EPFL

ICC
Programmation

SIE/GCG, **Cours 11**

6 mai 2024

Jean-Philippe Pellet

Previously, on ICC Programmation…

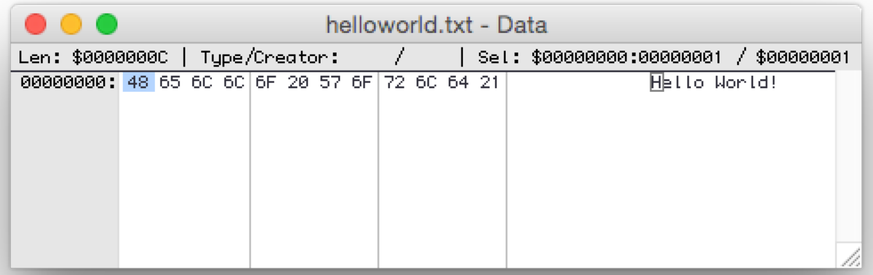
- Types de base en Python: int, float, str, bool
- Méthodes, fonctions et slicing pour calculer des valeurs dérivées
- Conditions pour exécuter du code selon la valeur d'une expression booléenne: `if <condition>: ... else: ...` et ses variantes
- Boucles pour exécuter du code plusieurs fois:
- Déclaration de fonctions avec type de retour et paramètres:
  - `def calculate_area(r: float) -> float: return ...`
- Utilisation de listes, sets et dictionnaires
- Déclaration de classes: `@dataclass class Rectangle: ...`
- Création, chargement, manipulation et sauvegarde d'images
- Programmation dynamique pour trouver des seams
- Fonctions comme valeurs, paramètres; fonctions d'ordre supérieur

Cours de cette semaine

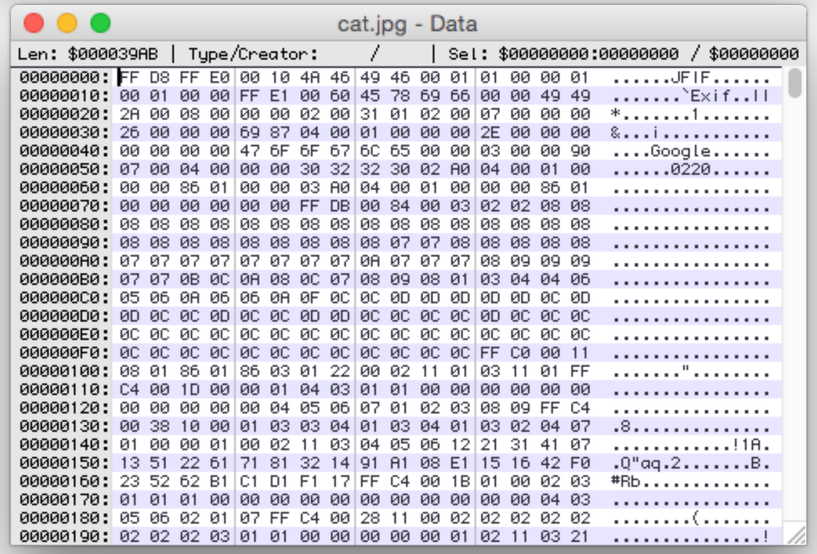
Lecture/écriture de fichiers texte
Opérations sur des chaînes de caractères

