Ouverture vers un autre langage de programmation : Python

Table des matières

I. Contexte

Durée : 1 h

Environnement de travail : Repl.it / Environnement local

Pré-requis : Connaître les bases de JavaScript et l'utilisation de la ligne de commande

Contexte

Dans le monde du développement informatique, il existe de nombreux langages, chacun possédant ses avantages et inconvénients. Dans le Web en particulier, on peut trouver des langages tels que JavaScript côté client et d'autres tels que PHP, Java ou .NET pour la partie serveur. Parmi ces langages, on trouve Python. Celui-ci permet aussi de réaliser du développement web côté serveur. Il est aussi souvent utilisé pour des scripts d'administration, des tâches de maintenance ou de déploiement, du calcul mathématique, etc.

II. Introduction au langage Python

Objectifs

- Appréhender la syntaxe de Python
- Installer un compilateur
- Exécuter un premier script sur sa machine

Mise en situation

JavaScript est un langage plutôt orienté client qui s'exécute dans un navigateur web (bien qu'il existe une version serveur avec NodeJS). Python est un langage destiné à être utilisé sur un poste de travail ou un serveur. Il faut ainsi installer un interpréteur sur son poste ou sur le serveur. Nous allons aborder ensemble les différences syntaxiques entre ces deux langages.

Rappel **La factorielle en mathématiques**

En mathématiques, la factorielle d'un entier naturel n est le produit des nombres entiers strictement positifs inférieurs ou égaux à n. Wikipedia[1]

Ainsi, pour calculer la factorielle de 5 (aussi noté **5!**), on obtient **5 * 4 * 3 * 2 * 1**, soit **120**.

Exemple **Syntaxe de Python**

Voici comment l'on pourrait calculer la factorielle d'un nombre. Ici, on demande à l'utilisateur de saisir un nombre entier : on déclenche une erreur s'il n'est pas positif, on calcule sa factorielle, puis on l'affiche.

```
#!/usr/bin/env python3

n = int(input('Entrez un nombre, et sa factorielle sera affichée : '))

if n < 0:
    raise ValueError('Vous devez saisir un nombre positif')

fact = 1
```

[1] https://fr.wikipedia.org/wiki/Factorielle

```
i = 2
while i <= n:
    fact *= i
    i += 1

print(fact)
```

L'équivalent de ce programme pourrait être écrit comme ceci en JavaScript :

```
const n = parseInt(prompt('Entrez un nombre, et sa factorielle sera affichée: '))

if (n < 0) {
    throw 'Vous devez saisir un nombre positif'
}

let fact = 1
let i = 2
while (i <= n) {
    fact *= i
    i += 1
}

document.write(fact)
```

Remarque **Hashbang**

La première ligne du script Python permet de définir avec quel programme il va être exécuté. C'est très utile lorsqu'on veut l'exécuter depuis la ligne de commande, car le script sait quel logiciel il doit utiliser pour se lancer.

Complément

Il est possible de créer un projet Python https://repl.it/. Pour cela, il faut créer un nouveau **repl** et choisir Python. Une fois l'environnement ouvert, il faut copier-coller le code et cliquer sur Run pour pouvoir l'exécuter.

Méthode **Installer un interpréteur Python**

Même s'il est possible de travailler dans un environnement comme Repl.it, il peut être plus simple de travailler dans un environnement local.

Pour Windows, il est fort probable que nous ayons à l'installer manuellement. Sous Linux, il est possible que Python soit déjà installé. Si ce n'est pas le cas, il est aussi possible d'utiliser un gestionnaire de paquets, si notre système d'exploitation nous le permet.

Pour vérifier si Python est déjà installé, nous pouvons exécuter les commandes suivantes :

```
python --version
python3 --version
```

Si l'une de ces commandes renvoie un numéro de version, c'est que Python est installé sur notre machine. Si ce n'est pas le cas, cela signifie que nous avons besoin d'installer un interpréteur Python.

Heureusement pour nous, Python met à disposition des interpréteurs pour tous les systèmes d'exploitation. Pour cela, nous allons pouvoir utiliser un gestionnaire de paquets ou nous rendre sur la page de téléchargement[1] pour télécharger directement l'interpréteur.

[1] https://www.python.org/downloads/

Attention

Sur Windows, il faudra cocher la case **Add Python to PATH** au moment de l'installation pour pouvoir utiliser Python depuis la ligne de commande Windows. Il ne faut donc pas oublier de le faire, ou il sera impossible de lancer les scripts avec les commandes de la prochaine partie.

