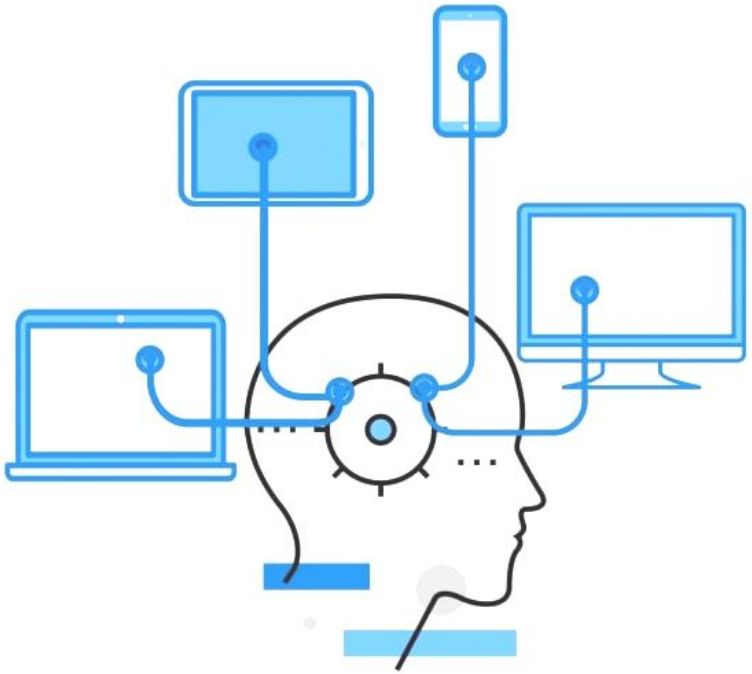


Image: pngtree

Option IA & Programmation

Cycle 2: Architecture d'un projet

Alexandre Mazel

Année 2021-2022

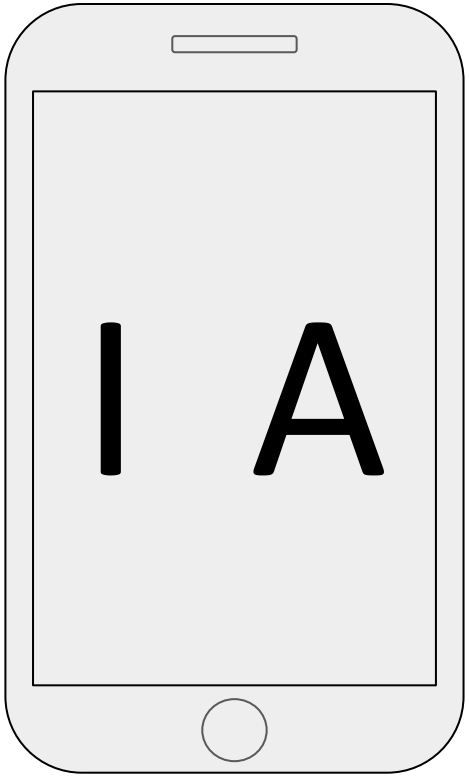
alexandre.zelma@gmail.com

But de cette option

I A

- Comprendre ce qui se cache derrière les fameuses lettres I et A
- Démystifier cette technologie par la pratique
- Réfléchir à:
  - Ce que cela fait,
  - ne fait pas (encore?),
  - les risques humains que cela pourrait engendrer.

Programme de l’année



Découvrir et expérimenter les bases de l’IA et de sa programmation.

5 cycles:

1. Bases de l’algorithmie
2. Architecture d’un projet
3. Intelligence Artificielle
4. Expérimentation et amélioration autour d’un projet par petits groupes
5. Continuation du projet

Contenu du cycle 2

Cycle 2: Découpage et architecture d'un projet: fonction, classes et objets.

Comment créer des modules indépendant facile à interconnecter.

