Python
Types de base

Nicolas Delestre

INSA
ROUEN NORMANDIE

Objets, attributs et méthodes

- Tout est objet en Python
- Un objet a un état défini par les ≪ valeurs ≫ de ces attributs
- On accède à un attribut `a` d'un objet `o` à l'aide de la notation pointée `o.a`
- On peut agir (interroger, modifier, obtenir d'autres objets) sur l'objet en envoyant un message à un objet
- On envoie un message (`m`) à un objet à l'aide de la notation pointée avec entre parenthèses (obligatoires) des paramètres effectifs (optionnels) : `o.m([param1, ...])`
- Python cherche alors la méthode (le code python) à invoquer (interpréter)

INSA ROUEN NORMANDIE

Identité vs égalité

- En python une variable référence un objet
- Deux variables `a` et `b` sont considérées comme égales si elles référencent des objets qui sont égaux (même état mais références différentes)
  - `a == b` retourne `True`
  - `a is b` retourne `False`
- Deux variables sont considérées commes identiques si elles référencent le même objet
  - `a == b` retourne `True`
  - `a is b` retourne `True`

INSA ROUEN NORMANDIE

Le type `bool`

- Deux objets immuables à référence unique : `True` et `False`
- Trois opérateurs : `or`, `and` et `not`

Opérateurs de comparaison qui retourne un booléen

`<`, `>`, `<=`, `>=`, `==` (égal), `!=`, `is` (identique) et `is not`

INSA ROUEN NORMANDIE

Nombres 1 / 2

Les types numériques

Types de données immuables

- `int` représentant les entiers signés
  - préfixe des constantes (par défaut représentation décimale) : `0b` ou `0B` pour binaire, `0o` ou `0O` pour octale, `0x` ou `0X` pour notation hexadécimale,
  - `1_000_000`
  - de 0 à 256 références uniques

```
>>> a = 2
>>> b = 1 + 1
>>> a is b
True
```

```
>>> a = 300
>>> b = 300
>>> a is b
False
```

- `float` représentant les flottants signés : `1.`, `1.0`, `1e10`, `1.E-10`
- `complex` représentant les nombres complexes : `2+3j`, `3.j+2`

Opérateurs et fonctions sur les types numériques

- Opérateurs : `+`, `-`, `*`, `/`, `//`, `%`, `**`
- Fonctions : `abs`, `int`, `float`, `complex`, `divmod`, `pow`
- Méthode (uniquement sur les complexes) : `conjugate`
- Opérateurs bits à bits (uniquement sur les `int`) : `~`, `<<`, `>>`

INSA ROUEN NORMANDIE

Sequences 1 / 4

tuple

- Suite immuable d'objets, qui peuvent être muables, de type divers
- Exemples : `1,1.0,"abc",a` ou `(1,1.0,"abc",a)` ou `tuple(s)` avec `s` une séquence

range

- Suite immuable d'entiers ordonnés séparés d'un certains pas (par défaut de 1)
- Syntaxe : `range(fin)`, `range(debut, fin[, pas])`
- L'entier de `fin` n'est pas inclu
- Exemple : `range(10)`

list

- Suite muable d'objets de type divers
- Exemples : `[1,1.0,"abc",a]` ou `list(s)` avec `s` une séquence

str

- Suite immuable de caractères UTF-8
- Constantes peuvent utiliser des simples quotes, doubles quotes ou des triples simples quotes ou triples doubles quotes
- De nombreuses méthodes permettant d'interroger, de découper, de retrouver, de remplacer et de formater une chaîne. À chaque fois elles retournent une valeur (par exemple la nouvelle chaîne calculée) `https://docs.python.org/fr/3/library/stdtypes.html#text-sequence-type-str`
- Mise en forme de chaîne avec l'utilisation de l'opérateur `%` (utilisation d'un tuple si plusieurs valeurs) ou des *f-string* (à partir de la version 3.6)

```
>>> a=1
>>> b="une chaîne"
>>> "a vaut %d et b vaut '%s'" % (a,b)
"a vaut 1 et b vaut 'une chaîne'"
>>> f"a vaut {a} et b vaut {b}"
'a vaut 1 et b vaut une chaine'
```

