

# Python pour la science

---

Outils Numériques / Semestre 5

*Julien VILLEMEJANE*

- La **physique** est la science qui essaie de **comprendre**, de **modéliser** et d'**expliquer** les **phénomènes naturels** de l'Univers.

- Recherche fondamentale
- Physique expérimentale

EXPERIENCES

OBSERVATIONS

MODELISATION

- ✓ Traiter des données d'expériences
- ✓ Faire ressortir les « tendances »
- ✓ Simuler / Modéliser les phénomènes

# Python et la science











- Langage de programmation
  - **Généraliste** : grâce à une multitude de bibliothèques développées indépendamment
  - **Interprété** (opposé à compilé)
  - **Accessible**

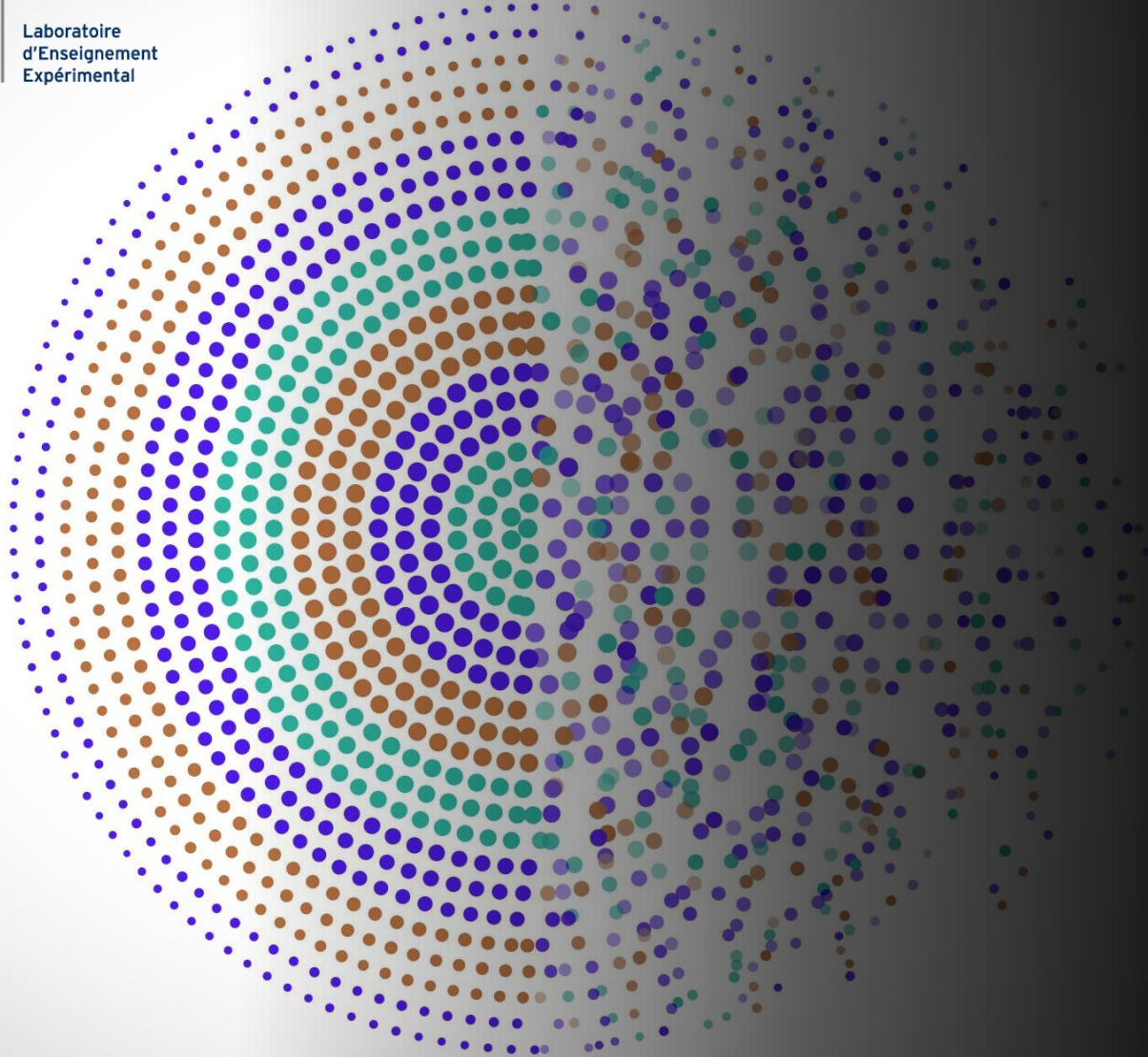


# Python et la science

- TIOBE Index

- An indicator of the **popularity of programming languages**. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors.
- It is important to note that the TIOBE index is **not about the best programming language** or the language in which most lines of code have been written.

Aug 2025	Aug 2024	Change	Programming Language		Ratings	Change
1	1			Python	26.14%	+8.10%
2	2			C++	9.18%	-0.86%
3	3			C	9.03%	-0.15%
4	4			Java	8.59%	-0.58%
5	5			C#	5.52%	-0.87%
6	6			JavaScript	3.15%	-0.76%
7	8	^		Visual Basic	2.33%	+0.15%
8	9	^		Go	2.11%	+0.08%
9	25	^^		Perl	2.08%	+1.17%
10	12	^		Delphi/Object Pascal	1.82%	+0.19%



# Python / Syntaxe et conventions

---

Outils Numériques / Semestre 5



# Python / Syntaxe et conventions

Être capable de **d'écrire un script réutilisable dans un langage de haut niveau** (à but scientifique)

- Utiliser des fonctions du langage avec des paramètres adaptés
- Ecrire des fonctions dans un langage de haut niveau afin de **rendre des parties du code réutilisable**
- Fournir un **code lisible et réutilisable** (convention d'écriture dans le langage, commentaires, documentation...)