Méthode **Exécuter un script Python**

Une fois Python installé et disponible depuis la ligne de commande, nous allons créer un fichier quelque part sur notre système avec un éditeur de texte standard (Visual Studio Code, par exemple). Appelons-le `factorielle.py`.

`py` est l'extension des fichiers Python.

Une fois créé, il faut se rendre dans le répertoire où se trouve le fichier grâce à la ligne de commande, et taper l'une des commandes suivantes :

```
python factorielle.py
python3 factorielle.py
```

Le programme devrait démarrer et nous proposer de saisir une valeur afin d'en calculer la factorielle[1].

Grâce à l'ajout du **hashbang**, il est aussi possible de lancer le script grâce à cette commande :

```
test.py
```

Grâce au hashbang, le système d'exploitation est capable de déterminer de quel programme il a besoin pour exécuter le script.

Remarque

Même si un simple éditeur de texte suffit pour écrire des scripts Python, il est conseillé d'utiliser un IDE pour des projets dont l'envergure est plus conséquente qu'un simple test. Pour cela, nous pourrons nous tourner vers PyCharm[2], qui est développé par IntelliJ, par exemple.

Syntaxe **À retenir**

- Même si elle semble différente, la syntaxe de Python n'est pas si éloignée de celle de JavaScript et d'autres langages.
- Afin d'exécuter des scripts Python sur un système d'exploitation, nous devons installer un interpréteur. Il est ensuite possible d'exécuter des scripts sur notre machine afin de les tester plus facilement.
- Pour écrire un script Python, il faut le nommer avec l'extension `.py` et l'éditer avec un simple éditeur de texte ou un IDE.

Complément

Site officiel de Python[3]

[1] https://fr.wikipedia.org/wiki/Factorielle
[2] https://www.jetbrains.com/fr-fr/pycharm/
[3] https://www.python.org/

III. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

[1]

Question

En vous aidant du script suivant, écrivez un script permettant de calculer le carré d'un nombre.

```python
#!/usr/bin/env python3

n = int(input('Entrez un nombre, et sa factorielle sera affichée: '))

if n < 0:
    raise ValueError('Vous devez saisir un nombre positif')

fact = 1
i = 2
while i <= n:
    fact *= i
    i += 1

print(fact)
```

IV. Les variables et les opérateurs

Objectifs

- Affecter une variable
- Connaître les différents types de données
- Manipuler les données grâce à la concaténation et aux opérations arithmétiques

Mise en situation

Avec Python, comme avec n'importe quel langage de programmation, il est possible de manipuler des variables et d'effectuer des opérations de concaténation et arithmétiques. En plus de cela, nous allons voir les différents types de données disponibles dans ce langage et en quoi ils varient avec un langage de programmation tel que JavaScript.

Méthode **Affecter une variable**

Dans n'importe quel langage de programmation, avant d'affecter une variable, il convient de la déclarer. En Python, c'est un peu différent. Il n'existe pas vraiment de mot-clé permet de déclarer une variable, il suffit simplement de l'utiliser dans le cadre d'une affectation.

Par exemple, en JavaScript, nous devrions écrire le code suivant :

```javascript
let x
x = 10
// ou directement
let x = 10
```

[1] https://repl.it/

En Python, c'est simplement :

```
x = 10
```

En Python, donc, pas besoin de déclarer une variable avec un mot-clé, il suffit de simplement l'utiliser.

Fondamental **Les différents types de données**

Parmi les différents types utilisables en Python, nous pouvons retrouver :

- un type texte : `str`
- des types numériques : `int`, `float`, `complex`
- des types séquences : `list`, `tuple`, `range`
- des types mapping : `dict`
- des types set : `set`, `frozenset`
- un type booléen : `bool`
- des types binaires : `bytes`, `bytearray`, `memoryview`

Nous ne rentrerons pas ici dans l'explication de tous les types de données, mais il faut savoir qu'ils sont décrits dans la documentation[1].

Exemple **Une variable de type texte**

Pour utiliser le type texte par exemple, nous pouvons délimiter une chaîne de caractères directement avec des " ou en utilisant la syntaxe avec `str()` :

```
message = "Hello world"
print(type(message))
# ou
message = "Hello world"
print(type(message))
```

Attention **La notation des booléens**

Contrairement à JavaScript, lors de l'utilisation du type booléen, la définition d'une valeur booléenne prend une majuscule : `True` ou `False`. Écrites `true` ou `false`, ces chaînes seront considérées comme des variables et non des valeurs.

```
true = True
```

Ici, nous stockons la valeur booléenne `vrai` dans la variable se nommant `true`.

