Chp 0

Révision

Programme Python

Données

.py

Traitements de données

Résultats

Erreurs

1 Données saisies à partir du clavier


```
nom = input("votre nom : ")
type(nom)

n = input("Saisir un nombre positif : ")
type(n)
```

La fonction **input**(message)

- Permet de lire les données saisies au clavier
- Retourne toujours une chaine de caractères : str

1 Données saisies à partir du clavier


```
>>> n = input("Saisir un nombre positif : ")
Saisir un nombre positif : 4
type(n) ➔ str
>>> n + 1
TypeError: Can't convert 'int' object to str implicitly
```

Les nombres saisis au clavier (de type str) nécessitent une conversion vers le type désiré avant leur utilisation.

```
>>> n = int(input("Saisir un nombre entier positif : "))

>>> f = float(input("Saisir un nombre réel positif : "))
```

1 Données saisies à partir du clavier


```
>>> n = int(input("Saisir un nombre entier positif : "))
Saisir un nombre entier positif : 4*
```

➔ Python génère une exception de type “ValueError”

Toute conversion de type doit être entourée du bloc **try :… except :…**

```
try:
        n = int(input("Saisir un nombre entier positif : "))
except ValueError:
        print(" Valeur invalide !")
Saisir un nombre entier positif : 4*
Valeur invalide !
```

➔L’erreur de saisie n’interrompe pas l’exécution du programme.

1 Données saisies à partir du clavier

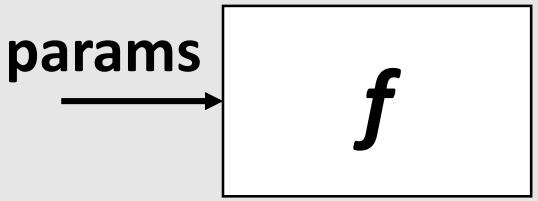

```
while 1:
      try:
            n = int(input("Saisir un nombre entier positif : "))
            break
      except ValueError:
            print(" Valeur invalide !")
```

```
Saisir un nombre entier positif : 4*   ➔ Valeur invalide !
Saisir un nombre entier positif : 4
```

➔Le programme boucle jusqu'à ce que la conversion passe correctement.

2 Donnés récupérées à partir des paramètres d'une fonction


```
def f(annee, nom):
    …
```



Bonne pratique !

Vérifiez toujours la validité des paramètres via la commande python **assert**

Exemple :

```
def f(annee, nom):
    assert type(annee) == int, " Année invalide. "
    assert type(nom) == str and nom.isalpha()
    # suite des instructions
```

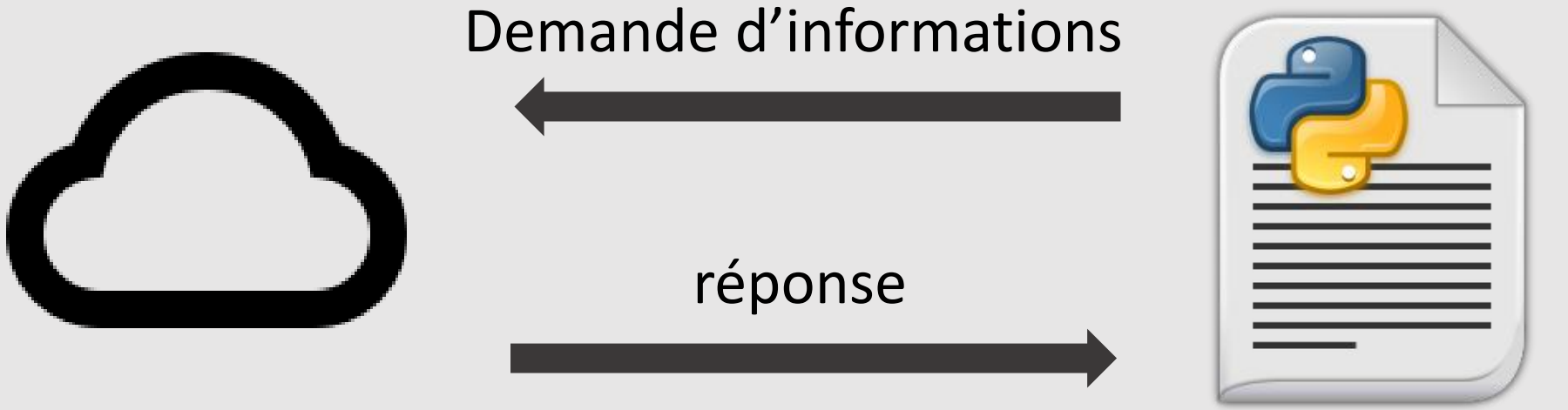
Donnés récupérées à partir d'un fichier texte


```
# Ouverture de fichier
f = open ("etudiants.txt", "r")

# Lecture des données
data = f.readlines() # f.read()

# afficher le contenu du fichier
print(data)
```