# Python / Syntaxe et conventions

- Syntaxe et conventions

**Convention d'écriture en Python** (« résumée » dans des **PEP** (*Python Enhancement Proposals*))

**But : *Rendre son code compréhensible par d'autres***

<https://peps.python.org/pep-0008/>

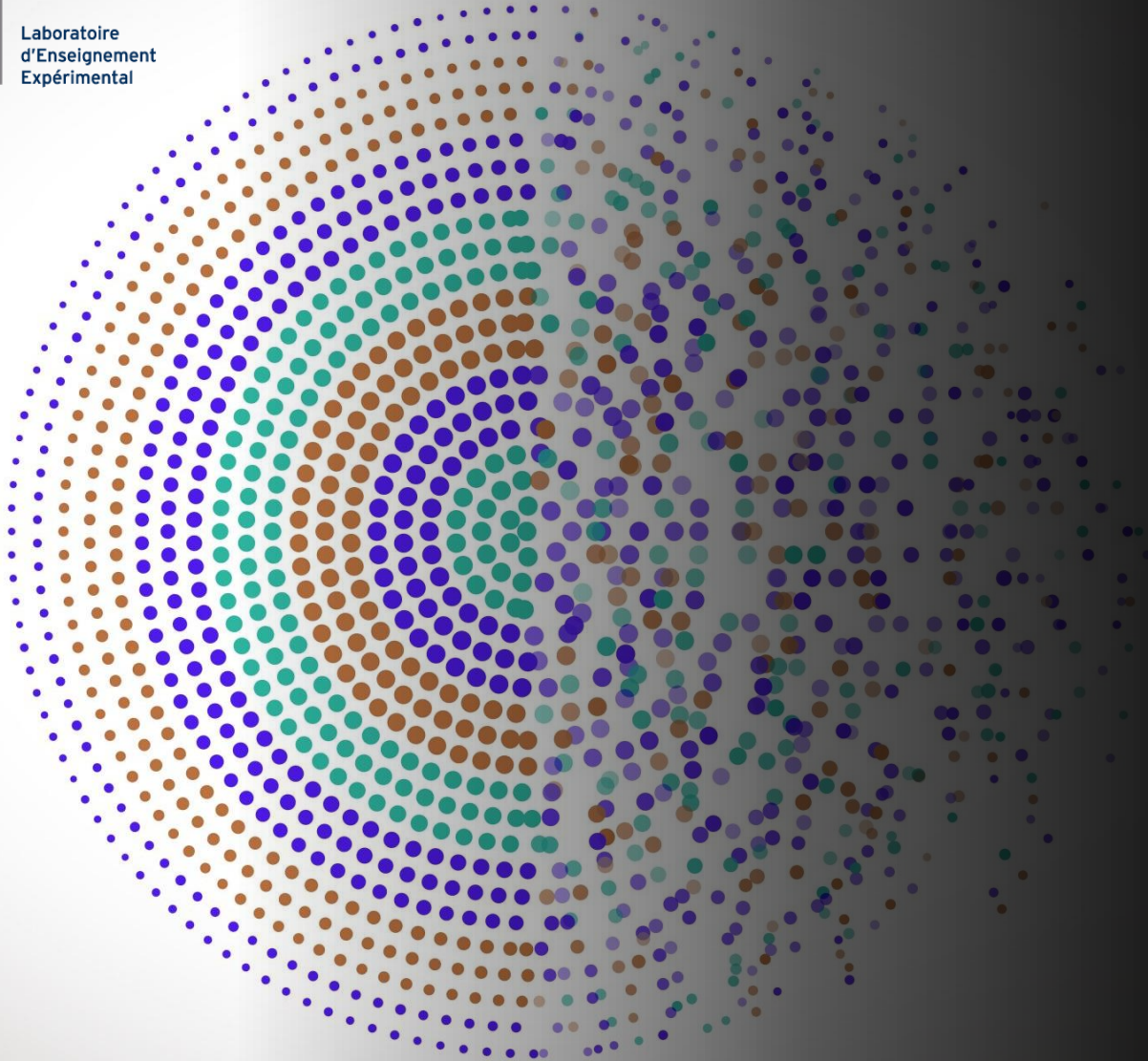
```
my_function(x, y)
my_dictionary = {"lambda": 632, "intensity": 10}
my_array = ('r', 'g', 'b')
my_array[0]
my_dictionary["lambda"]
```

- Créer des fonctions :
  - réutilisables,
  - Paramétrables,
  - fonctionnelles

**But : *Rendre son code réutilisable dans de multiples situations***

**Créer des scripts de test de vos fonctions !**





# Test Python

---

Outils Numériques / Semestre 5

# Python / Syntaxe / QCM 01

- Soit le code suivant :

```
a = 2  
b = 7  
p = a * b
```

- Que vaut **p** à la fin de l'exécution de ce code ?



27



14



9

# Python / Syntaxe / QCM 01

- Soit le code suivant :

```
a = 2  
b = 7  
p = a * b
```

- Que vaut ***p*** à la fin de l'exécution de ce code ?

	27
	14
	9

# Python / Syntaxe / QCM 02

- Soit le code suivant :

```
a = 2  
b = 7  
p = a * b
```

- Quelle(s) syntaxe(s) parmi les suivantes permettent d'afficher la valeur de la variable ***p*** ?



```
print('p')
```



```
print(f'p = {p}')
```




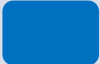

```
print(p)
```

# Python / Syntaxe / QCM 02

- Soit le code suivant :

```
a = 2  
b = 7  
p = a * b
```

- Quelle(s) syntaxe(s) parmi les suivantes permettent d'afficher la valeur de la variable ***p*** ?

	<code>print('p')</code>
	<code>print(f'p = {p}')</code>
	<code>print(p)</code>

# Python / Syntaxe / QCM 02

```
a = 2  
b = 7  
p = a * b
```

```
>>> print(f'p = {p}')
```

```
p = 14
```

**Affichage formaté.** Tout ce qui est entre {} est remplacé par la valeur de la variable.

```
>>> print(p)
```

```
14
```

**Affichage standard.**

```
>>> print('p')
```

```
p
```

Dans ce cas, il s'agit de la **chaîne de caractères** 'p'

# Python / Syntaxe / QCM 03

- Soit le code suivant :

```
A = np.linspace(0,1,101)
b = 3000
C = 10
d = C * np.sin(b*2*np.pi*A)
```

- Que pouvez-vous dire de ce code ?