Remarque **Les commentaires**

En Python, les lignes de commentaires commencent par un `#`. Il n'existe pas vraiment de syntaxe pour un commentaire multi-lignes, mais nous pouvons utiliser la syntaxe de chaîne de caractères multi-lignes :

```
#This is a
#multiline comment
```

```
"""
This is a
multiline comment
"""
```

[1] https://docs.python.org/3/library/datatypes.html

Comme cette chaîne n'est affectée à aucune variable, elle ne sera pas affichée ni utilisée.

Méthode **Opérations de concaténation**

En Python, il existe plusieurs formes de concaténation que nous pouvons mettre en parallèle avec les différentes formes disponibles avec JavaScript. La première s'utilise avec le caractère `+` et l'autre avec une **syntaxe avec tokens**.

Pour concaténer deux chaînes en JavaScript, on utilisera la syntaxe suivante :

```
let message = "Hello " + "world"
```

En Python, c'est pareil :

```
message = "Hello " + "world"
```

Pour l'expression `string literal` en JavaScript, on aura :

```
let hello = "Hello",
    world = "world"
let message = `${hello} ${world}`
```

En Python, c'est un peu différent. Il faudra utiliser une syntaxe avec tokens de remplacement :

```
hello = "Hello"
world = "world"
message = "%s %s" % (hello, world)
```

Méthode **Opérations arithmétiques**

Parmi les opérateurs arithmétiques, nous retrouverons en Python les types que l'on connaît déjà en JavaScript : `+`, `-`, `*`, `/` ou `%`. Nous trouverons aussi des opérateurs inédits, tels que :

- `**` qui calcule l'exponentiel d'un nombre, par exemple `2 ** 5` donnera `32`.
- `//` qui calcule le quotient entier arrondi à l'inférieur d'une division, par exemple `19 // 5` donnera `3`.

Leur utilisation est identique, que ce soit en JavaScript ou en Python.

```
sum = 17 + 5
```

Syntaxe **À retenir**

- En Python, les variables n'ont pas besoin d'être déclarées, il n'existe pas de mot-clé pour cela. On peut les utiliser directement grâce à l'opération d'affectation `=`.
- Les types de données basiques sont similaires aux types JavaScript, mais il en existe des plus complexes.
- La concaténation se fait grâce à l'opérateur `+` ou une syntaxe avec tokens de remplacement.
- Pour les opérations arithmétiques, leur utilisation est identique, même s'il en existe des supplémentaires.

Complément

Types de données[1]

Opérateurs[2]

[1] https://docs.python.org/fr/3/library/datatypes.html
[2] https://docs.python.org/fr/3/reference/lexical_analysis.html#operators

V. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

[1]

Question 1

Écrivez un script qui permettrait à un utilisateur de créer un nouveau produit.

Pour cela, il devra renseigner les informations suivantes :

- Le nom du produit, de type texte
- Son prix HT, un nombre à virgule

Une fois ces informations renseignées, vous afficherez la phrase suivante en utilisant les tokens de remplacement : "*Produit ... créé au prix de ... € HT*".

Question 2

Affichez un second message en calculant le prix TTC du produit, on y appliquera une TVA de 20 %.

VI. Les conditions et les boucles

Objectifs

- Apprendre à écrire une expression booléenne
- Utiliser les structures conditionnelles
- Utiliser les boucles

Mise en situation

Comme en JavaScript, Python permet d'écrire des expressions booléennes et de les utiliser dans le cadre de structure conditionnelle et de boucles. Nous allons voir ensemble la syntaxe à utiliser pour chacune d'entre elles.

Définition **Expression booléenne**

Une expression booléenne est une opération qui permet de déterminer si un cas est vrai ou faux. En Python, nous pouvons en écrire grâce aux opérateurs de comparaison, logiques, d'identité et d'appartenance.

[1] https://repl.it/

Syntaxe **Opérateurs de comparaison**

Parmi les opérateurs de comparaison, on retrouve les mêmes opérateurs qu'en JavaScript. Nous avons donc :

- L'égalité : ==
- L'inégalité : !=
- La supériorité : >
- L'infériorité : <
- Et leur déclinaison avec la nuance d'égalité : >= et <=

Syntaxe **Opérateurs logiques**

Les mêmes opérateurs logiques existent, mais s'écrivent de manière différente par rapport à JavaScript. Nous avons :

- `and` qui retourne vrai, si les deux conditions sont vraies
- `or` qui retourne vrai, si l'une des deux conditions est vraie
- `not` qui retourne vrai, si la condition est fausse