Donnés récupérées à partir d'une requête HTTP

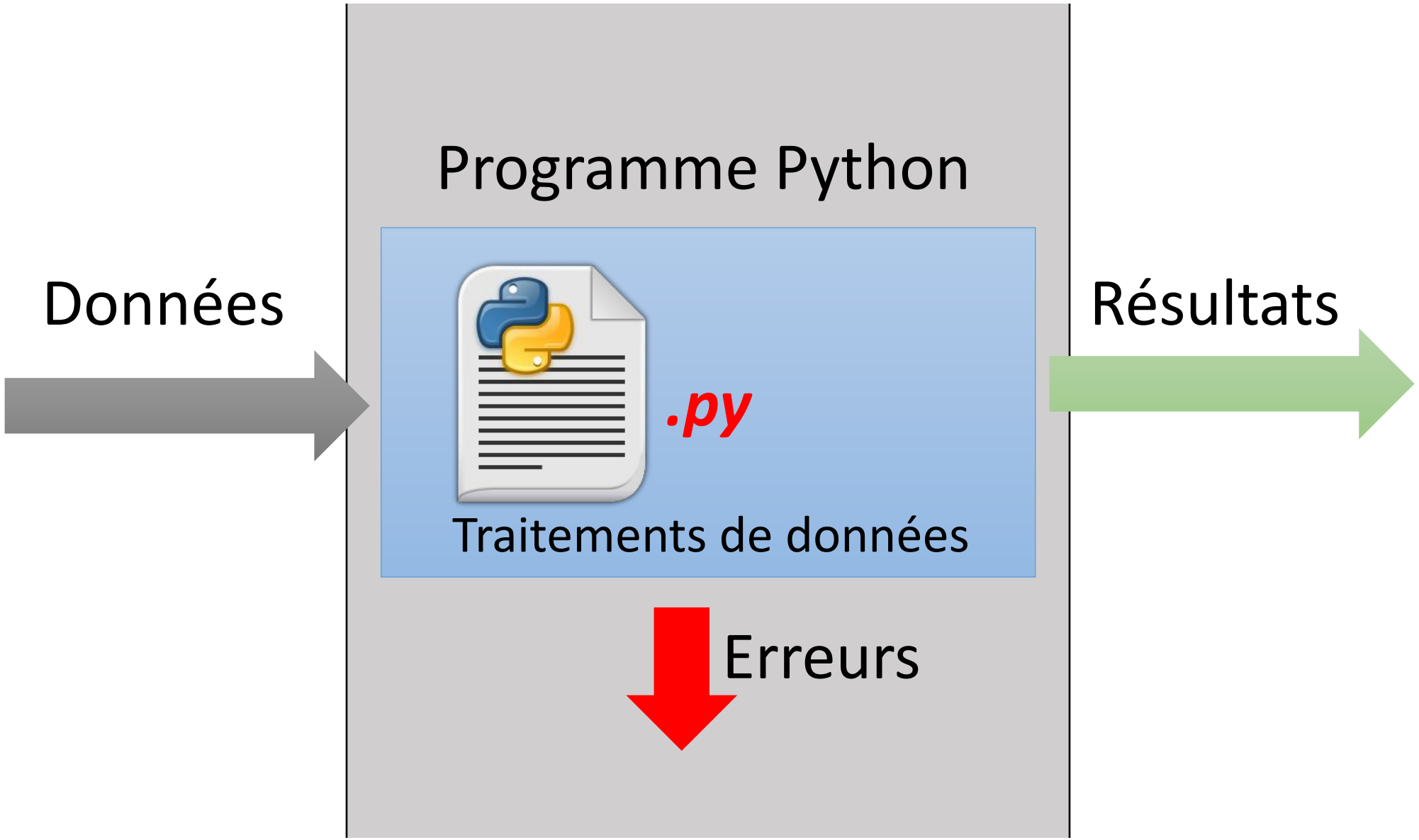


Donnés récupérées à partir d'une autre machine

Machine / capteurs …

Demande d'informations

réponse

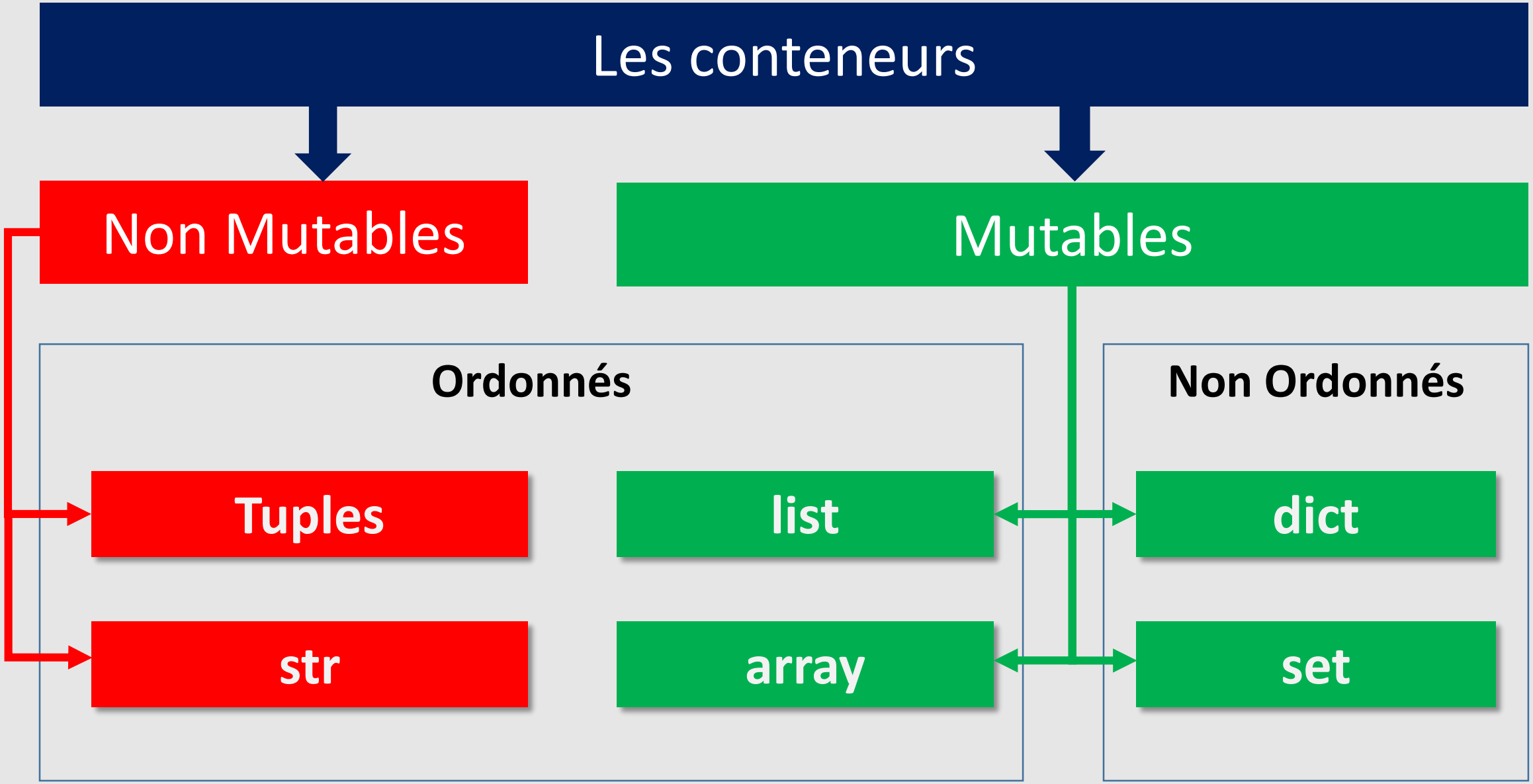

Programme Python
Données
.py
Traitements de données
Résultats
Erreurs

Traitements des données et gestion des erreurs

- Stocker les données reçues dans des **variables**
- Manipuler ces variables avec des fonctions applicables/offertes par leurs **types**
- Les fonctions et les opérateurs appliquées à ces variables peuvent générer des **exceptions**
  - Donc, il faut rattraper ces exceptions et les gérer afin d'éviter l'arrêt brutal du programme : **try** … **except**
  - On peut même générer nous même des exceptions grâce à la commande **raise**

Traitements des données et gestion des erreurs

- Stocker les données reçues dans des **variables**