Ce code ne s'exécute pas



Les noms des variables sont bien choisis



Ce code donne un résultat faux



# Python / Syntaxe / QCM 03

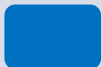
- Soit le code suivant :

```
A = np.linspace(0,1,101)
b = 3000
C = 10
d = C * np.sin(b*2*np.pi*A)
```

- Que pouvez-vous dire de ce code ?



Ce code ne s'exécute pas



Les noms des variables sont bien choisis



Ce code donne un résultat faux

# Python / Syntaxe / QCM 03

```
A = np.linspace(0,1,101)
b = 3000
C = 10
d = C * np.sin(b*2*np.pi*A)
```

<https://peps.python.org/pep-0008/>

## Les noms des variables sont bien choisis

Convention d'écriture en Python (« résumée » dans des **PEP** (*Python Enhancement Proposals*))

- Noms de variable en minuscule et explicites (si possible en anglais) :

```
frequency = 3000
output_signal = np.sin(2*np.pi*frequency*time)
```

- Commentaires pour aider à la compréhension
- Docstrings pour documenter les modules, classes, fonctions...

# Python / Syntaxe / QCM 03

```
time = np.linspace(0,1,101)
frequency = 3000
amplitude = 10
output_signal = amplitude * np.sin(2*np.pi*frequency*time)
```

**Ce code donne un résultat faux**

Ce code génère un **signal sinusoïdal** basé sur le vecteur **time**.

Mais attention ! **Informatique = Monde discrétisé** !

Quelle est le pas d'échantillonnage du vecteur **time** (équivalent au temps) ?

Quelle est la fréquence souhaitée du signal sinusoïdal ?

# Python / Syntaxe / QCM 03

```
time = np.linspace(0,1,101)
frequency = 3000
amplitude = 10
output_signal = amplitude * np.sin(2*np.pi*frequency*time)
```

**Ce code donne un résultat faux**

Ce code génère un **signal sinusoïdal** basé sur le vecteur **time**.

**Pas = 10ms**

Mais attention ! **Informatique = Monde discrétisé !**

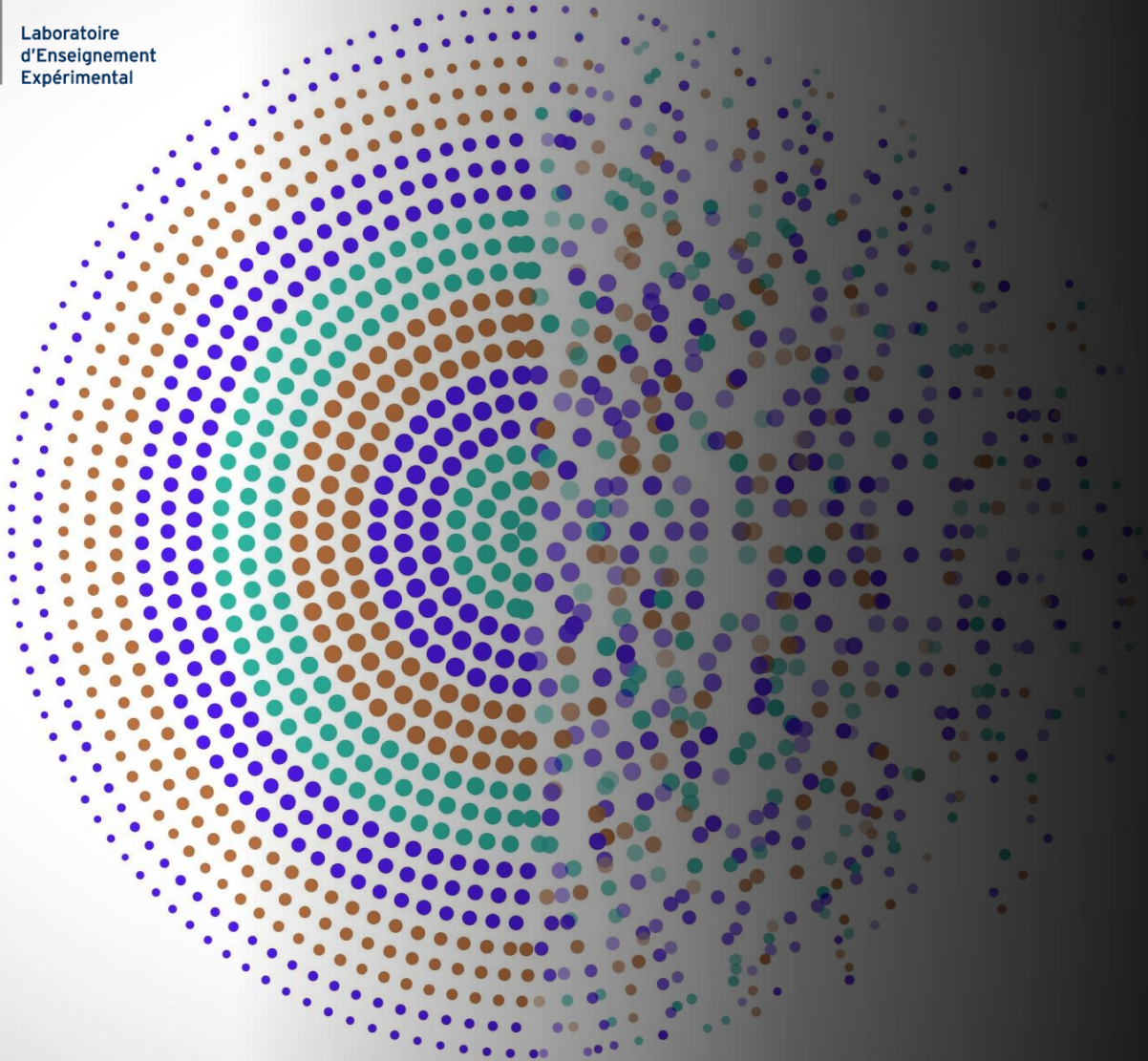
**Période = 0,33ms**

Quelle est le pas d'échantillonnage du vecteur **time** (équivalent au temps) ?

Quelle est la fréquence souhaitée du signal sinusoïdal ?

Est-ce compatible ?

**Critère de Shannon non respecté !**



# Python / Typage des données

---

Outils Numériques / Semestre 5

# Python / Typage / QCM 11

- Soit le code suivant :

```
a = "6"  
n = 4  
c = 1.6  
z = c * n
```

- De quel type est la variable **z** ?