Syntaxe **Opérateurs d'identité**

Les opérateurs d'identité en Python permettent de vérifier que deux objets sont les mêmes, pas au niveau de leur valeur, mais au niveau de leur emplacement mémoire. Nous retrouvons :

- `is` qui retourne vrai, si les deux objets sont les mêmes
- `is not` qui retourne vrai, si les deux objets sont différents

Syntaxe **Opérateurs d'appartenance**

Les opérateurs d'appartenance sont utilisés pour tester qu'une valeur est présente dans un objet. Il existe :

- `in` qui retourne vrai, si la valeur est trouvé dans l'objet
- `not in` qui retourne vrai, si la valeur est absente de l'objet

Exemple **Construire une expression booléenne**

En JavaScript, nous écririons :

```
let expression = x > 5 && y <= 10
```

Alors qu'en Python, ce sera :

```
expression = x > 5 and y <= 10
```

Syntaxe **Structure conditionnelle**

La seule structure conditionnelle disponible en Python est le `if/else`. On déclare sa structure comme ceci :

```
if b > a:
  print("b est plus grand que a")
else:
  print("a est plus grand que b")
```

On peut bien sûr omettre les lignes 3 et 4 si nous n'en avons pas besoin.

Attention

Contrairement à beaucoup d'autres langages, la délimitation des blocs de code en Python se fait grâce à l'indentation. De ce fait, il est très important d'être strict lorsque nous indentons notre code. N'oubliez pas non plus d'ajouter un `:` à la fin de la ligne de déclaration d'un bloc.

Par exemple, ce code ne fonctionnera pas :

```
if b > a:
print("b est plus grand que a")
```

Syntaxe

Le mot-clé `elseif` présent en JavaScript devient `elif` en Python. Nous allons voir un exemple avec une expression booléenne plus complexe :

```
if b > a:
  print("b est plus grand que a")
elif a == b and b > c:
  print("a et b sont égaux et b est plus grand que c")
```

Complément **Imbriquer des structures conditionnelles**

On peut bien sûr imbriquer des structures conditionnelles les unes dans les autres. Attention cependant à l'indentation.

```
if x > 0:
  print("x est positif ")
  if x = 0:
    print("et égal à 0")
  else:
    print("strictement")
```

Syntaxe **Boucles**

Il existe deux types de boucles en Python : `for` et `while`.

Exemple **Boucle for**

La boucle `for` s'utilise comme dans d'autres langages. En JavaScript, nous ferions :

```
let users = ["Nicolas", "Laurent", "Laure"]

for (let user of users) {
  console.log(user)
}
```

En Python, la syntaxe est à peu près la même :

```
users = ["Nicolas", "Laurent", "Laure"]

for user in users:
  print(user)
```

Exemple **Boucle while**

En Python, la syntaxe de la boucle `while` ressemble beaucoup à celle en JavaScript :

```
users = ["Nicolas", "Laurent", "Laure"]
i = 0

```

```
while i < len(users):
    print(users[i])
    i += 1
```

Syntaxe **À retenir**

- En Python, on peut utiliser les mêmes opérateurs de comparaison que dans les autres langages. Les opérateurs logiques, eux, s'écrivent en toutes lettres : `and`, `or`, `not`. Il existe aussi les opérateurs d'identité et d'appartenance.
- Nous pouvons les combiner pour obtenir une expression booléenne, qui sera ensuite utilisée dans des structures conditionnelles (`if`) ou itératives (`for` et `while`). Attention cependant à la syntaxe, et surtout à l'indentation.
- L'indentation a une signification très particulière en Python, car elle sert à identifier un bloc de code.

Complément

Opérations booléennes[1]

Instructions if, for et while[2]

VIII. Exercice : Appliquez la notion

Vous disposez du script suivant, permettant de créer un produit :

```
#!/usr/bin/env python3

name = str(input('Indiquez le nom du produit: '))
price = float(input('Indiquez son prix: '))
available = bool(input('Indiquez si le produit est déjà disponible: '))

message = "Produit %s créé au prix de %s € HT, disponible à la vente = %s" % (name, price, available)
print(message)

priceWithTaxes = price * 1.2
priceWithTaxesMessage = "Le prix TTC sera de %s €" % (priceWithTaxes)
print(priceWithTaxesMessage)
```

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

[3]

Question

À l'aide d'une boucle, modifiez ce script afin de pouvoir ajouter plusieurs produits. L'utilisateur décidera de continuer ou non sa saisie, au moyen d'une question qui lui sera posée.

