Python
Listes, Tuples, Dictionnaires

© Domingo Palao Muñoz

Tuples

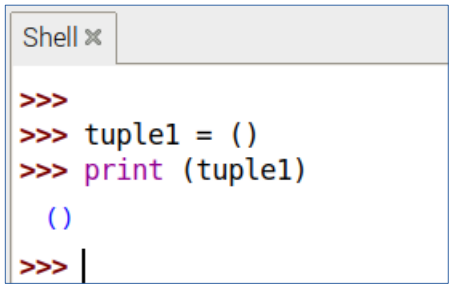
Les Tuples

- Lorsque vous travaillez avec des variables, il est parfois pratique de regrouper les valeurs des données afin de pouvoir les parcourir plus tard dans votre script. Vous ne pouvez pas facilement faire cela avec des variables séparées, mais Python fournit une solution pour vous.
- Un tuple est un type de données en Python qui vous permet de faire cela.
- Les valeurs d'un tuple son **immuables**, c'est à dire, elles ne changent pas.

© Domingo Palao Muñoz

Les Tuples

- Une fois que vous avez créé un tuple, vous pouvez travailler avec le tuple comme une seule valeur individuelle dans votre code Python.
- Création d'un tuple vide :



© Domingo Palao Muñoz

Les Tuples

- Un tuple vs une variable

```
Shell
>>> tuple2 = 1
>>> print (tuple2)
  1
>>> tuple2 = 1,
>>> print (tuple2)
  (1,)
```

- Un tuple avec plusieurs valeurs :

```
Shell
>>> tuple3 = 1, 2, 3, 4
>>> print (tuple3)
  (1, 2, 3, 4)
```


© Domingo Palao Muñoz

Les Tuples

- Un tuple composé de chaînes de caractères :

```
Shell
>>> tuple5 = "Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi"
>>> print (tuple5)
  ('Dimanche', 'Lundi', 'Mardi', 'Mercredi', 'Jeudi', 'Vendredi', 'Samedi')
```


© Domingo Palao Muñoz

Les Tuples

- Pour accéder à un élément précise de notre tuple:

```
Shell
>>> tuple5 = "Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi"
>>> print (tuple5)
  ('Dimanche', 'Lundi', 'Mardi', 'Mercredi', 'Jeudi', 'Vendredi', 'Samedi')
>>> print (tuple5[0])
  Dimanche
>>> print (tuple5[7])
  Traceback (most recent call last):
    File "<pyshell>", line 1, in <module>
  IndexError: tuple index out of range
```


© Domingo Palao Muñoz

Les Tuples

- Pour accéder à un range de valeurs on peut utiliser :

```
Shell
>>> tuple5 = "Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi"
>>> tuple6 = tuple5[1:5]
>>> print (tuple6)
  ('Lundi', 'Mardi', 'Mercredi', 'Jeudi')
```

- ATTENTION! Le premier index est le point de départ, mais le deuxième est l'élément juste avant la valeur indiquée, on peut dire que pour l'expression : tuple[i:j] nous allons prendre les valeurs de x selon : i <=x < j

Les Tuples

- Une première utilisation d'un tuple :

jourssemaine.py

```
1  semainelaborale = "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi"
2  if "Samedi" in semainelaborale:
3      print ("Nous allons travailler aujord'hui")
4  else:
5      print ("Journée de congé")
```

Shell

```
Python 3.7.3 (/usr/bin/python3)
>>> %cd /home/pi/Documents/Scripts
>>> %Run jourssemaine.py

  Journée de congé
```


© Domingo Palao Muñoz

Les Tuples

- Utilisation d’un tuple :

```
Shell
>>> min (semainelaborale)
'Jeudi'
>>> max (semainelaborale)
'Vendredi'
```


© Domingo Palao Muñoz

Les Tuples

- Concaténation de tuples :

```
Shell
>>> impairs = 1, 3, 5, 7, 9
>>> pairs = 2, 4, 6, 8
>>> tous = pairs + impairs
>>> print (tous)
  (2, 4, 6, 8, 1, 3, 5, 7, 9)
```

Listes

© Domingo Palao Muñoz

Les Listes

- Les listes se ressemblent aux tuples.
- Une liste peut stocker plusieurs valeurs référencées par une seule variable de type liste.
- La différence est que la liste est **mutable**, c'est à dire, elle peut changer!
- Vous pouvez modifier, ajouter ou enlever des données dans une liste.
- Ceci ajoute plus de versatilité aux scripts.