Entière



Réelle (flottant)



Caractère


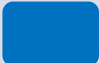

# Python / Typage / QCM 11

- Soit le code suivant :

```
a = "6"  
n = 4  
c = 1.6  
z = c * n
```

```
>>> print(type(z))  
<class 'float'>
```

- De quel type est la variable **z** ?

	Entière
	Réelle (flottant)
	Caractère



# Python / Typage / QCM 12

- Soit le code suivant :

```
a = "6"  
n = 4  
c = 1.6  
res = n*a
```

- Que donne l'affichage de la variable **res** ?



6666



4'a'



Rien

# Python / Typage / QCM 12

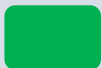
- Soit le code suivant :

```
a = "6"  
n = 4  
c = 1.6  
res = n*a
```

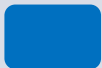
```
>>> print(res)  
6666
```

```
>>> print(type(res))  
???
```

- Que donne l'affichage de la variable **res** ?



6666



4 'a'



Rien

# Python / Typage / QCM 12

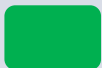
- Soit le code suivant :

```
a = "6"  
n = 4  
c = 1.6  
res = n*a
```

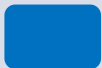
```
>>> print(res)  
6666
```

```
>>> print(type(res))  
<class 'str'>
```

- Que donne l'affichage de la variable **res** ?



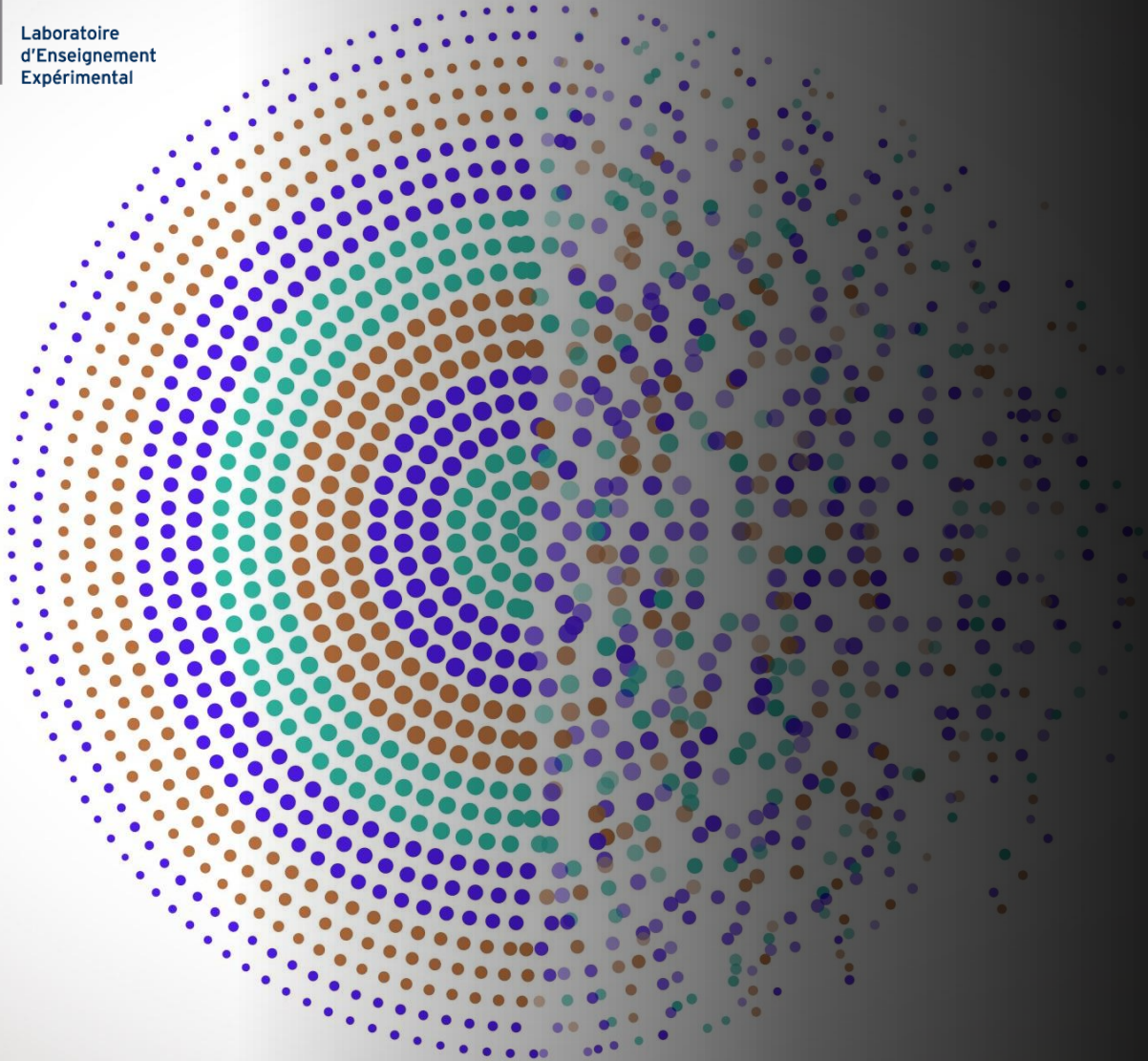
6666



4 'a'



Rien



# Python / Calcul et conditions

---

Outils Numériques / Semestre 5

# Python / Typage / QCM 21

- Soit le code suivant :

```
a = 4  
c = a // 5  
d = a / 5
```

- Quelle est la réponse correcte ?



**c = 0 / d = 0**



**c = 0.8 / d = 0.8**



**c = 0 / d = 0.8**

# Python / Typage / QCM 21


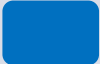

- Soit le code suivant :

```
a = 4  
c = a // 5  
d = a / 5
```

```
>>> print(f'C = {c} / {type(c)}')  
C = 0 / <class 'int'>
```

```
>>> print(f'D = {d} / {type(d)}')  
D = 0.8 / <class 'float'>
```

- Quelle est la réponse correcte ?

	<code>c = 0 / d = 0</code>
	<code>c = 0.8 / d = 0.8</code>
	<code>c = 0 / d = 0.8</code>

# Python / Typage / QCM 22

- Soit le code suivant :

```
a = 7
if a <= 3:
    print(2*a)
```

```
elif a <= 10:
    print(3*a)
else:
    print(4*a)
```

- Quelles sont les énoncés corrects ?