[2] https://docs.python.org/fr/3/reference/expressions.html?highlight=boolean#boolean-operations
[2] https://docs.python.org/fr/3/reference/compound_stmts.html
[3] https://repl.it/

Attention, il n'est pas possible de récupérer un input booléen aussi facilement qu'un entier, vous devrez donc utiliser une condition pour vérifier que la valeur saisie vaut bien `true` ou `false`. Vous pourrez par exemple vérifier que l'utilisateur a bien saisi "1".

IX. Les fonctions

Objectifs

- Connaître comment appeler une fonction native
- Apprendre à déclarer et appeler une fonction déclarée

Mise en situation

Une fonction s'utilise de la même manière en Python que dans la majorité des langages, dans le sens où il faut dans un premier temps la déclarer, puis l'appeler. On peut lui passer des paramètres en entrée et renvoyer un résultat en sortie. La syntaxe, cependant, est différente de ce qu'on a pu déjà voir en JavaScript.

Méthode **Appeler une fonction native**

On appelle **fonctions natives** les fonctions incluses dans un langage. Ces fonctions sont très utiles pour faire des opérations basiques, et il en existe beaucoup en Python.

Parmi les plus couramment utilisées, il existe :

- Les fonctions de conversion de **type**, aussi appelé *cast* : `int()`, `str()`, `bool()`, etc.
- `input()` qui permet de récupérer une saisie utilisateur
- `print()` qui permet d'afficher un texte à l'écran
- `len()` qui permet de récupérer la longueur d'un objet (chaîne de caractères, tableau, etc.)
- `min()` et `max()` qui permettent de récupérer respectivement la valeur minimale et maximale d'un tableau

Une liste des fonctions est accessible sur cette page[1].

L'appel d'une fonction en Python se fait de la même manière que dans la majorité des langages.

Exemple **Récupérer la longueur d'un objet**

```
1 string = "Hello world"
2 length = len(string)
3 print("La chaîne \"%s\" fait %s caratères de long" % (string, length))
```

Complément **Les modules**

On peut accéder à d'autres fonctions présentes dans des **modules**. Ceux-ci doivent être importés avec le mot-clé `import`. La ligne d'import d'un module doit être positionnée tout en haut du script.

```
1 import random
2
3 print(random.randint(0, 100))
```

Ici, on affiche une valeur entre `0` et `99`, `100`, non inclus.

[1] https://docs.python.org/3.0/library/functions.html

Syntaxe **Déclarer une fonction**

Afin de déclarer une fonction en Python, on va utiliser le mot-clé `def` plutôt que `function` en JavaScript. La syntaxe, ensuite, est identique à ce qu'on a pu voir avec les structures conditionnelles et les boucles. On peut lui passer des paramètres.

Exemple

```
def say_hello(name, age = 0):
  print("Bonjour " + name)
  if age > 0:
    print("Tu as %s ans" % (age))

say_hello("Nicolas")
```

On peut déclarer des paramètres avec des valeurs par défaut (ici, `age`) et écrire d'autres blocs de code à l'intérieur.

Résultat :

```
Bonjour Nicolas
```

Syntaxe **Retourner une valeur**

Si on veut pouvoir retourner une valeur, on utilise le mot-clé `return`, comme dans les autres langages.

Exemple

```
def get_full_name(first_name, last_name):
  return first_name + " " + last_name

say_hello(get_full_name("Romain", "Delpierre"), 25)
```

Résultat :

```
Bonjour Romain Delpierre
Tu as 25 ans
```

Complément

Lorsqu'on appelle une fonction en Python, on peut aussi intervertir les paramètres d'une fonction en les nommant.

```
say_hello(age = 25, name = "Romain")
```

Syntaxe **À retenir**

- En Python, il existe des fonctions natives disponibles dans le langage qui nous permettent de faire des opérations basiques.
- Pour déclarer une fonction, on commence avec le mot-clé `def`, puis on indique son nom et ses paramètres éventuels. On délimite le code de la fonction avec l'indentation, et on peut retourner un résultat avec le mot-clé `return`.
- On l'appelle ensuite simplement par son nom en indiquant entre parenthèses les paramètres, dans l'ordre de leur définition. Si on souhaite les passer dans un autre ordre, on peut les nommer dans l'appel de la fonction.