- Retour en détail sur les fonctions
- Concept d'objet
- Implémentation en python

Petit quizz des familles

3 petits quizz

Alexandre bascule sur:

https://www.mentimeter.com/app

Et lance OIA_fin_cycle1_la_suite puis OIA_fin_cycle1_quizz_1 et 2

Les élèves vont sur menti.com et rentre les numéros de quizz affichés à l'écran.

Go to **www.menti.com** and use the code **4516 3366**

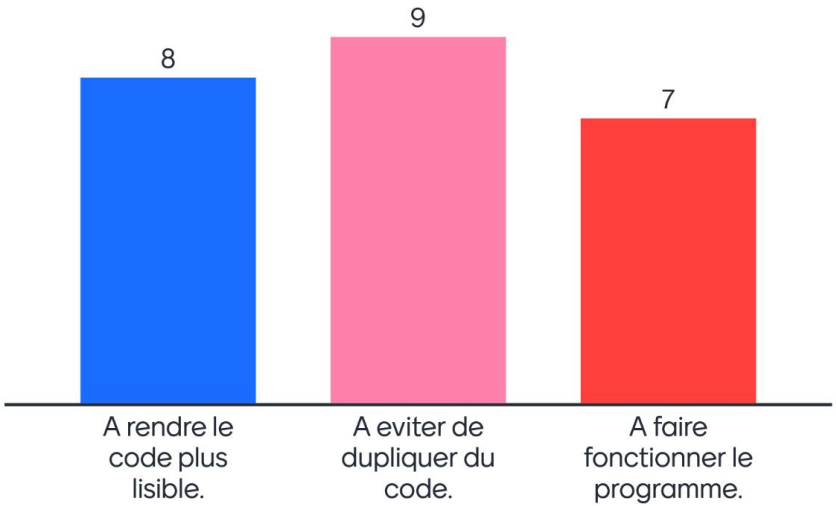
Qu'avez vous retenu du cycle précédent?

Mentimeter

Go to **www.menti.com** and use the code **9195 2768**

A quoi sert une fonction?

Mentimeter

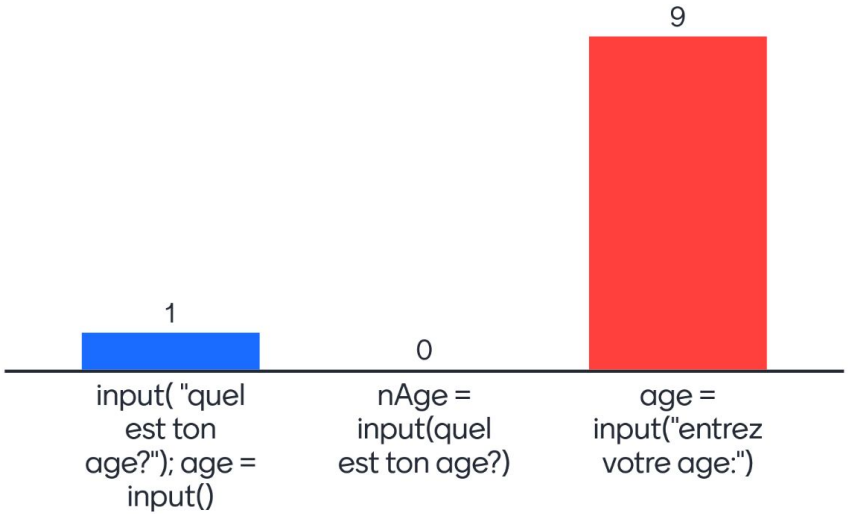


9

Go to **www.menti.com** and use the code **9195 2768**

Je veux demander à l'utilisateur de saisir son age:

Mentimeter

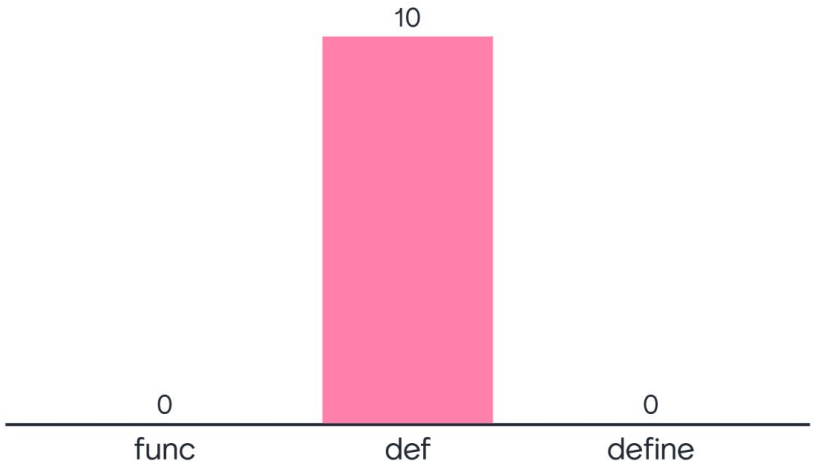


10

Go to **www.menti.com** and use the code **9887 1394**

Quel mot clé pour creer une fonction?

Mentimeter

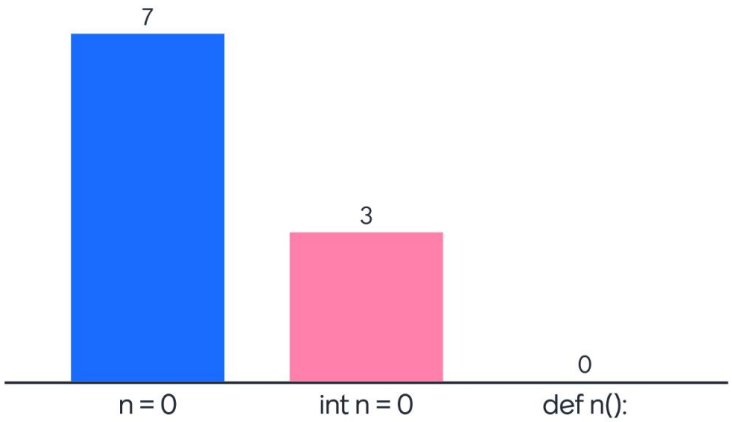


10

Go to **www.menti.com** and use the code **9887 1394**

Je veux creer une variable entiere pour compter une quantité

Mentimeter



10

Retour sur les fonctions

Les fonctions: mais pourquoi ?

credits: ebay.fr

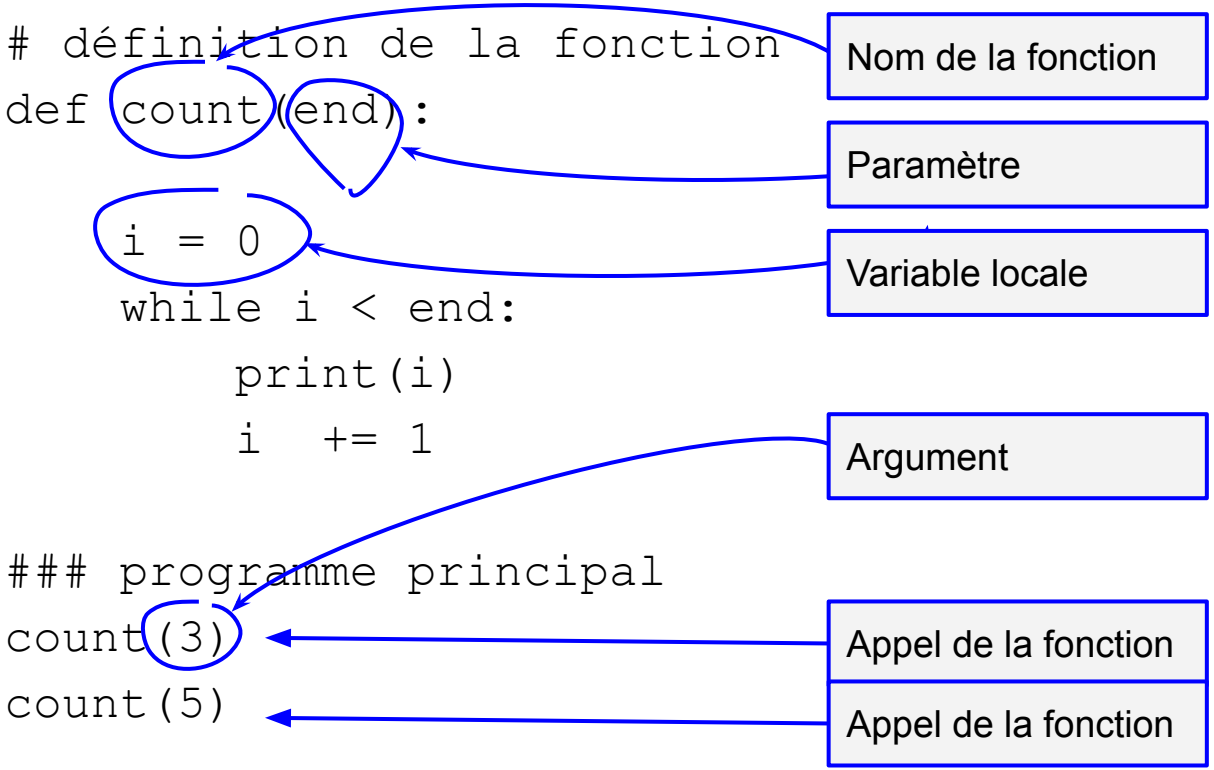
Les copier-coller c’est LE mal:

Avoir plus de 3 lignes identiques à 2 endroits, c’est moche.

Rappel

```
# définition de la fonction
def count(end):

    i = 0
    while i < end:
        print(i)
        i  += 1

### programme principal
count(3)
count(5)
```

Vocabulaire 1/2




```
# définition de la fonction
def count(end):

    i = 0
    while i < end:
        print(i)
        i  += 1

### programme principal
count(3)
count(5)
```

Vocabulaire 2/2

```
# définition de la fonction
def count(end = 3):

    i = 0
    while i < end:
        print(i)
        i  += 1
    return end

### programme principal
count()
count(5)
```

Valeur par défaut d'un paramètre

Ajout d'une valeur de retour

L'argument devient optionnel

Variable locale

```
def print_val():
    a = 2
    print("a=%d, b=%d" % (a,b))


### programme principal
a = 3
b = 5
print_val()
print("a=%d, b=%d" % (a,b))
```

Variable locale

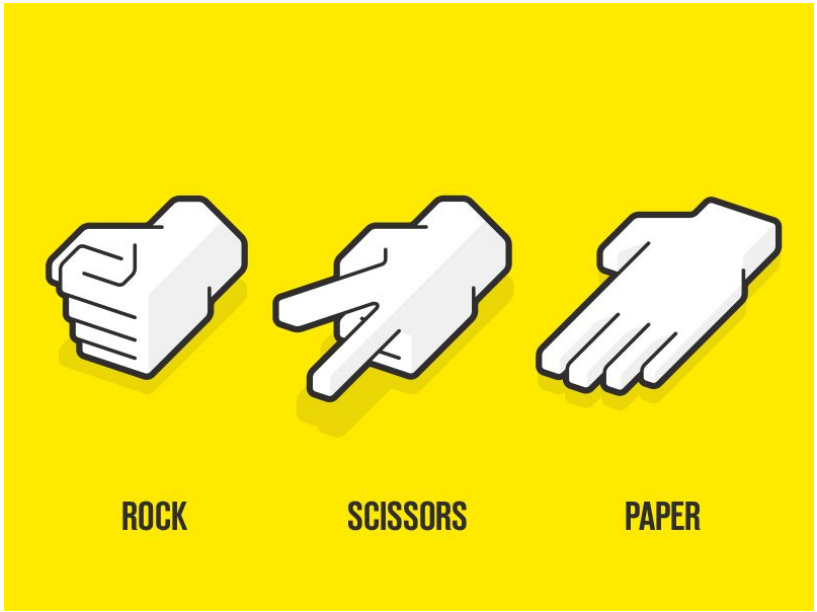

```
def print_val():
    a = 1
    print("a=%d, b=%d" % (a,b))


### programme principal
a = 3
b = 5
print_val()
print("a=%d, b=%d" % (a,b))
```

Variable locale

Variable globale

```
Résultat d'exécution
--------------------

C:\>Python local_variable.py
a=1, b=5
a=3, b=5
```

Rock Scissors Papper



Faire un rock scissors paper contre l'ordinateur.