Les Listes

- Création d'une liste vide :

```
Shell
>>> liste1 = []
>>> print (liste1)
  []
```

- Pour initialiser une lista on utilise les parenthèses carrés et la comma comme séparateur entre les valeurs:

```
Shell
>>> liste2 = [1, 2, 3, 4]
>>> print (liste2)
  [1, 2, 3, 4]
```


© Domingo Palao Muñoz

Les Listes

- Il est possible de convertir un tuple en liste à l'aide de la fonction list( ):

```
Shell
>>> tuple1 = 1, 2, 3, 4
>>> liste3 = list(tuple1)
>>> print (liste3)
  [1, 2, 3, 4]
```

- Pour accéder à un élément de la liste :

```
Shell
>>> liste_semaine = ["Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi"]
>>> print (liste_semaine[0])
  Dimanche
>>> print (liste_semaine[7])
  Traceback (most recent call last):
    File "<pyshell>", line 1, in <module>
  IndexError: list index out of range
```

Les Listes

- Pour accéder à un élément de la liste :

```
Shell
>>> liste_semaine = ["Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi"]
>>> print (liste_semaine[-2])
  Vendredi
```

- Pour accéder à un range de la liste :

```
Shell
>>> liste_semaine = ["Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi"]
>>> print (liste_semaine[0:3])
  ['Dimanche', 'Lundi', 'Mardi']
```

Les Listes

- Remplacer des valeurs dans une liste :

```
Shell
>>> liste1 = [1, 2, 3, 4]
>>> liste1[1] = 10
>>> print (liste1)
  [1, 10, 3, 4]
```

- Remplacer des valeurs avec un tuple :

```
Shell
>>> liste1 = [1, 2, 3, 4]
>>> tuple1 = 10, 11
>>> liste1[1:3] = tuple1
>>> print (liste1)
  [1, 10, 11, 4]
```


© Domingo Palao Muñoz

Les Listes

- Suppression des valeurs de la liste :

```
Shell
>>> liste1 = [1, 2, 3, 4]
>>> del liste1[1]
>>> print (liste1)
  [1, 3, 4]
```

- La fonction pop( ) :

```
Shell
>>> liste1 = [1, 2, 3, 4]
>>> liste1.pop(2)
3
>>> print (liste1)
  [1, 2, 4]
```


© Domingo Palao Muñoz

Les Listes

- Ajout d'éléments à la fin d'une liste :

```
Shell
>>> liste1 = [1, 2, 3, 4]
>>> liste1.append (5)
>>> print (liste1)
  [1, 2, 3, 4, 5]
```

- Insertion d'éléments à une position quelconque :

```
Shell
>>> liste1 = [1, 2, 3, 4]
>>> liste1.insert (1, 5)
>>> print (liste1)
  [1, 5, 2, 3, 4]
```

Les Listes

- Concaténation de listes :

```
Shell
>>> liste1 = [1, 2, 3, 4]
>>> liste2 = [5, 6, 7, 8]
>>> liste1.append(liste2)
>>> print (liste1)
  [1, 2, 3, 4, [5, 6, 7, 8]]
```

- Il faut faire attention avec la manière de ajouter de la fonction append(). Elle fait l'ajout comme un seul valeur à la liste.
- Pour faire la concaténation de deux listes il faut utiliser la fonction extend()

```
Shell
>>> liste1 = [1, 2, 3, 4]
>>> liste2 = [5, 6, 7, 8]
>>> liste1.extend (liste2)
>>> print (liste1)
  [1, 2, 3, 4, 5, 6, 7, 8]
```


© Domingo Palao Muñoz

Les Listes

- Combien de fois une valeur est dans la liste?

```
Shell
>>> liste1 = [1, 2, 3, 4, 1, 2, 3]
>>> liste1.count(1)

2
```

- On peut trier la liste :

```
Shell
>>> liste1 = [1, 2, 3, 4, 1, 2, 3]
>>> liste1.sort()
>>> print (liste1)

  [1, 1, 2, 2, 3, 3, 4]
```


© Domingo Palao Muñoz

Les Listes

- Listes à plusieurs dimensions :

```
Shell
>>> liste=[[1,2,3],[4,5,6],[7,8,9]]
>>> print (liste)
  [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

- Pour accéder aux éléments individuellement

```
Shell
>>>
>>> liste=[[1,2,3],[4,5,6],[7,8,9]]
>>> print (liste)
  [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> print (liste[0][0])
  1
>>> print (liste[1][2])
  6
>>> print (liste[2][1])
  8
```


© Domingo Palao Muñoz

Les Listes

- Il est possible de parcourir tous les éléments d'une liste :

Shell