X. Exercice : Appliquez la notion

Vous disposez du script suivant, permettant de créer plusieurs produits :

```python
#!/usr/bin/env python3

addNew = True

while addNew is True:
  name = str(input('Indiquez le nom du produit: '))
  price = float(input('Indiquez son prix: '))


  message = "Produit %s créé au prix de %s € HT" % (name, price)
  print(message)

  priceWithTaxes = price * 1.2
  priceWithTaxesMessage= "Le prix TTC sera de %s €" % (priceWithTaxes)
  print(priceWithTaxesMessage)

  response = int(input('Souhaitez-vous ajouter un nouveau produit ? '))

  if 1 == response:
    addNew = True
  else:
    addNew = False

```

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

[1]

Question 1

Découpez votre code afin de définir une fonction `createProduct` permettant de créer un produit, et une autre intitulée `calcPriceWithTaxes` permettant de calculer un prix TTC.

Question 2

À l'aide de la documentation[2], importez le module `datetime` nécessaire et, lors de la création d'un produit, indiquez la date et l'heure courante au format **jj/mm/YYYY hh:mm**.

Aidez-vous des fonctions `now()` et `strftime()`.

XI. Auto-évaluation

A. Exercice final

Exercice

Exercice

En Python, laquelle de ces syntaxes permettrait d'affecter la valeur `5` à une variable intitulée `n` ?

- ○ let n = 5
- ○ n = 5

[1] https://repl.it/
[2] https://docs.python.org/fr/3.8/library/datetime.html

- $n = 5;
- ^n = 5

Exercice

La première ligne du script Python permet de définir avec quel programme il va être exécuté. Comment l'appelle-t-on ?

- Le cashback
- Le hashtag
- Le hashbank
- Le hashbang

Exercice

Quelle sera l'extension d'un fichier Python ?

- .pt
- .pn
- .py

Exercice

En Python, la définition d'une valeur booléenne...

- Ne prend pas de majuscule
- Prend une majuscule

Exercice

En Python, quel est l'opérateur de concaténation ?

- .
- +
- &&

Exercice

En Python, que permet l'opérateur ** ?

- Il permet d'effectuer une multiplication
- Il n'existe pas
- Il permet de calculer un exponentiel

Exercice

En Python, quel mot-clé permet de traduire "sinon si" ?

Exercice

En Python, quels sont les types de boucles disponibles ?

- [ ] while
- [ ] for...in
- [ ] foreach

Exercice

En Python, grâce à quel mot-clé est-il possible d'accéder à d'autres fonctions que les fonctions natives ?

Exercice

En Python, quel est l'équivalent de la fonction JavaScript `prompt()` ?

- ○ ask()
- ○ prompt()
- ○ input()
- ○ print()

Exercice

En Python, si l'on souhaite déclarer une nouvelle fonction...

- ○ Il n'y a pas de mot-clé
- ○ On doit utiliser le mot-clé `def`
- ○ On doit utiliser le mot-clé `let`
- ○ On doit utiliser le mot-clé `new`

B. Exercice : Défi

Grâce à toutes les notions, et surtout aux syntaxes, que nous avons abordées dans ce cours, nous allons réaliser ensemble un jeu du pendu.

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

[1]

Question

La réalisation du jeu se fera en plusieurs étapes :

1. Définissez une liste de mots dans un tableau
2. Créez une première boucle de jeu qui va gérer le lancement d'une nouvelle partie
3. Initialisez les différentes variables du jeu (compteur d'essai, lettres proposées, etc.)
4. Récupérez un nombre au hasard entre 0 et le nombre d'éléments du tableau, puis récupérez le mot correspondant
5. Demandez à l'utilisateur des propositions de lettres

[1] https://repl.it/

6. Pour chaque lettre, indiquez que l'utilisateur :
    - a juste, en lui affichant le mot sous la forme d'une chaîne avec les lettres non découvertes masquées (ex : _a__a___ pour *pantalon* lorsqu'il découvre le a)
    - a faux, en lui affichant le mot masqué avec le pendu représentant le compteur d'erreurs (cf annexe)
7. À la fin du jeu, indiquez au joueur s'il a gagné (s'il a trouvé toutes les lettres du mot) ou s'il a perdu (s'il a atteint le maximum de tentatives)
8. Proposez-lui de recommencer une nouvelle partie

Pour afficher le pendu, vous pourrez vous aider de la fonction suivante, qui prend en paramètre le nombre d'erreurs :

```
def display_hangman(num_tries):
  print(" ____")
  print("|    |")
  print("|    %s" % ("O" if num_tries >= 1 else " "))
  print("|   %s%s%s" % ("/" if num_tries >= 3 else " ", "|" if num_tries >= 2 else " ", "\\" if num_tries >= 4 else " "))
  print("|   %s %s" % ("/" if num_tries >= 5 else " ", "\\" if num_tries >= 6 else " "))