slice

- Permet de désigner une partie d'une séquence

```
>>> a=tuple(range(0,20,2)) # tuple composé des 10 premiers nombres pairs
>>> a
(0, 2, 4, 6, 8, 10, 12, 14, 16, 18)
```

```
>>> a[slice(None)]
(0, 2, 4, 6, 8, 10, 12, 14, 16, 18)
>>> # d'indice >=0 et <5
>>> a[slice(0,5)]
(0, 2, 4, 6, 8)
>>> a[slice(1,4)]
(2, 4, 6)
>>> a[slice(-2,None)]
(16, 18)
>>> # d'indice >=0 et <5 par pas de 2
>>> a[slice(0,5,2)]
(0, 4, 8)
```

```
>>> a[:]
(0, 2, 4, 6, 8, 10, 12, 14, 16, 18)
>>> # d'indice >=0 et <5
>>> a[0:5]
(0, 2, 4, 6, 8)
>>> a[1:4]
(2, 4, 6)
>>> a[-2:]
(16, 18)
>>> # d'indice >=0 et <5 par pas de 2
>>> a[0:5:2]
(0, 4, 8)
```

Opérations, fonctions, méthodes

opérations `in`, `not in`, `+`, `*`, `[i]`, `[i:j]`, `[i:j:k]`

fonctions `len`, `min`, `max`

méthodes `index(X[, i[, i]])`, `count(x)`

Opérations supplémentaires pour les séquences muables

opérations `s[i] = x`, `s[i:j] = t`, `s[i:j:k] = t`

fonctions `del s[i:j]`, `del s[i:j:k]`

méthodes `append`, `clear`, `copy`, `extend`, `insert`, `pop`, `remove`, `reverse`

INSA ROUEN NORMANDIE

Ensemble

set et frozenset

- Ensemble d'objets *hashables*
- Exemples : `{1,2,3,1}` ou `set(s)` avec `s` une séquence pour obtenir un ensemble muable, ou `frozenset(s)` pour obtenir un ensemble immuable

Opérations, fonctions

fonction	méthode	opérateur
		`in`
		`not in`
`len`		
	`isdisjoint`	
	`issubset`	`<=`
		`<`
	`issuperset`	`>=`
		`>`

fonction	méthode	opérateur
	`union`	`|`
	`intersection`	`&`
	`difference`	`-`
	`symmetric_difference`	`^`

Dictionnaire

dict

- Association de clés et de valeurs
- Les clés doivent être *hashables* (pas possible pour les séquences muables)

Exemples

```
>>> d={1:"a",(1,2):"b"}
>>> d[1]
'a'
>>> d[(1,2)]
'b'
```

Opérations, fonctions, méthodes

opérations `in`, `not in`, `[i]`,

fonction `len`

méthodes `clear`, `copy`, `get`, `items`, `keys`, `pop`, `popitem`, `update`, `values`

Conclusion

Dans ce cours nous avons

- rappelé ce qu'est un objet, un attribut, une méthode et un message
- rapellé la différence entre égalité et identité et les opérateurs associés
- vu que de base Python propose :
  - un type pour représenter les booléens : `bool` (immuable)
  - trois types pour les nombres : `int` (immuable), `float` (immuable), `complex` (immuable)
  - des types pour les séquences : de caractères : `str` (immuable), d'entiers positifs : `range` (immuable), d'objets : `list` (muable), `tuple` (immuable)
  - deux types pour les ensembles mathématiques : `set` (muable), `frozenset` (immuable)
  - un type pour les dictionnaires (ou tableaux associatifs) : `dict` (muable)
- vu que l'on peut désigner une sous partie d'une séquence à l'aide des `slice`
- vu que Python propose du ≪ sucre syntaxique ≫ facilitant la création des listes (`[]`), des tuples (`()`), des ensembles muables (`{}`), des dictionnaires (`{}`) et des slices (`:`)

INSA ROUEN NORMANDIE