```
>>> liste_semaine = ["Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi"]
>>> for jour in liste_semaine:
        print ("Le jour de la semaine est:", jour)


Le jour de la semaine est: Dimanche
Le jour de la semaine est: Lundi
Le jour de la semaine est: Mardi
Le jour de la semaine est: Mercredi
Le jour de la semaine est: Jeudi
Le jour de la semaine est: Vendredi
Le jour de la semaine est: Samedi
```


© Domingo Palao Muñoz

Les Listes

- Il est possible d'inverser la liste :

Shell

```
>>>
>>> liste_semaine = ["Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi"]
>>> liste_inverse = reversed(liste_semaine)
>>>
>>> for jour in liste_inverse:
        print ("Le jour de la semaine est:", jour)


  Le jour de la semaine est: Samedi
  Le jour de la semaine est: Vendredi
  Le jour de la semaine est: Jeudi
  Le jour de la semaine est: Mercredi
  Le jour de la semaine est: Mardi
  Le jour de la semaine est: Lundi
  Le jour de la semaine est: Dimanche
>>>
```

Dictionnaires

Les Dictionnaires

- Un dictionnaire est une structure simple souvent nommé array associatif.
- Les données contenus dans un dictionnaire sont composés de deux parties :
  - La valeur
  - La clé
- La clé est immuable et est associé à la valeur.
- Chaque élément du dictionnaire est un pair clé/valeur.

© Domingo Palao Muñoz

Les Dictionnaires

- Création d'un dictionnaire vide :

```
Shell
>>> etudiants={}
>>> type (etudiants)
<class 'dict'>
>>> print (etudiants)
 {}
```

- Pour ajouter des éléments au dictionnaire :

```
Shell
>>> etudiants={102345:"Paul",102348:"Marie",102350:"Anne"}
>>> etudiants
{102345: 'Paul', 102348: 'Marie', 102350: 'Anne'}
```

- Les dictionnaires ne sont pas ordonés!

© Domingo Palao Muñoz

Les Dictionnaires

- Pour ajouter un élément au dictionnaire :

```
Shell
>>> etudiants[102360]="Richard"
>>> etudiants
{102345: 'Paul', 102348: 'Marie', 102350: 'Anne', 102360: 'Richard'}
```


© Domingo Palao Muñoz

Les Dictionnaires

- Quelques points importants à considérer sur la clé:
  - La clé ne peut pas être une liste.
  - La clé doit être un objet immuable.
  - La clé peut être une string.
  - La clé peut être un nombre (integer ou float)
  - La clé peut être un tuple.
  - La clé doit être associé à une seule valeur (il n'y a pas de clés dupliquées)
- Par contre, une valeur peut être pratiquement n'importe quel objet.

Les Dictionnaires

- Pour accéder à un élément du dictionnaire :

```
Shell
>>>
>>> etudiants
{102345: 'Paul', 102348: 'Marie', 102350: 'Anne', 102360: 'Richard'}
>>> etudiants [102345]
'Paul'
```

- L'opération get()

```
Shell
>>>
>>> etudiants
{102345: 'Paul', 102348: 'Marie', 102350: 'Anne', 102360: 'Richard'}
>>> etudiants.get(102348,'Not Found')
'Marie'
>>> etudiants.get(102349,'Not Found')
'Not Found'
```


© Domingo Palao Muñoz

Les Dictionnaires

- Pour traverser un dictionnaire :

Shell

```
>>>
>>>
>>> etudiants
{102345: 'Paul', 102348: 'Marie', 102350: 'Anne', 102360: 'Richard'}
>>> liste_cle = etudiants.keys()
>>> liste_cle
dict_keys([102345, 102348, 102350, 102360])
>>> for la_cle in liste_cle:
        print (la_cle, end=' ')
        etudiants[la_cle]


 102345 'Paul'
 102348 'Marie'
 102350 'Anne'
 102360 'Richard'
```


© Domingo Palao Muñoz

Les Dictionnaires

- Pour mettre à jour un dictionnaire :

Shell

```
>>> etudiants
{102345: 'Paul', 102348: 'Marie', 102350: 'Anne', 102360: 'Richard'}
>>> etudiants[102345] = 'Jean Paul'
>>> etudiants
{102345: 'Jean Paul', 102348: 'Marie', 102350: 'Anne', 102360: 'Richard'}
```

Les Dictionnaires

- Pour supprimer un élément d'un dictionnaire :

Shell