```

Résultat final :

```
Quelle lettre voulez-vous ?
Vous devez proposer une lettre
Quelle lettre voulez-vous ? a
Mot à trouver : _a__a___
Quelle lettre voulez-vous ? aa
Vous devez proposer une lettre
Quelle lettre voulez-vous ? e
 ____
|    |
|    O
|
|
Mot à trouver : _a__a___
Quelle lettre voulez-vous ? i
 ____
|    |
|    O
|    |
|
Mot à trouver : _a__a___
Quelle lettre voulez-vous ? o
Mot à trouver : _a__a_o_
Quelle lettre voulez-vous ? n
Mot à trouver : _an_a_on
Quelle lettre voulez-vous ? t
Mot à trouver : _anta_on
Quelle lettre voulez-vous ? p
Mot à trouver : Panta_on
Quelle lettre voulez-vous ? k
 ____
|    |
|    O
|   /|
|
Mot à trouver : Panta_on
Quelle lettre voulez-vous ? l
```

```
Mot à trouver : Pantalon
Vous avez gagné malgré 3 erreurs ! Le mot était "Pantalon"
Voulez-vous recommencer ? O/n o
Quelle lettre voulez-vous ? e
Mot à trouver : ____e
Quelle lettre voulez-vous ? i
 ____
|    |
|    O
|
|
Mot à trouver : ____e
Quelle lettre voulez-vous ? o
Mot à trouver : _o__e
Quelle lettre voulez-vous ? u
 ____
|    |
|    O
|    |
|
Mot à trouver : _o__e
Quelle lettre voulez-vous ? s
 ____
|    |
|    O
|   /|
|
Mot à trouver : _o__e
Quelle lettre voulez-vous ? r
 ____
|    |
|    O
|   /|\
|
Mot à trouver : _o__e
Quelle lettre voulez-vous ? l
 ____
|    |
|    O
|   /|\
|   /
Mot à trouver : _o__e
Quelle lettre voulez-vous ? n
 ____
|    |
|    O
|   /|\
|   / \
Mot à trouver : _o__e
Vous avez perdu ! Le mot était "Botte"
Voulez-vous recommencer ? O/n n
```

Solutions des exercices

Exercice p. Solution n°1

```python
#!/usr/bin/env python3

n = int(input('Entrez un nombre, et son carré sera affiché: '))

square = n *n

print(square)
```

Exercice p. Solution n°2

```python
#!/usr/bin/env python3

name = str(input('Indiquez le nom du produit: '))
price = float(input('Indiquez son prix: '))

message = "Produit %s créé au prix de %s € HT" % (name, price)
print(message)
```

Exercice p. Solution n°3

```python
#!/usr/bin/env python3

name = str(input('Indiquez le nom du produit: '))
price = float(input('Indiquez son prix: '))

message = "Produit %s créé au prix de %s € HT)
print(message)

priceWithTaxes = price * 1.2
priceWithTaxesMessage = "Le prix TTC sera de %s €" % (priceWithTaxes)
print(priceWithTaxesMessage)
```

Exercice p. Solution n°4

```python
#!/usr/bin/env python3

addNew = True

while addNew is True:
  name = str(input('Indiquez le nom du produit: '))
  price = float(input('Indiquez son prix: '))


  message = "Produit %s créé au prix de %s € HT" % (name, price)
  print(message)

  priceWithTaxes = price * 1.2
  priceWithTaxesMessage = "Le prix TTC sera de %s €" % (priceWithTaxes)
  print(priceWithTaxesMessage)

```

```
  response = int(input('Souhaitez-vous ajouter un nouveau produit ? '))

  if 1 == response:
    addNew = True
  else:
    addNew = False

```

Exercice p. Solution n°5

```
#!/usr/bin/env python3
def calcPriceWithTaxes(price):
  priceWithTaxes = price * 1.2
  priceWithTaxesMessage = "Le prix TTC sera de %s €" % (priceWithTaxes)
  print(priceWithTaxesMessage)

def createProduct():
  name = str(input('Indiquez le nom du produit: '))
  price = float(input('Indiquez son prix: '))
  message = "Produit %s créé au prix de %s € HT" % (name, price)
  print(message)
  calcPriceWithTaxes(price)

addNew = True

while addNew is True:
  createProduct()

  response = int(input('Souhaitez-vous ajouter un nouveau produit ? '))

  if 1 == response:
    addNew = True
  else:
    addNew = False