- Manipuler ces variables avec des fonctions applicables/offertes par leurs **types**
- Les types de données disponibles en Python sont:
- **Données simples :** une seule valeur
  - ➔ type simple : **int, float, complex**,
- **Données complexes :** plusieurs valeurs identifiées par une seule variable


Les conteneurs
Non Mutables
Mutables
Ordonnés
Non Ordonnés
Tuples
list
dict
str
array
set

Programme Python

Données

.py

Traitements de données

Résultats

Erreurs

1 Afficher des résultats sur écran


```
print("Bonjour")

n = print("Saisir un nombre positif : ")
type(n) ?

print(5) + 5 ?
```

La fonction **print**(message)

- Permet d'afficher des données sur un écran.
- Accepte toujours une chaine de caractères : str
- Ne retourne aucun résultat

1 Afficher des résultats sur écran

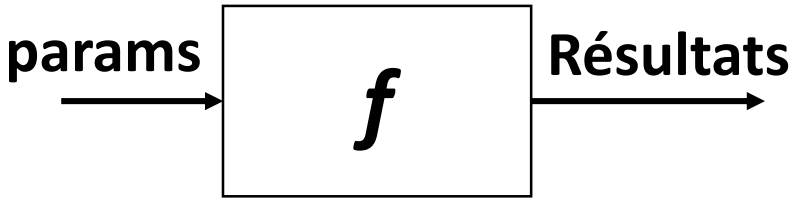

```
>>> n = print(5)
```

La fonction **print**(message)

