TD 3 - Dictionnaires

Ferdinand Mom

May 2019

Figure 1: Meme de qualité

1 Classiques

Exercice 1.1: addDic

Créez la fonction **addDic(dic, elt)** qui retourne un dictionnaire avec *elt* ajouté.

```
res = addDic({1: "Salut"}, ("", ""))
print(res)
>>> {1: "Salut", "": ""}

res = addDic({1: 42}, (2, 323))
print(res)
>>> {1: 42, 2: 323}
```

Exercice 1.2: removeDic

Créez la fonction **removeDic(dic, key)** qui retourne un dictionnaire sans *key*. N'utilisez pas les méthodes *popitem()* et *del*.

```
res = removeDic({1: "Salut", "Ferdi": "Bye"}, "Ferdi")
print(res)
>>> {1: "Salut"}
```

Exercice 1.3: concatDic

Créez la fonction **concatDic(dic1, dic2)** qui retourne la concaténation de *dic1* et *dic2* sans utiliser la méthode *update()*.

```
res = concatDic({1: "Salut"}, {"Ferdi": "Bye"})
print(res)
>>> {1: "Salut", "Ferdi": "Bye"}

res = concatDic({1: "Salut"}, {1: "Bye"})
print(res)
>>> {1: "Bye"}
```

Exercice 1.4: isInDic

Créez la fonction **isInDic(dic, key)** qui retourne *True* si *key* est dans *dic*. N'utilisez pas la méthode *get()*.

```
res = isInDic({1: "Salut", "Ferdi": "Bye"}, "Ferdi")
print(res)
>>> True

res = isInDic({1: "Salut", "Ferdi": "Bye"}, "")
print(res)
>>> False
```

Exercice 1.5: dispDic

Créez la fonction **dispDic(dic)** qui retourne une string avec le contenu du dictionnaire comme suivant.

```
res = dispDic({1: "Salut", 2: "Bye", 3: "Ferdi"})
print(res)
>>>
"1 : Salut
2 : Bye
3 : Ferdi"
```

Exercice 1.6: multDic

Créez la fonction **multDic(dic, x)** qui retourne un dictionnaire avec les valeurs de *dic* multiplié par x.

```
res = multDic({1: 2, 2: 42, 3: 23}, 10)
print(res)
>>>  {1: 20, 2: 420, 3: 230}

res = multDic({1: 2, 2: 42, 3: 23}, 0)
print(res)
>>>  {1: 0, 2: 0, 3: 0}
```

Exercice 1.7: buildFrequencyDic

Créez la fonction **buildFrequencyDic(L)** qui retourne un dictionnaire avec les fréquences des éléments de L.

```
res = buildFrequencyDic([])
print(res)
>>>  {}

res = buildFrequencyDic([1, 2, 3, 3, 4, 5, 6, 6, 6])
print(res)
>>>  {1: 1, 2: 1, 3: 2, 4: 1, 5: 1, 6: 3}
```

Exercice 1.8: repeteTimeDic

Créez la fonction **repeteTimeDic(dic, k)** qui retourne un dictionnaire avec seulement les éléments répétés k fois dans *dic*.

```
res = repeteTimeDic({}, 1)
print(res)
>>>  {}

res = repeteTimeDic({1: 1, 2: 1, 3: 2, 4: 1, 5: 1, 6: 3}, 1)
print(res)
>>>  {1: 1, 2: 1, 4: 1, 5: 1}
```

Exercice 1.9: frequencyLetterDic

Créez la fonction **frequencyLetterDic(s)** qui retourne un dictionnaire des fréquences des lettres dans *s*.

```
res = frequencyLetterDic("")
print(res)
>>> {}

res = frequencyLetterDic("saaluttt")
print(res)
>>> {'s': 1, 'a': 2, 'l': 1, 'u': 1, 't': 3}
```

Exercice 1.10: anagram

Créez la fonction **anagram(s1, s2)** qui retourne *True* si *s1* et *s2* sont des anagrammes. Réutilisez la fonction de l'exercice précédente pour vous aider.

```
res = anagram("salut", "a")
print(res)
>>> False

res = anagram("salut", "saalut")
print(res)
>>> False

res = anagram("salut", "tuasl")
print(res)
>>> True
```

True knowledge exists in knowing that you know nothing.

- Socrates