Les indentations sont inutiles



Ce code affiche : 21



Les nombres négatifs ne sont pas traités



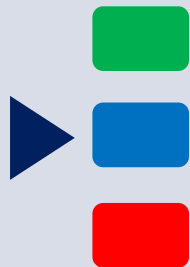
# Python / Typage / QCM 22

- Soit le code suivant :

```
a = 7  
if a <= 3:  
    print(2*a)
```

```
elif a <= 10:  
    print(3*a)  
else:  
    print(4*a)
```

- Quelles sont les énoncés corrects ?



Les indentations sont inutiles

Ce code affiche : 21

Les nombres négatifs ne sont pas traités

# Python / Typage / QCM 23

- Soit le code suivant :

```
a = 49  
b = 49  
c = ( a != b )
```

- Quelles sont les énoncés corrects ?



**c = False**



**c est une variable entière**



**c = not True**

# Python / Typage / QCM 23

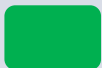
- Soit le code suivant :

```
a = 49  
b = 49  
c = ( a != b )
```

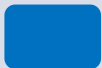
```
>>> print(type(c))  
<class 'bool'>
```

```
>>> print(c)  
False
```

- Quelles sont les énoncés corrects ?



**c = False**



**c est une variable entière**



**c = not True**

# Python / Typage / QCM 23

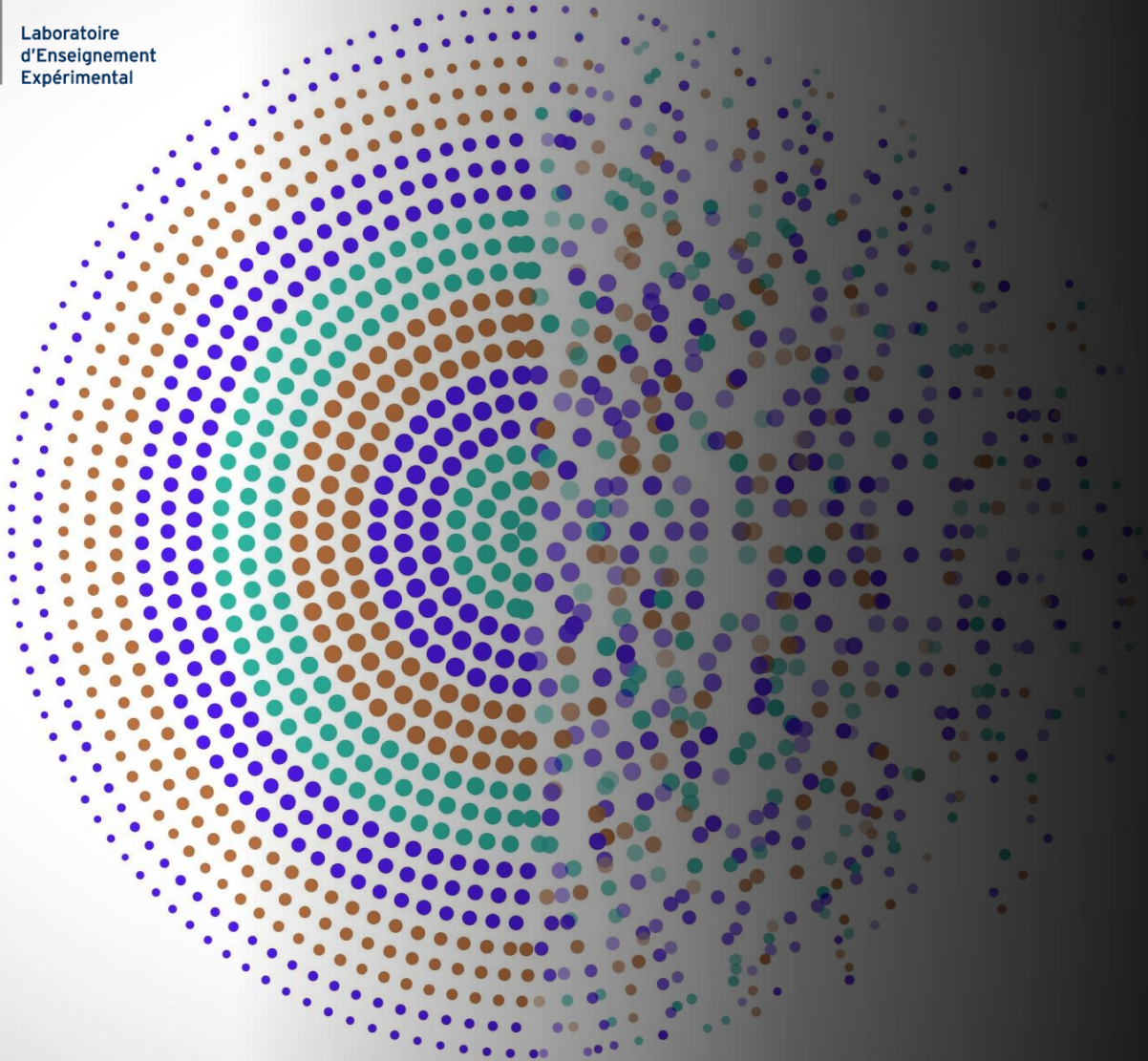
- Soit le code suivant :

```
a = 49  
b = 49  
c = ( a != b )
```

```
>>> if c is not True: print('ok')  
...  
ok
```

- Quelles sont les énoncés corrects ?

- ▶ ☒ **c = False**
- ☐ **c est une variable entière**
- ▶ ☒ **c = not True**



# Python / Listes et itérations

---

Outils Numériques / Semestre 5

# Python / Listes / QCM 31

- Soit le code suivant :

```
a=['7', '8', '2', '0']  
b=['V', 'P', 'A', 'C']  
c=(a*2+b*2)[1:-1]
```

- Quel est le type de la variable **c** ?



Une chaîne de caractères



Une liste



Un entier

# Python / Listes / QCM 31

- Soit le code suivant :

```
a=['7', '8', '2', '0']  
b=['V', 'P', 'A', 'C']  
c=(a*3+b*3)[1:-1]
```

```
>>> print(type(c))  
<class 'list'>
```

- Quel est le type de la variable **c** ?

