Principaux types de données

Données numériques

- Type *integer*

```
7
8 for i in range(50):
9     print(i, ":", type(i))
```

- Type *float*

Données alphanumériques

Le type *string*

```
>>> phrase1 = 'les oeufs durs.'
>>> phrase2 = '"Oui", répondit-il,'
>>> phrase3 = "j'aime bien"
>>> print(phrase2, phrase3, phrase1)
"Oui", répondit-il, j'aime bien les oeufs durs.
```

Le caractère spécial ****

- Permet d'écrire sur plusieurs lignes une commande qui serait trop longue pour tenir sur une seule
- À l'intérieur d'une chaîne de caractères, il permet d'insérer un certain nombre de codes spéciaux (apostrophes, sauts de ligne, guillemets…)

```
>>> txt3 = '"N\'est-ce pas ?" répondit-elle.'
>>> print(txt3)
"N'est-ce pas ?" répondit-elle.
>>> Salut = "Ceci est une chaîne plutôt longue\n contenant plusieurs lignes \
... de texte (Ceci fonctionne\n de la même façon en C/C++.\n\
...     Notez que les blancs en début\n de ligne sont significatifs.\n"
>>> print(Salut)
Ceci est une chaîne plutôt longue
 contenant plusieurs lignes de texte (Ceci fonctionne
 de la même façon en C/C++.
    Notez que les blancs en début
 de ligne sont significatifs.
```

Triple quotes

- Pour insérer plus aisément des caractères spéciaux ou exotiques dans une chaîne, sans utiliser \ ou pour faire accepter \ lui-même dans la chaîne :

```
>>> a1 = """
... Usage: trucmuche[OPTIONS]
... { -h
...   -H hôte
... }"""

>>> print(a1)

Usage: trucmuche[OPTIONS]
{ -h
  -H hôte
}
```

Accès aux caractères individuels d'une chaîne

- Chaînes de car. font partie des données appelées **données composites :**
- Caractères d'une chaîne sont disposées toujours dans un certain ordre

```
>>> ch = "Christine"
>>> print(ch[0], ch[3], ch[5])
C i t
```

Opérations sur les chaînes

- Concaténation

```
a = 'Petit poisson'
b = ' deviendra grand'
c = a + b
print(c)
petit poisson deviendra grand
```

- Déterminer la longueur (le nombre de car.) : fonction len()

```
>>> ch ='Georges'
>>> print(len(ch))
7
```

- Convertir en nombre entier : fonction int() et en nombre réel : fonction float()

```
>>> ch = '8647'
>>> print(ch + 45)
```

→ *** erreur *** : on ne peut pas additionner une chaîne et un nombre

```
>>> n = int(ch)
>>> print(n + 65)
8712                    # OK : on peut additionner 2 nombres
```

- Indiçage, extraction, longueur

```
>>> nom = 'Cédric'
>>> print(nom[1], nom[3], nom[5])
é r c
>>> print(nom[-1], nom[-2], nom[-4], nom[-6])
c i d C
>>>
```

```
>>> print(len(nom))
6
```

- Extraction de fragments de chaînes

```
>>> print(ch[:3])       # les 3 premiers caractères
Jul
>>> print(ch[3:])       # tout ce qui suit les 3 premiers caractères
iette
```

- Concaténation, répétition

```
>>> n = 'abc' + 'def'        # concaténation
>>> m = 'zut ! ' * 4         # répétition
>>> print(n, m)
abcdef zut ! zut ! zut ! zut !
```

Appartenance d'un élément à une séquence : l'instruction *in* utilisée seule

```
car = "e"
voyelles = "aeiouyAEIOUYàâéèêëùîï"
if car in voyelles:
    print(car, "est une voyelle")
```

Les chaînes sont des séquences non modifiables

- Tester :

```
salut = 'bonjour à tous'
salut[0] = 'B'
print(salut)
```

```
salut = 'bonjour à tous'
salut = 'B' + salut[1:]
print(salut )
```

Les chaînes sont comparables

- utile pour trier par ordre alphabétique

```
 7
 8 n="hello"
 9 if n<"salut":
10     place="précède"
11 elif n>"salut":
12     place="suit"
13 else:
14     place="se confond avec"
15 print("le mot", n, place, "le mot 'salut' dans l'ordre alphabétique")
```

- ne fonctionne bien que pour des mots qui sont tous entièrement en minuscules, ou entièrement en majuscules, et qui ne comportent aucun caractère accentue car les majuscules et minuscules utilisent des ensembles de codes distincts.
- les caractères accentuées sont encodés en dehors de l'ensemble constitué par les caractères du standard *ASCII*.

