Python Cheat Sheet - BUT Informatique

Structures natives

list `[1,2,3]` : Séquence ordonnée modifiable
dict `{"a":1}` : Associations clé-valeur
set `{1,2,3}` : Éléments uniques
tuple `(1,2)` : Séquence immutable

List Comprehensions

```
[x**2 for x in range(10)]
[x for x in range(20) if x%2==0]
{x: x**2 for x in range(5)}
{x%3 for x in range(10)}
```

Unpacking & Enumerate

```
a,b,c = [1,2,3]
premier,*milieu,dernier = [1,2,3,4]
a,b = b,a

for i,val in enumerate(liste):
    print(f"{i}: {val}")

for nom,age in zip(noms,ages):
    print(f"{nom} a {age} ans")
```

Méthodes utiles

```
# List
liste.append(x); liste.extend([...])
liste.pop(); liste.sort(); sorted(liste)

# Dict
d.get(key, defaut); d.setdefault(k, v)
d.update(autre); d.pop(key)

# Set
s.add(x); s1|s2; s1&s2; s1-s2
```

Built-in essentiels

```
len(obj)  # Longueur
sum(iterable)  # Somme
min(iterable); max(iterable)
all(iterable); any(iterable)
sorted(iterable)  # Trier
zip(it1, it2)  # Combiner
map(func, iter); filter(func, iter)
```

String formatting

```
nom = "Alice"; age = 25
print(f"{nom} a {age} ans")
print(f"{nom:>10}")
print(f"{3.14159:.2f}")
```

Slicing

```
lst = [0,1,2,3,4,5]
lst[2:5]   # [2,3,4]
lst[:3]    # [0,1,2]
lst[3:]    # [3,4,5]
lst[::2]   # [0,2,4]
lst[::-1]  # Inverser
lst[-1]    # Dernier
lst[-3:]   # 3 derniers
```

Context Managers

```
# Mauvais
f = open('f.txt')
contenu = f.read()
f.close()

# Bon
with open('f.txt') as f:
    contenu = f.read()

# Multiple
with open('in.txt') as fi, \
     open('out.txt','w') as fo:
    fo.write(fi.read())
```

Gestion erreurs

```
try:
    val = int(input("Nb: "))
    res = 10 / val
except ValueError:
    print("Pas un nombre")
except ZeroDivisionError:
    print("Division par zéro")
except Exception as e:
    print(f"Erreur: {e}")
finally:
    print("Toujours exécuté")

if age < 0:
    raise ValueError("Négatif")
```

EAFP vs LBYL

```
# LBYL (non pythonique)
if key in dico:
    val = dico[key]
else:
    val = default

# EAFP (pythonique)
try:
    val = dico[key]
except KeyError:
    val = default

# Meilleur
val = dico.get(key, default)
```

Idiomes pythoniques

```
# Liste vide
if not liste:       # OK
```

```python
if len(liste)==0:   # NON

x = valeur or default

for k,v in dico.items():  # OK

", ".join(liste_str)  # OK

from collections import Counter
compteur = Counter(liste)
```

Arguments de fonctions

```python
# Par défaut
def saluer(nom, titre="M."):
    return f"{titre} {nom}"

# *args : nb variable d'arguments
def somme(*nombres):
    return sum(nombres)

# **kwargs : arguments nommés
def info(**data):
    for k, v in data.items():
        print(f"{k}: {v}")

# Ordre: pos, *args, kw="def", **kwargs
```

Type Hints

```python
def saluer(nom: str, age: int) -> str:
    return f"{nom} a {age}"

from typing import List, Dict, Optional
def trouve(id: int) -> Optional[str]:
    return res.get(id)
```

Astuces pratiques

```python
print(f"{variable=}")  # Debug Py3.8+
a, b = b, a  # Swap
"ha"*3; [0]*5  # Répéter

# Ternaire
x = v_vrai if cond else v_faux

# Chaîner comparaisons
if 0 < x < 10:  # OK

# Walrus operator (Python 3.8+)
if (n := len(liste)) > 10:
    print(f"Liste longue: {n}")

# defaultdict - valeur par défaut
from collections import defaultdict
d = defaultdict(list)
d['key'].append(1)  # Pas de KeyError

# enumerate avec start
for i, val in enumerate(liste, start=1):
    print(f"Item {i}: {val}")
```

Bibliothèques courantes

```python
# Data science
import numpy as np; import pandas as pd
import matplotlib.pyplot as plt

# Web & API
import requests; from flask import Flask

# Utilitaires
import json, csv, datetime
from pathlib import Path
from collections import Counter, defaultdict
from itertools import chain, combinations
```

Le Zen de Python

```python
import this
```

Principes clés :

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Readability counts
- One obvious way to do it
- Hard to explain = bad idea

Conventions PEP 8

```python
variable_name   # snake_case
ClassName       # PascalCase
CONSTANTE       # MAJUSCULES
_private        # Interne
```

Indentation: 4 espaces (PAS tabs)

Espaces: `x = 1` OK / `x=1` NON; `func(x, y)` OK / `func(x,y)` NON

Max: 79 caractères/ligne

Docstrings

```python
def moyenne(notes):
    """Calcule moyenne.
    Args: notes (list)
    Returns: float
    Raises: ValueError si vide
    """
    if not notes:
        raise ValueError("Vide")
    return sum(notes)/len(notes)
```

Outils qualité

```
pip install black    # Formatage auto
black fichier.py

pip install flake8   # Vérif. PEP 8
flake8 fichier.py

pip install mypy     # Type checking
mypy fichier.py
```

Ressources utiles

Documentation officielle

- docs.python.org/3/
- pep8.org/

Pratique et exercices

- exercism.org/tracks/python
- leetcode.com/

Tutoriels

- realpython.com/
- python.plainenglish.io/