Variables

bool `booleen = True`
int `entier = 5`
float `reel = 5.7`
str `chaine = "Bonjour"`
list `liste = [3, 7, 8]`

Opérateurs numériques

`+`	addition
`-`	soustraction
`/`	division
`*`	multiplication
`%`	modulo
`**`	exposant
`//`	division entière

Opérateurs de comparaison

`==`	égalité
`!=`	différence
`>`	plus grand
`<`	plus petit
`>=`	plus grand ou égal
`<=`	plus petit ou égal

Opérateurs logiques

`and`	ET logique
`or`	OU logique
`not`	NON logique

Opérations sur les chaînes de caractères

`chaine[i]` accès à la lettre i
`chaine[-1]` accès à la dernière lettre
`chaine[i:j]` accès aux lettres de i à j-1
`"Bonjour"+"àvous"` concaténation
`len(chaine)` obtenir la longueur de `chaine`

Opérations sur les listes

`liste[i]` accès à l'élément i
`liste[-1]` accès au dernier élément
`liste[i:j]` accès aux éléments de i à j-1
`[3, 5]+[9, 10]` concaténation
`len(liste)` obtenir la longueur de `liste`

Structures de test

Test simple

```
if entier > 0:
    print("entier est positif")
```

Test avec alternative

```
if entier > 0:
    print("entier est positif")
else:
   print("entier est négatif ou
      nul")
```

Test avec plusieurs conditions

```
if entier > 0:
    print("entier est positif")
elif entier == 0:
   print("entier est nul")
else:
   print("entier est négatif")
```

Test d'appartenance

```
if "o" in chaine:
    print("le mot", chaine, "
       contient la lettre o")
```

Intructions itératives

Instruction while

```
a = 0
while a < 5:
   a = a + 1
   print(a)
```

Instruction for

```
for element in liste:
   print(element)


for i in range(1, 51):
   print(i)
```

Opérations sur les dictionnaires

`dico = {}` création d'un dictionnaire vide
`dico[cle] = val` la valeur val est associée à cle
`dico[cle]` accès à la valeur associée à cle
`del dico[cle]` supprime la clé et sa valeur
`if cle in dico` teste s'il y a une valeur associée à cle
`len(dico)` nombre de paires cle-valeur
`for cle in dico` parcours des clés
`dico.keys()` séquence de toutes les clés
`dico.values()` séquence de toutes les valeurs
`dico.items()` séquence de tous les tuples clé-valeur

Classes

`def MaClasse` création d'une classe MaClasse
`__init__` nom du constructeur
`self` nom de l'instance courante
`m1 = MaClasse()` instantiation (cas d'un constructeur sans paramètres)

Modules

Imports

`import un_module` import simple du module un_module
`from un_module import un_element` import d'uniquement un élément (classe, fonction, ...) d'un module
`import un_module as m` import et renommage (alias)

Utilisation (cas d'un import simple)

`un_module.une_fonction()` appel d'une fonction de un_module
`un_module.une_classe` récupération d'une classe de un_module
`un_module.une_variable` récupération d'une variable de un_module

Création de module

Le code du module se met dans un fichier mon_module.py
Le nom du fichier (sans le .py) représente le nom du module qu'il faudra importer

Interfaces graphiques

`import PySimpleGUI as sg` import de PySimpleGUI
`layout = [[ element1], [element2 ]]` création d'un layout
`window = sg.Window('Titre de la fenêtre', layout )` création de la fenêtre

Gestion des événements

```
while True:  # La boucle d'évènements
    # On récupère les évènements dans la fenêtre
    event, values = window.read()

    # On vérifie si l'utilisateur quitte
    if event == sg.WIN_CLOSED or event == 'Exit':
        break

    # Si c'est le bouton Afficher qui a été cliqué
    if event == 'Afficher':
        # On récupère le texte saisi dans le champ
           nommé -INPUT-
        texte_saisi = values['-INPUT-']

        # On le met dans le champ nommé -OUTPUT-
        window['-OUTPUT-'].update(texte_saisi)
```