un ensemble pour établir les acides aminés uniques de la protéine et afficher le résultat.

Dictionnaires

Les listes sont utiles dans de nombreux contextes, mais nous avons souvent des données sans ordre inhérent auquel nous voulons accéder par un nom utile plutôt que par un index. Par exemple, à la suite de certaines expériences, nous pouvons avoir un ensemble de gènes et des valeurs d'expression correspondantes. Nous pourrions mettre les valeurs d'expression dans une liste, mais il faudrait ensuite savoir quel index de la liste correspond à quel gène et cela se compliquerait rapidement.

Pour ces situations, un dictionnaire est une structure de données très utile.

Propriétés

- Contient un mappage de clés en valeurs (comme un mot et sa définition correspondante dans un dictionnaire)
- Les clés d'un dictionnaire sont uniques, c'est-à-dire qu'elles ne peuvent pas être répétées.
- Les valeurs d'un dictionnaire peuvent être de n'importe quel type de données
- Les clés d'un dictionnaire ne peuvent pas être un type modifiable en interne (par exemple, des listes, mais vous pouvez utiliser des Tuplets)
- Les dictionnaires ne stockent pas de données dans un ordre particulier

```
dna = {"A": "Adenine", "C": "Cytosine", "G": "Guanine", "T": "Thymine"}

print(dna)
```

Vous pouvez accéder aux valeurs d'un dictionnaire en utilisant la touche entre crochets.

```
dna = {"A": "Adenine", "C": "Cytosine", "G": "Guanine", "T": "Thymine"}
print("A represents", dna["A"])
```

Une erreur est déclenchée si une clé est absente du dictionnaire

```
dna = {"A": "Adenine", "C": "Cytosine", "G": "Guanine", "T": "Thymine"}
print("What about N?", dna["N"])
```

Vous pouvez accéder aux valeurs en toute sécurité avec la méthode **get**, qui ne renvoie Aucune si la clé est absente et vous pouvez également fournir des valeurs par défaut.

```
dna = {"A": "Adenine", "C": "Cytosine", "G": "Guanine", "T": "Thymine"}
print("What about N?", dna.get("N"))
print("With a default value:", dna.get("N", "unknown"))
```

Vous pouvez vérifier si une clé est dans un dictionnaire avec l'opérateur **in** et vous pouvez le nier avec

```
dna = {"A": "Adenine", "C": "Cytosine", "G": "Guanine", "T": "Thymine"}
"T" in dna
```

```
dna = {"A": "Adenine", "C": "Cytosine", "G": "Guanine", "T": "Thymine"}
"Y" not in dna
```

La fonction **len** () renvoie le nombre de paires (clé, valeur) dans le dictionnaire :

```
dna = {"A": "Adenine", "C": "Cytosine", "G": "Guanine", "T": "Thymine"}
print(len(dna))
```

Vous pouvez introduire de nouvelles entrées dans le dictionnaire en attribuant une valeur avec une nouvelle

```
dna = {"A": "Adenine", "C": "Cytosine", "G": "Guanine", "T": "Thymine"}
dna['Y'] = 'Pyrimidine'
print(dna)
```

Vous pouvez changer la valeur d'une clé existante en la réaffectant:

```
dna = {'A': 'Adenine', 'C': 'Cytosine', 'T': 'Thymine', 'G': 'Guanine', 'Y': 'Pyrimidine'}
dna['Y'] = 'Cytosine or Thymine'
print(dna)
```

Vous pouvez supprimer des entrées du dictionnaire avec la fonction **del**.

```
dna = {'A': 'Adenine', 'C': 'Cytosine', 'T': 'Thymine', 'G': 'Guanine', 'Y': 'Pyrimidine'}
del dna['Y']
print(dna)
```

Vous pouvez obtenir une liste de toutes les clés (dans un ordre arbitraire) à l'aide de la fonction **.keys ()**

```
dna = {'A': 'Adenine', 'C': 'Cytosine', 'T': 'Thymine', 'G': 'Guanine', 'Y': 'Pyrimidine'}
print(list(dna.keys()))
```

Et obtenir de manière équivalente une liste des valeurs à l'aide de la fonction .**values ()**

```
dna = {'A': 'Adenine', 'C': 'Cytosine', 'T': 'Thymine', 'G': 'Guanine', 'Y': 'Pyrimidine'}
print(list(dna.values()))
```

Et une liste de tuples contenant des paires (clé, valeur) à l'aide de la fonction **.items()**

```
dna = {'A': 'Adenine', 'C': 'Cytosine', 'T': 'Thymine', 'G': 'Guanine', 'Y': 'Pyrimidine'}
print(list(dna.items()))
```

Exercices 2

1. Afficher les noms des acides aminés qui seraient produits par la séquence d'ADN "GTT GCA CCA CAA CCG" (voir le tableau des codons d'ADN). Divisez cette chaîne en codons individuels, puis utilisez un dictionnaire pour mapper les séquences de codons avec les acides aminés qu'ils codent.

2. Afficher chaque codon et son acide aminé correspondant.

3. Pourquoi ne pourrions-nous pas construire un dictionnaire où les clés sont des noms d'acides aminés et les valeurs sont les codons ADN?

1st base	2nd base								3rd base
	T		C		A		G		
T	TTT	(Phe/F) Phenylalanine	TCT	(Ser/S) Serine	TAT	(Tyr/Y) Tyrosine	TGT	(Cys/C) Cysteine	**T**
	TTC		TCC		TAC		TGC		**C**
	TTA	(Leu/L) Leucine	TCA		TAA	Stop (*Ochre*) [B]	TGA	Stop (*Opal*) [B]	**A**
	TTG		TCG		TAG	Stop (*Amber*) [B]	TGG	(Trp/W) Tryptophan	**G**
C	CTT		CCT	(Pro/P) Proline	CAT	(His/H) Histidine	CGT	(Arg/R) Arginine	**T**
	CTC		CCC		CAC		CGC		**C**
	CTA		CCA		CAA	(Gln/Q) Glutamine	CGA		**A**
	CTG		CCG		CAG		CGG		**G**
A	ATT	(Ile/I) Isoleucine	ACT	(Thr/T) Threonine	AAT	(Asn/N) Asparagine	AGT	(Ser/S) Serine	**T**
	ATC		ACC		AAC		AGC		**C**
	ATA		ACA		AAA	(Lys/K) Lysine	AGA	(Arg/R) Arginine	**A**
	ATG[A]	(Met/M) Methionine	ACG		AAG		AGG		**G**
G	GTT	(Val/V) Valine	GCT	(Ala/A) Alanine	GAT	(Asp/D) Aspartic acid	GGT	(Gly/G) Glycine	**T**
	GTC		GCC		GAC		GGC		**C**
	GTA		GCA		GAA	(Glu/E) Glutamic acid	GGA		**A**
	GTG		GCG		GAG		GGG		**G**

Exercice avancé (à faire à domicile)

En commençant par un dictionnaire vide, comptez l'abondance des différents types de résidus présents dans la séquence de la protéine lysozyme à une lettre

MKALIVLGLVLLSVTVQGKVFERCELARTLKRLGMDGYRGISLANWMCLAKWESGYNTRATNYNAGDRSTDYGIFQINSRY WCNDGKTPGAVNACHLSCSALLQDNIADAVACAKRVVRDPQGIRAWVAWRNRCQNRDVRQYVQGCGV

et afficher les résultats à l'écran par ordre alphabétique. .