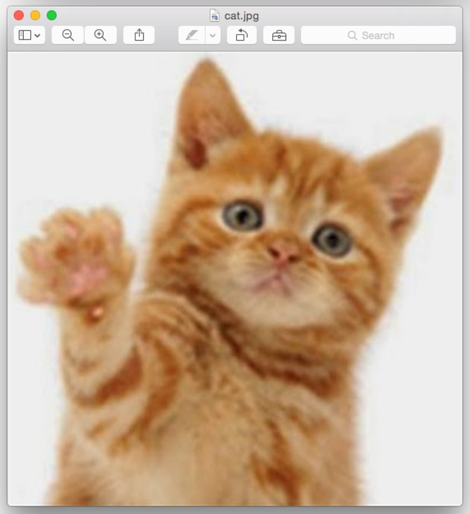
Fichiers et bytes

helloworld.txt



cat.jpg

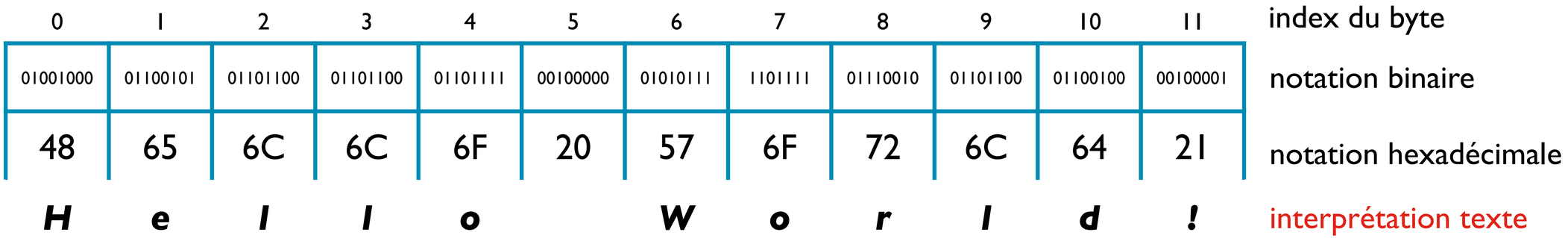




Un fichier est une séquence de bytes qui, interprétés correctement, ont du sens

Les fichiers texte

une séquence de bytes (1 byte = 8 bits)



+ quelques informations supplémentaires:
nom, extension/format, taille, droits d'accès, date de création,
date de la dernière modification, ..., position physique sur le disque

Le système qui interprète chaque byte (ou séquence de bytes) pour retrouver le caractère correspondant repose sur un certain encodage des caractères

ASCII Charset

American Standard Code for Information Interchange (1963)

0	Null character
1	Start of header
...	
9	Tabulator (“\t”)
10	Line feed (“\n”)
...	
32	Space
...	
48	“0”
...	...
57	“9”

65	“A”
66	“B”
67	“C”
...	...
90	“Z”
...	
97	“a”
...	...
122	“z”
...	
127	Delete

Manquent: à, é, ä, €, ...

Unicode, UTF-8 Charset

http://fr.wikipedia.org/wiki/Unicode

http://fr.wikipedia.org/wiki/UTF-8

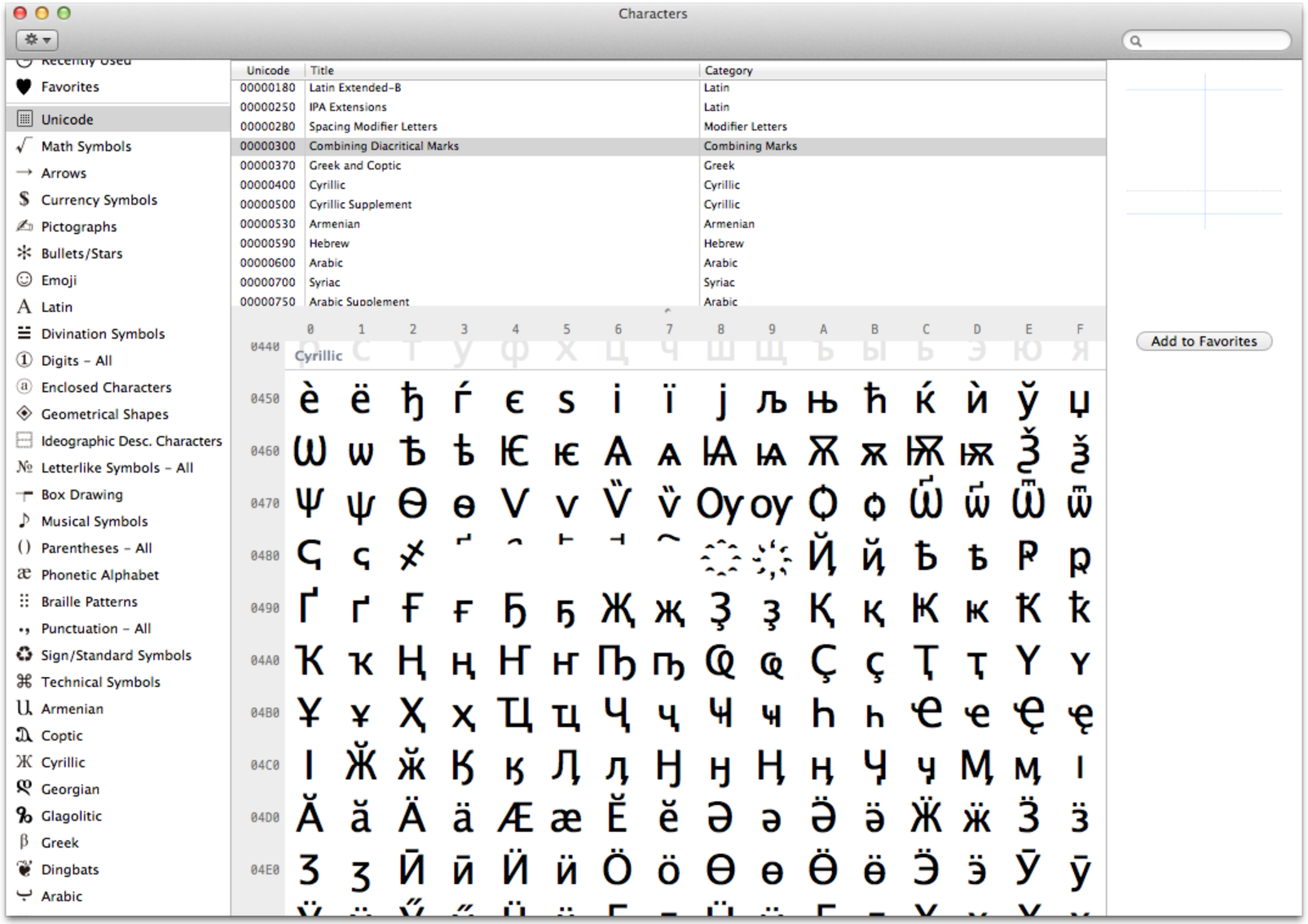
Unicode:

Liste avec tous les caractères qui existent,
y compris les caractères chinois, japonais, ...
(env. 1 million théoriquement, env. 120'000 utilisés)

UTF-8:

Format de codage des signes Unicode;
utilise entre 1 et 4 bytes pour un caractère

Tableau des caractères Unicode



Lire un fichier texte

Démo

```
file = open("movies.txt", "r", encoding="utf-8")
contents = file.read()
file.close()
```

En fait, on n'écrit pas ceci!

Pour être sûr de ne pas oublier le close(), on utilise un *with… as*

On ouvre ce fichier en lecture (*read*)

```
with open("movies.txt", "r", encoding="utf-8") as file:
    contents = file.read()

print(contents)
```

Lecture!

On donne toujours l'encodage à utiliser pour ne pas avoir de surprise

Contient, sous forme de grand string unique, l'intégralité du fichier *movies.txt*

Écrire un fichier texte

Comme pour la lecture, on utiliser un *with… as*, qui fait un *close()* automatique

On ouvre ce fichier en écriture (*write*)

```
with open("results.txt", "w", encoding="utf-8") as file:
    file.write(...)
```

Le travail est fait par la méthode *write(…)*

```
    file.write("une ligne\nune autre ligne")
```

Pour écrire plusieurs lignes, on doit indiquer le caractère de nouvelle ligne \n

```
    for ...:
        file.write(" ... \n")
```

Si les résultats à écrire sont générés au fur et à mesure, on peut les écrire petit à petit (ici, ligne par ligne) avec des appels répétés de *write()*

Traiter les données lues

- Pour de petits fichiers, on peut tout lire d'un coup…
- … Mais comment traiter le contenu ligne par ligne?
  - Méthode split(), qui sépare une chaîne de caractère en une liste de sous-chaînes
  - Conversions de sous-chaînes en int, en datetime, en bool, etc.
  - Tranformations de chaînes avec tests, split(), replace()
  - Modélisation des données lues avec des classes
- Exemple: lecture du fichier *movies.txt*