	Une chaine de caractères
	Une liste
	Un entier



# Python / Listes / QCM 32

- Soit le code suivant :

```
a=['7', '8', '2', '0']  
b=['V', 'P', 'A', 'C']  
c=(a*2+b*2)[1:-1]
```

- Quelle est la taille de la liste **c** ?



**1 élément**



**16 éléments**



**14 éléments**

# Python / Listes / QCM 32

- Soit le code suivant :

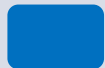
```
a=['7', '8', '2', '0']  
b=['V', 'P', 'A', 'C']  
c=(a*2+b*2)[1:-1]
```

```
>>> print(c)  
['8', '2', '0', '7', '8', '2', '0',  
'V', 'P', 'A', 'C', 'V', 'P', 'A']
```

- Quelle est la taille de la liste **c** ?



1 élément



16 éléments



14 éléments

# Python / Listes / QCM 33

- Soit le code suivant :

```
a=['7', '8', '2', '0']  
b=['V', 'P', 'A', 'C']  
c=(a*2+b*2)[1:-1]
```

- Comment obtenir la longueur de **c** ?



```
print(c.len())
```



```
print(len(c))
```



```
print(c.len)
```

# Python / Listes / QCM 33

- Soit le code suivant :

```
a=['7', '8', '2', '0']  
b=['V', 'P', 'A', 'C']  
c=(a*2+b*2)[1:-1]
```

```
>>> print(len(c))  
14
```

- Comment obtenir la longueur de **c** ?

	<code>print(c.len())</code>
	<code>print(len(c))</code>
	<code>print(c.len)</code>

# Python / Listes

- Soit la liste suivante :

```
a = ['1', '2', '3', '4', '5']
```

- Accès à l'élément  $n$  :

```
>>> print(a[2])  
3
```

- Accès à tous les éléments **à partir de**  $n$  :

```
>>> print(a[2:])  
['3', '4', '5']
```

- Accès à tous les éléments sans les  $n$  derniers :

```
>>> print(a[:-3])  
['1', '2']
```

# Python / Listes / QCM 34

- Soit le code suivant :

```
a = ['7', '8', '2', '0']
```

- Comment ajouter un nouvel élément après le dernier de la liste **a** ?



```
a.append('7')
```



```
a[4] = '7'
```



```
a = a + '7'
```

# Python / Listes / QCM 34

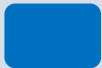
- Soit le code suivant :

```
a = ['7', '8', '2', '0']
```

- Comment ajouter un nouvel élément après le dernier de la liste **a** ?



```
a.append('7')
```



```
a[4] = '7'
```



```
a = a + '7'
```

# Python / Listes / QCM 34

```
a = ['7', '8', '2', '0']
```

```
a[4] = '7'
```

```
>>> a[4] = '7'  
IndexError: list assignment index out of range
```

```
a = a + '7'
```

```
>>> a = a + '7'  
TypeError: can only concatenate list (not "str") to  
list
```

```
>>> a = a + ['7']
```



# Python / Listes / QCM 35

- Soit le code suivant :

```
for k in range(3):  
    print(k)
```

- Quelle suite de nombres affiche ce code ?



0 1 2 3



0 1 2



Aucune

# Python / Listes / QCM 35

- Soit le code suivant :

```
for k in range(3):  
    print(k)
```

- Quelle suite de nombres affiche ce code ?

<input type="radio"/>	0 1 2 3
<input checked="" type="radio"/>	0 1 2
<input type="radio"/>	Aucune

# Python / Listes / QCM 36

- Soit le code suivant :

```
for k in range(1 ,10 ,3):  
    print(k)
```

- Quelle suite de nombres affiche ce code ?



1 4 7 10



1 2 4 5 6 7 8 9



1 4 7

# Python / Listes / QCM 36

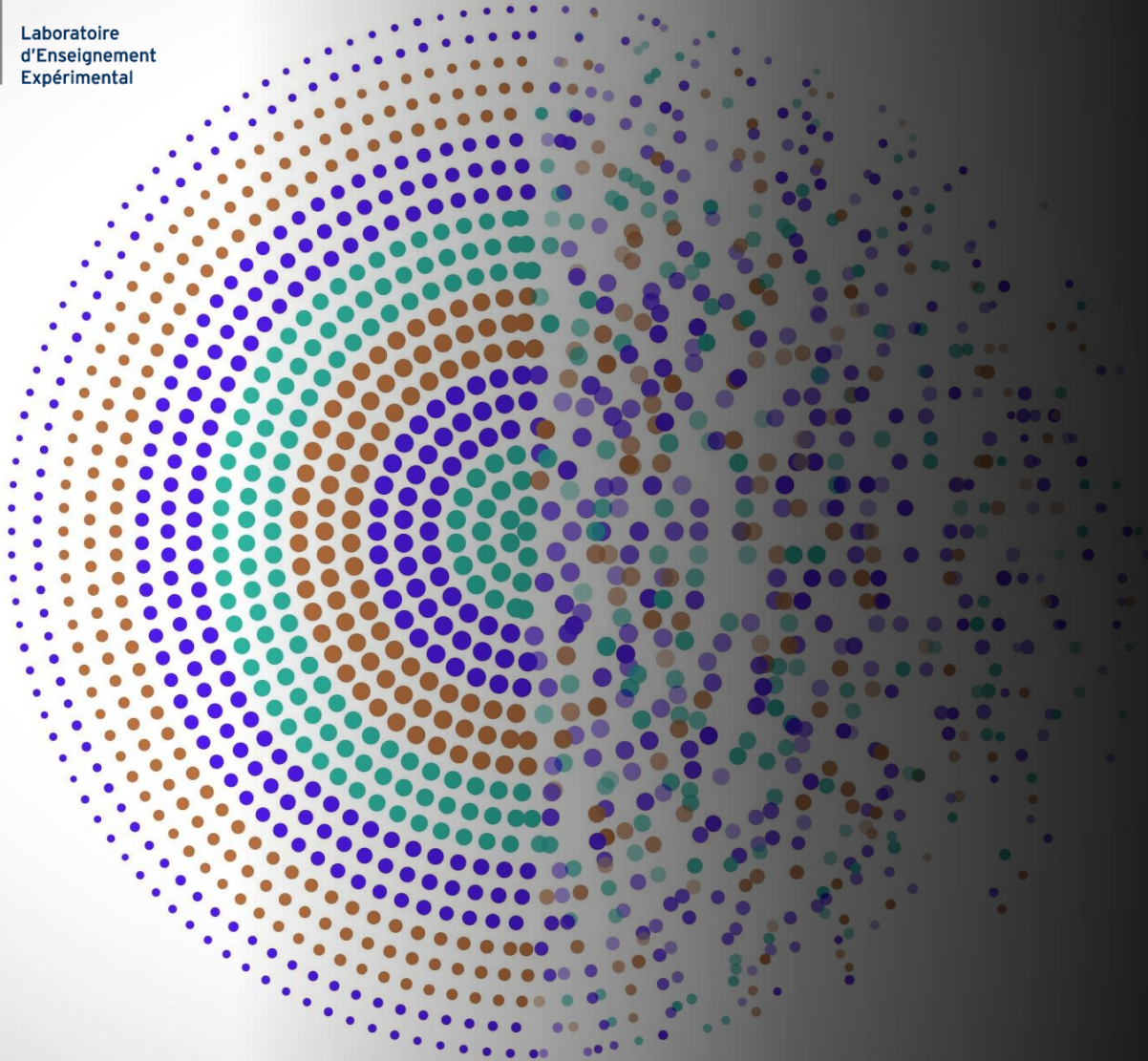
- Soit le code suivant :

```
for k in range(1 ,10 ,3):  
    print(k)
```