L'ordinateur demande R, S ou P ?
Puis annonce ce qu'il a joué et qui gagne.
Il peut compter les points.

Fonctions:

- choose_cpu_play() => "R", "S" or "P"
- ask_human_play() => "R", "S" or "P"
- compute_winner(human,computer) => 1,0,-1
- run_new_round() => 1,0,-1
- launch_many_round(nbr_round) => score

credits: ?

rsp.py - ébauche

```
def choose_cpu_play():
      """Computer decide what to play
      Return “R”, “S” or “P”
      """
      ...

def ask_human_play():
      """Ask human what he want to play
      Return “R”, “S” or “P”
      """
      ...

def compute_winner(human_choice,cpu_choice):
      """Who win ?
      Return 1 if human win, -1 if cpu win, 0 for draw
      """
      ...

def run_new_round():
      """A complete round
      Return 1 if human win, -1 if cpu win, 0 for draw
      """
      c = choose_cpu_play()
      h = ask_human_play()
      score = ...
      ...
```

rsp.py - ébauche 2

```
def choose_cpu_play():
      """Computer decide what to play
      Return "R", "S" or "P"
      """
      if random.random()<0.33:
            return "R"
      if random.random()<0.5:
            return "S"
      return "P"
```

```
# autre methode:
a = random.random()
if a < 0.33:
      return "R"
if a < 0.66:
      return "S"
return "P"
```

```
def ask_human_play():
      """Ask human what he want to play
      Return "R", "S" or "P"
      """
      s = input("R for Rock, S for Scissors, P for
Paper?" )
      return s
```

```
def compute_winner(human_choice,cpu_choice):
      """Who win ?
      Return 1 if human win, -1 if cpu win, 0 for draw
      """
      ...


def run_new_round():
      """A complete round
      Return 1 if human win, -1 if cpu win, 0 for draw
      """
      c = choose_cpu_play()
      h = ask_human_play()
      score = ...
      ...
```

rsp.py - final

```
def choose_cpu_play():
      """Computer decide what to play
      Return "R", "S" or "P"
      """
      if random.random()>0.33:
            return "R"
      if random.random()>0.33:
            return "S"
      return "P"


def ask_human_play():
      """Ask human what he want to play
      Return "R", "S" or "P"
      """
      s = input("R for Rock, S for Scissors, P for
Paper?" )
      return s
```

```
def compute_winner(human,cpu):
      """Who win ?
      Return 1 if human win, -1 if cpu win, 0 for draw
      """
      if human == cpu:
            return 0
      if (human == "R" and cpu == "S")
            or (human == "S" and cpu == "P")
            or (human == "P" and cpu == "R")
            return 1
      return -1

def run_new_round():
      """A complete round
      Return 1 if human win, -1 if cpu win, 0 for draw
      """
      c = choose_cpu_play()
      h = ask_human_play()
      score = compute_winner(h,c)
      return score

def launch_many_round(nbr_round):
      score = 0
      for i in range(nbr_round):
            score += run_new_round()

launch_many_round(2)
```

Concept d'objets

Définition

La **programmation orientée objet (POO)**, ou **programmation par objet**, est une technique de programmation informatique.

Elle consiste en la définition et l'interaction de briques logicielles appelées *objets* ;

un objet représente:

- un concept,
- une idée
- ou toute entité du monde physique, comme:
  - une voiture,
  - une personne
  - ou encore une page d'un livre.