Démo

```
Year;Length;Title;Subject;Actor;Actress;Director;Popular
INT;INT;STRING;CAT;CAT;CAT;CAT;INT;BOOL;STRING
1990;111;Tie Me Up! Tie Me Down!;Comedy;Banderas, Antoni
1991;113;High Heels;Comedy;Bosé, Miguel;Abril, Victoria;
1983;104;Dead Zone, The;Horror;Walken, Christopher;Adams
1979;122;Cuba;Action;Connery, Sean;Adams, Brooke;Lester,
1978;94;Days of Heaven;Drama;Gere, Richard;Adams, Brooke
1983;140;Octopussy;Action;Moore, Roger;Adams, Maud;Glen,
1984;101;Target Eagle;Action;Connors, Chuck;Adams, Maud;
1989;99;American Angels: Baptism of Blood, The;Drama;Ber
1985;104;Subway;Drama;Lambert, Christopher;Adjani, Isabe
1990;149;Camille Claudel;Drama;Depardieu, Gérard;Adjani,
1982;188;Fanny and Alexander;Drama;Ahlstedt, Börje;Adolp
1982;117;Tragedy of a Ridiculous Man;Drama;Tognazzi, Ugo
```

Lecture et modélisation

```
from dataclasses import dataclass
from typing import List

with open("movies.txt", "r", encoding="utf-8") as file:
    contents = file.read()

lines = contents.split("\n")


@dataclass
class Movie:
    title: str
    year: int
    duration: int
    has_awards: bool
```

Lecture de tout le fichier d'un coup

Séparation du contenu de chaque ligne,
à chaque apparition du caractère \n

On décide d'une représentation sous de forme de classe pour nos données à traiter

Création des objets depuis le texte lu

```
movies: List[Movie] = []

substitutions = {
    "&": "and",
    "’": "’",
    "vs.": "vs",
}

for line in lines[2:]:
    parts = line.split(";")

    duration_str = parts[1]
    if duration_str == "":
        duration = 0
    else:
        duration = int(duration_str)

    title = parts[2]
    if ", " in title:
        title = " ".join(title.split(", ")[::-1])
    for key, value in substitutions.items():
        title = title.replace(key, value)

    movie = Movie(title, int(parts[0]), duration, parts[8] == "Yes")
    movies.append(movie)
```

On prépare une liste de *Movies*, pour l’instant vide

Les substitutions de caractères qu’on voudra faire dans le titre des films

On saute les deux premières lignes (cf. exemple de contenu)

Chaque ligne est segmentée selon les points-virgules qu’elle contient

De temps en temps, la durée est vide… on la met alors à 0. Sinon, on convertit de *str* vers *int*

On «nettoie» le titre, on applique les substitutions

On crée un nouvel objet de type *Movie* et on l’ajoute à la liste

Traitement des données et écriture

```
movies.sort(key=lambda m: m.duration)
```

On trie les films selon un critère de tri donné par ce que cette fonction lambda retourne pour chaque film; ici, sa durée

On prépare un fichier de sortie pour y écrire

```
with open("results.txt", "w", encoding="utf-8") as file:
    total_duration = 0

    for movie in movies:
        if movie.has_awards:
            file.write(f"{movie.duration:3} min: “{movie.title}” ({movie.year})\n")
            total_duration += movie.duration

    file.write(f"\nTotal duration: about {total_duration // 60} hours")
```

On y écrit ce qu’on veut: ici, la durée, le titre et l’année des films qui ont reçu un prix, chacun sur une ligne

Dans la boucle, on en a profité pour calculer la durée totale de ces films, qu’on écrit comme ligne finale de ce fichier

Résumé: opérations sur les strings

- Tests
  - Test de longueur, de casse
  - Test de préfixe, suffixe, contenance

```
text = "..."

if len(text) > 20:
if text == text.lower():
```

```
if text.startswith("..."):
if text.endswith("..."):
if "..." in text:
```

- Split
  - Sépare un string en une liste selon un délimiteur

```
text = "a,b,c"
parts = text.split(",")
```

- Join
  - Crée un string à partir d'une liste en insérant un délimiteur

```
", ".join(parts)
```

- Replace
  - Rechercher-remplacer

```
new_text = text.replace("a", "A")
```

Pas discuté: les expressions régulières, un moyen très puissant de manipuler du texte

Résumé Cours 11

- Un caractère est représenté par une série de bytes selon un encodage précis
  - Aujourd'hui, UTF-8 est le plus utilisé
- On peut facilement lire et écrire des fichiers texte en Python
  - Nouvelle structure: le *with*… *as* pour un *close()* automatique
- On peut traiter les données lues avec:
  - Des transformations de strings
  - Des conversions vers d'autres types
  - La construction d'objets via une modélisation par des classes