```

Exercice p. Solution n°6

```
#!/usr/bin/env python3
from datetime import datetime

def calcPriceWithTaxes(price):
  priceWithTaxes = price * 1.2
  priceWithTaxesMessage = "Le prix TTC sera de %s €" % (priceWithTaxes)
  print(priceWithTaxesMessage)

def createProduct():
  name = str(input('Indiquez le nom du produit: '))
  price = float(input('Indiquez son prix: '))
  now = datetime.now()
  currentTime = now.strftime("%d/%m/%Y à %H:%M")
  message = "Produit %s créé au prix de %s € HT créé le %s" % (name, price, currentTime)
  print(message)
  calcPriceWithTaxes(price)

```

```
addNew = True

while addNew is True:
  createProduct()

  response = int(input('Souhaitez-vous ajouter un nouveau produit ? '))

  if 1 == response:
    addNew = True
  else:
    addNew = False

```

Exercice p. 15 Solution n°7

Exercice

En Python, laquelle de ces syntaxes permettrait d'affecter la valeur `5` à une variable intitulée `n` ?

- ○ let n = 5
- ◉ n = 5
- ○ $n = 5;
- ○ ^n = 5

Exercice

La première ligne du script Python permet de définir avec quel programme il va être exécuté. Comment l'appelle-t-on ?

- ○ Le cashback
- ○ Le hashtag
- ○ Le hashbank
- ◉ Le hashbang

Exercice

Quelle sera l'extension d'un fichier Python ?

- ○ .pt
- ○ .pn
- ◉ .py

Exercice

En Python, la définition d'une valeur booléenne...

- ○ Ne prend pas de majuscule
- ◉ Prend une majuscule

Exercice

En Python, quel est l'opérateur de concaténation ?

- ○ .
- ◉ +
- ○ &&

Exercice

En Python, que permet l'opérateur `**` ?

- ○ Il permet d'effectuer une multiplication
- ○ Il n'existe pas
- ◉ Il permet de calculer un exponentiel

Exercice

En Python, quel mot-clé permet de traduire "sinon si" ?

elif

Exercice

En Python, quels sont les types de boucles disponibles ?

- ☑ while
- ☑ for...in
- ☐ foreach

Exercice

En Python, grâce à quel mot-clé est-il possible d'accéder à d'autres fonctions que les fonctions natives ?

import

Exercice

En Python, quel est l'équivalent de la fonction JavaScript `prompt()` ?

- ○ ask()
- ○ prompt()
- ◉ input()
- ○ print()

Exercice

En Python, si l'on souhaite déclarer une nouvelle fonction...

- ○ Il n'y a pas de mot-clé
- ◉ On doit utiliser le mot-clé `def`
- ○ On doit utiliser le mot-clé `let`
- ○ On doit utiliser le mot-clé `new`

Exercice p. Solution n°8

```python
import random

def display_hangman(num_tries):
  print(" ____")
  print("|    |")
  print("|    %s" % ("O" if num_tries >= 1 else " "))
  print("|   %s%s%s" % ("/" if num_tries >= 3 else " ", "|" if num_tries >= 2 else " ", "\\" if num_tries >= 4 else " "))
  print("|   %s %s" % ("/" if num_tries >= 5 else " ", "\\" if num_tries >= 6 else " "))

def get_word_to_display(letters, word):
  word_to_display = ""

  for letter in word:
    if letter.lower() in letters:
      word_to_display += letter
    else:
      word_to_display += "_"

  return word_to_display

words = [
  "Serviette",
  "Poivre",
  "Chaise",
  "Vert",
  "Ventre",
  "Parapluie",
  "Goupille",
  "Pantalon",
  "Botte",
  "Girafe"
]

stop_game = False
max_tries = 6

while not stop_game:
  num_tries = 0
  letters = ""

  random_number = random.randint(0, len(words) - 1)

  word_to_find = words[random_number]

  while num_tries != max_tries and "_" in get_word_to_display(letters, word_to_find):
    answer = ""
    while len(answer) != 1:
      answer = input("Quelle lettre voulez-vous ? ")
      if len(answer) != 1:
        print("Vous devez proposer une lettre")

    letter = answer[0].lower()
    letters += letter

    if not letter in word_to_find.lower():
      num_tries += 1
      display_hangman(num_tries)
```

```
58
59      print("Mot à trouver : " + get_word_to_display(letters, word_to_find))
60
61    if num_tries == max_tries:
62      print("Vous avez perdu ! Le mot était \"%s\"" % (word_to_find))
63    else:
64      print("Vous avez gagné malgré %d erreurs ! Le mot était \"%s\"" % (num_tries,
word_to_find))
65
66    stop_game = input("Voulez-vous recommencer ? O/n ")[0].lower() == "n"
67
```