- Quelle suite de nombres affiche ce code ?

	<b>1 4 7 10</b>
	<b>1 2 4 5 6 7 8 9</b>
	<b>1 4 7</b>

▶



# Python / Fonctions

---

Outils Numériques / Semestre 5

# Python / Fonctions / QCM 41

- Soit le code suivant :

```
def mystere(a, b):  
    return 4*(a+b)
```

- Comment faire appel à cette fonction correctement ?



```
mystere(2, 5)
```



```
print(mystere('2', '4'))
```



```
a=5  b=2  mystere()
```

# Python / Fonctions / QCM 41

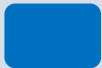
- Soit le code suivant :

```
def mystere(a, b):  
    return 4*(a+b)
```

- Comment faire appel à cette fonction correctement ?



```
mystere(2, 5)
```

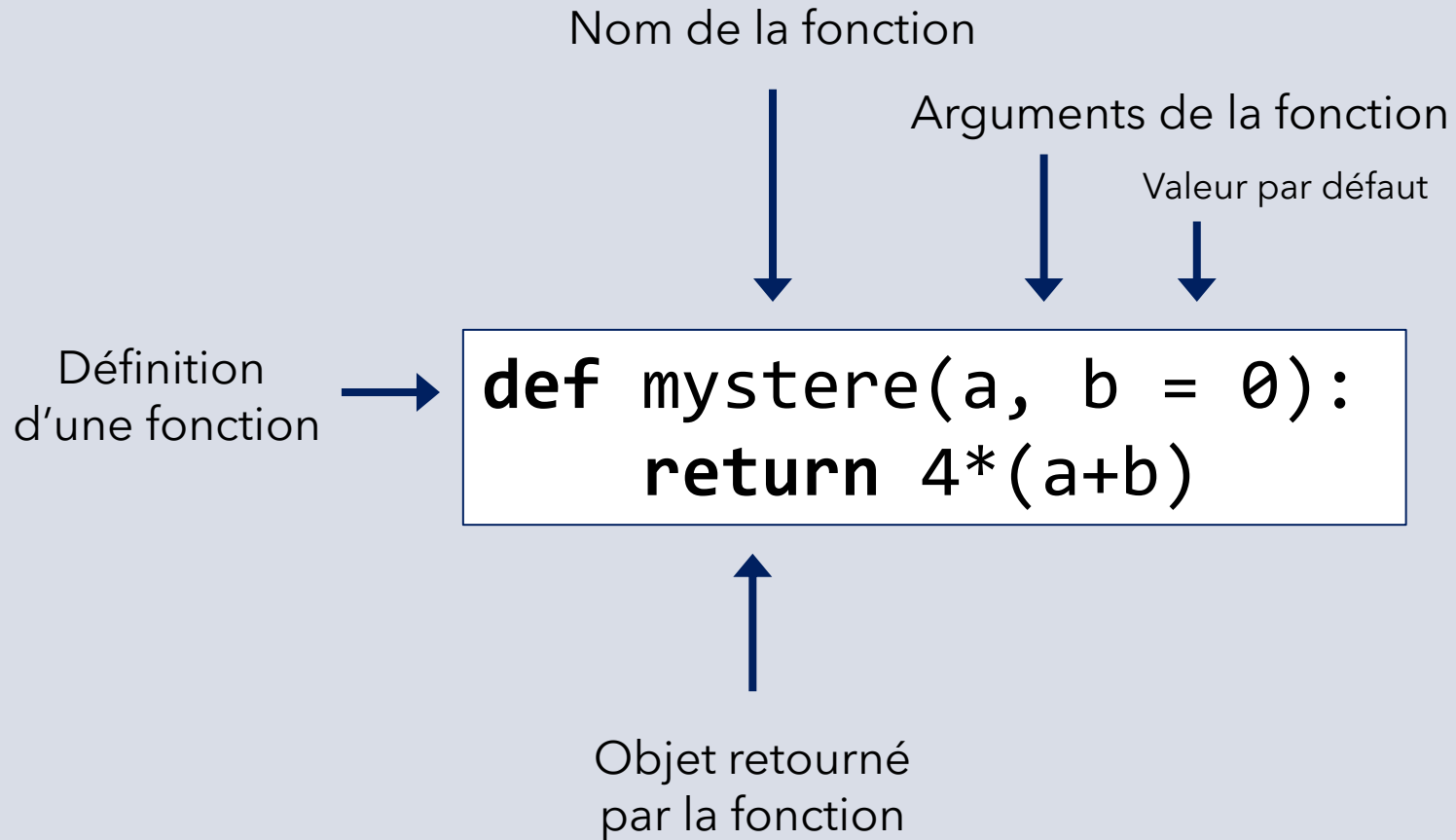


```
print(mystere('2', '4'))
```



```
a=5  b=2  mystere()
```

# Python / Fonctions



```
>>> c = mystere(1, 3)  
>>> print(c)  
16
```

```
>>> c = mystere(1)  
>>> print(c)  
4
```



# Python / Fonctions / QCM 41

- Soit une fonction qui prend en paramètre un entier  $n$  et renvoie **True** si cet entier est pair, et **False** si cet entier est im

```
def f(n):  
    if n%2 == 0:  
        return True  
    else:  
        return False
```

- Que peut-on reprocher à cette fonction ?