- Fait un appel implicite à str(message) pour convertir le type de message en str

2 Résultats retournées à la fin d'exécution d'une fonction


```
def f():
    …
    return resultat
```



Une fonction en Python toujours retourne un résultat
Si l'instruction **return est absente**,
Python retourne par défaut (implicitement) la valeur None de type **NoneType**

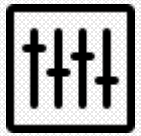
3 Résultats enregistrées dans un fichier texte


```
# Ouverture de fichier
f = open ("nombres.txt", "w")

# Ecriture des données dans un fichier texte
for i in range(5):
      f.write(str(i))

f.close() # enregistrement des données
```

Résultats envoyées à une autre machine


Résultats


Demande d'informations

réponse

Machine / capteurs …



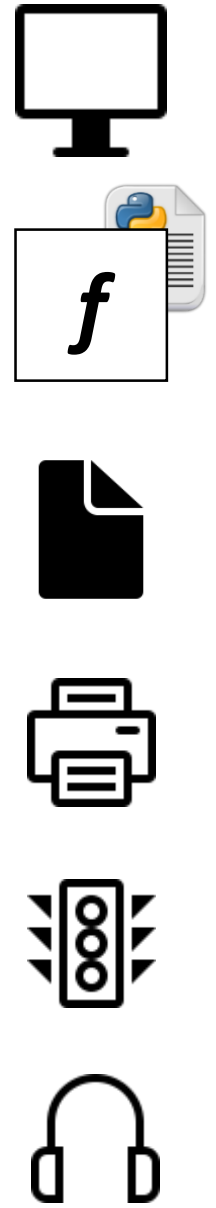

Données

Programme Python

.py

Traitements de données

Résultats



Erreurs