```
>>>
>>> etudiants
{102345: 'Jean Paul', 102348: 'Marie', 102350: 'Anne', 102360: 'Richard'}
>>> del etudiants[102346]
  Traceback (most recent call last):
    File "<pyshell>", line 1, in <module>
  KeyError: 102346
>>> if 102346 in etudiants:
        del etudiants[102346]

>>> etudiants
{102345: 'Jean Paul', 102348: 'Marie', 102350: 'Anne', 102360: 'Richard'}
>>> if 102345 in etudiants:
        del etudiants[102345]

>>> etudiants
{102348: 'Marie', 102350: 'Anne', 102360: 'Richard'}
```


© Domingo Palao Muñoz

Les Dictionnaires

- Exemple d'utilisation d'un dictionnaire:

testDict.py

```
# testDict.py- Dictionnaire pour les temperatures de la semaine
# Author: Domingo Palao
# Date: 29 janvier 2021
#############################################################
#
print()
print("Entrez la temperature de la semaine:")
#
temp_semaine={}        #Creation d'un dictionnaire vide
#
jours_semaine=('Dimanche','Lundi','Mardi','Mercredi','Jeudi','Vendredi','Samedi')

for jour in jours_semaine:  #Boucle pour les jours
#
    # Demandez l'utilisateur la temperature du jour
    prompt="Entrez la temperature de " + jour + ": "
    temp_jour=int(input(prompt))

    # Garder l'élément dans le dictionnaire
    temp_semaine[jour]=temp_jour
#
#############################################################
# Afficher les temperatures de la semaine
#
print()
print("Les temperatures de la semaine")
#
cles_semaine=temp_semaine.keys()   #Chercher les clés du dictionnaire
#
for le_jour in cles_semaine:         #Boucle pour afficher clé/valeur
    print("La temperature le ", le_jour, "est", end = ': ')
    print(temp_semaine[le_jour])
#
#############################################################
```

Shell

```
>>> %Run testDict.py

  Entrez la temperature de la semaine:
  Entrez la temperature de Dimanche: 10
  Entrez la temperature de Lundi: -5
  Entrez la temperature de Mardi: 0
  Entrez la temperature de Mercredi: 3
  Entrez la temperature de Jeudi: 12
  Entrez la temperature de Vendredi: -5
  Entrez la temperature de Samedi: 2

  Les temperatures de la semaine
  La temperature le  Dimanche est: 10
  La temperature le  Lundi est: -5
  La temperature le  Mardi est: 0
  La temperature le  Mercredi est: 3
  La temperature le  Jeudi est: 12
  La temperature le  Vendredi est: -5
  La temperature le  Samedi est: 2
```

Ensembles

© Domingo Palao Muñoz

Les Ensembles

- Un ensemble (set) est une collection d'éléments. À différence d'un dictionnaire, un ensemble garde seulement les valeurs, il n'y a pas de clés.
- Les éléments dans un ensemble ne sont pas ordonnés.
- Chaque élément est unique.

Les Ensembles

- Création d'un ensemble vide :

```
Shell
>>> ensemble_etudiants = set()
>>> type (ensemble_etudiants)
<class 'set'>
>>> ensemble_etudiants
set()
```

- Ajouter des éléments à un ensemble :

```
Shell
>>> ensemble_etudiants = set()
>>> ensemble_etudiants.add ('Paul')
>>> ensemble_etudiants.add ('Marie')
>>> ensemble_etudiants.add ('Anne')
>>> ensemble_etudiants.add ('Richard')
>>> ensemble_etudiants
{'Marie', 'Richard', 'Anne', 'Paul'}
```


© Domingo Palao Muñoz

Les Ensembles

- Opérations avec les ensembles :

Shell

```
>>>
>>> groupe1=set()
>>> groupe2=set()
>>> groupe1.add('Marie')
>>> groupe1.add('Paul')
>>> groupe1.add('Richard')
>>> groupe1.add('Anne')
>>> groupe2.add('Marie')
>>> groupe2.add('Bernard')
>>> groupe2.add('Sylvain')
>>> groupe_union = groupe1.union(groupe2)
>>> groupe_union
{'Marie', 'Bernard', 'Richard', 'Sylvain', 'Paul', 'Anne'}
>>> groupe_intersection = groupe1.intersection(groupe2)
>>> groupe_intersection
{'Marie'}
>>> groupe_difference = groupe1.difference(groupe2)
>>> groupe_difference
{'Anne', 'Paul', 'Richard'}
```


© Domingo Palao Muñoz

Questions?