Fonctions prédéfinies

print()

- Séparateur par défaut est l'espace

```
7
8 print("a","b")
```

- On peut le remplacer avec l'argument « sep »

```
>>> print("Bonjour", "à", "tous", sep ="*")
Bonjour*à*tous
 >>> print("Bonjour", "à", "tous", sep ="")
Bonjouràtous
```

- On peut remplacer le saut à la ligne terminal avec l'argument « end »

```
>>> n =0
>>> while n<6:
...     print("zut", end ="")
...     n = n+1
...
zutzutzutzutzut
```

input ()

- Interaction avec l'utilisateur
- Cette fonction provoque une interruption dans le programme courant et l'utilisateur est invité à entrer des caractères au clavier et à terminer avec <Enter>

```
prenom = input("Entrez votre prénom : ")
print("Bonjour,", prenom)
```

- La fonction renvoie toujours une chaîne de car.

```
>>> a = input("Entrez une donnée numérique : ")
Entrez une donnée numérique : 52.37
>>> type(a)
<class 'str'>
>>> b = float(a)      # conversion de la chaîne en un nombre réel
>>> type(b)
<class 'float'>
```

Fonctions intégrées

- **len(ch)** renvoie la longueur de la chaine **ch**
- **float(ch)** convertit la chaine **ch** en un nombre reel (*float*)

```
>>> a = float("12.36") # Attention : pas de virgule décimale !
>>> print(a + 5)
17.36
```

- **int(ch)** convertit la chaine **ch** en un nombre entier

```
>>> a = int("184")
>>> print(a + 20)
204
```

- **str(obj)** convertit (ou represente) l'objet **obj** en une chaine de caractères. **obj** peut être une donnée d'à peu près n'importe quel type :

```
>>>> a, b = 17, ["Emile", 7.65]
>>> ch =str(a) +" est un entier et " +str(b) +" est une liste."
>>> print(ch)
17 est un entier et ['Emile', 7.65] est une liste.
```

Véracité/fausseté d'une expression

- Toute valeur numérique autre que 0 est « vraie ». Seule la valeur 0 est « fausse »

```
ch = input('Entrez un nombre entier quelconque')
n =int(ch)
if n:
    print("vrai")
else:
    print("faux")
```

Ce script n'affiche « faux » que si vous entrez la valeur 0.

Astuce :

if n: désigne if n!=0

Véracité/fausseté d'une expression

- Testons le caractère « vrai » ou « faux » **d'une chaîne de car**.

```
ch = input("Entrez une chaîne de caractères quelconque")
if ch:
    print("vrai")
else:
    print("faux")
```

Ce script affiche « faux » pour toute chaîne vide et « vrai » pour toute chaîne contenant au moins un caractère.

L'instruction **if ch:** est équivalente à **if ch!=" ":** pour nous (humains)

```
ch =input("Veuillez entrer un nombre : ")
n =int(ch)
if n % 2:
     print("Il s'agit d'un nombre impair.")
else:
     print("Il s'agit d'un nombre pair.")
```

Une chaîne vide est égale à faux. Si on ne tape rien (enter), la boucle s'arrête.

```
while True:
    mot = input("Entrez un mot quelconque : (<enter> pour terminer)")
    if mot =="":
        break
    if mot < "limonade":
        place = "précède"
    elif mot > "limonade":
        place = "suit"
    else:
        place = "se confond avec"
    print("Le mot", mot, place, "le mot 'limonade' dans l'ordre alphabétique")
```

Booléens

- a=0
- a==5

False

- a>-8

True

Mots-clefs *break*

- Break : permet d'interrompre une boucle

```
while 1:  #1 est toujours vrai->boucle infini
        lettre=input("tapiez 'Q' pour quitter :")
        if lettre=="Q":
                print("fin de la boucle")
                break
```

La boucle *while* a pour condition 1, cad une condition sera toujours vraie.

Mot clé *continue*

```
i=1
while i<20:
        if i%3==0:
                i=+4          #on ajoute 4 à i
                print("on incrémente i de 4. i est maintenant égale à", i)
                continue #on retourne au while sans exécuter les autres lignes
        print ("la variable i=", i)
        i+=1
```

Arrivé au mot clé continue Python n'exécute pas la fin du bloc mais revient au début de la boucle en testant à nouveau la condition du while.

Travail à faire

- On a 7 jours de la semaine : *Jack, Kack, Lack, Mack, Nack, Oack, Pack et Qack*. Ecrivez un petit script qui génère tous ces noms à partir des deux chaînes suivantes : **prefixes = 'JKLMNOP'** et **suffixe = 'ack'**
- Ecrire un script qui détermine si une chaîne contient ou non le caractère « e »
- Ecrire un script qui parcourt une chaine : lorsqu'il rencontre une voyelle, il l'affiche, si c'est une consonne, il met « * »
- Ecrire un script qui recopie une chaîne (dans une nouvelle variable), en insérant des astérisques entre les caractères
  - Exemple : « iris » devra devenir « i*r*i*s »
- Ecrire un script qui recopie une chaîne (dans une nouvelle variable) en l'inversant
  - Exemple : « iris » devra devenir « siri »