La fonction est mal nommée



Il manque la documentation



Le corps est mal écrit

# Python / Fonctions / QCM 41

- Soit une fonction qui prend en paramètre un entier  $n$  et renvoie **True** si cet entier est pair, et **False** si cet entier est impair
- Que peut-on reprocher à cette fonction ?

```
def f(n):  
    if n%2 == 0:  
        return True  
    else:  
        return False
```



La fonction est mal nommée



Il manque la documentation



Le corps est mal écrit

# Python / Fonctions / QCM 41

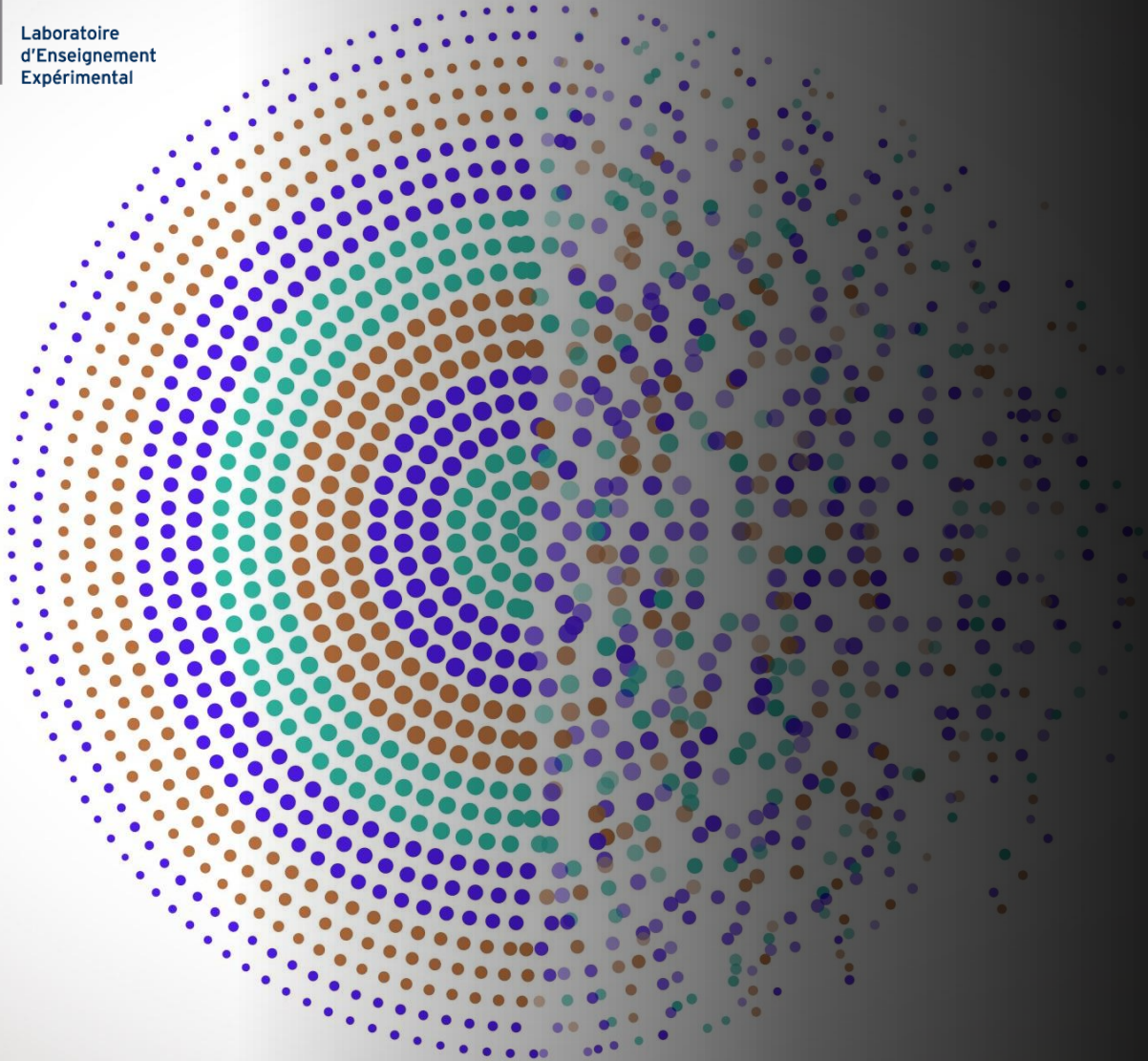
```
def est_pair(n:int):  
    """ renvoie True si n est pair, False  
    si n est impair  
    n -- entier  
    >>> est_pair(0)  
    True  
    >>> est_pair(101)  
    False  
    """  
    return n%2 == 0
```

```
def f(n):  
    if n%2 == 0:  
        return True  
    else:  
        return False
```

La fonction est mal nommée

Il manque la documentation

Le corps est mal écrit



# Python / Autres

---

Outils Numériques / Semestre 5

# Python / Dictionnaires / QCM 51

- Soit le code suivant :

```
d1={'x': 'w', 'k': 's', 'j': 'k', 'r': 'u'}  
d2={'k': 'n', 'u': 'y', 's': 'f', 'w': 'g'}  
c=d2[d1['r']]
```

- Que vaut la variable **c** ?



'g'



'r'



'y'

# Python / Dictionnaires / QCM 51

- Soit le code suivant :

```
d1={'x': 'w', 'k': 's', 'j': 'k', 'r': 'u'}  
d2={'k': 'n', 'u': 'y', 's': 'f', 'w': 'g'}  
c=d2[d1['r']]
```

- Que vaut la variable **c** ?