Il possède:

- une structure interne: ses variables
- un comportement par le biais de ses méthodes

Détail de la structure

L'objet est constitué de:

1 - Variables membres:

- Elles ne sont pas globales:
- Elles sont propres à chaque objet
- Elles décrivent son état
- On ne devrait pas pouvoir les modifier hors de l'état

2 - Méthodes

- Des fonctions qu'on applique à chaque objet
- C'est par ce biais là qu'on peut modifier l'état de l'objet
- L'utiliser, le faire "vivre".

Concept

Il s'agit donc de représenter ces objets et leurs relations ;

l'interaction entre les objets via leurs relations permet de concevoir et réaliser les fonctionnalités attendues, de mieux résoudre le ou les problèmes.

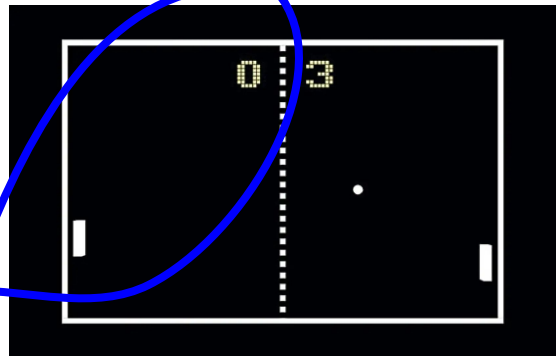
Dès lors, l'étape de modélisation revêt une importance majeure et nécessaire pour la Programmation Orientée Objet (POO).

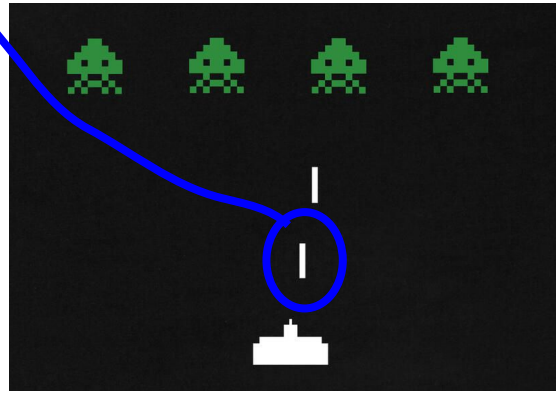
C'est elle qui permet de transcrire les éléments du réel sous forme virtuelle.

Exemple d’objet

Trouvons ensemble les données et méthodes de ces types d’objets représentant des concepts de la vie réelle.

- Voiture
- Ecole
- Encyclopédie
- Joueur humain d’un jeu vidéo de type pong
- Projectile d’un jeu vidéo de type space invaders





Exemple d’objet

Car Object

Members:

- name
- modele
- max_speed
- consumption
- current_gaz
- current_gear
- is_brake_engaged

Methods:

- Unlock()
- Start()
- Accelerate()
- ChangeGear(num)
- Brake()
- TurnLeft(degree)
- TurnRight(degree)
- TurnLight(b_on_or_off)

Exemple d’objet

School Object

Members:

- name
- year_of_build
- address
- list_rooms
- list_teachers
- opening_hour
- list_students_by_level

Methods:

- StartFireAlarm()
- SendEmailsToOneLevel(num_level)
- LaunchPayRoll()
- GetAllMenu()

Exemple d’objet

SchoolRoom Object

Members:

- name
- number_of_seat
- calendar
- has_video_projector

Methods:

- GetNumberOfSeat()
- Book(date)
- GetNextUseDate()

Exemple d’objet

SchoolTeacher Object

Members:

- name
- year_of_birth
- address
- subject
- list_classes

Methods:

- SendEmail(blabla)
- GetAllGrades(class)

Exemple d’objet

Encyclopedia Object

Members:

- name
- update_date
- list_definitions

Methods:

- LoadFromDisk(file)
- FindDefinition(word)
- PrintDefinition(definition)
- UpdateDefinition(word,definition)
- PrintBook()

Exemple d’objet

HumanPlayer Object

Members:

- name
- score
- racket_position
- key_for_up
- key_for_down

Methods:

- Update()
- IsUp()
- IsDown()
- UpdateRacket()
- Win()

Exemple d’objet

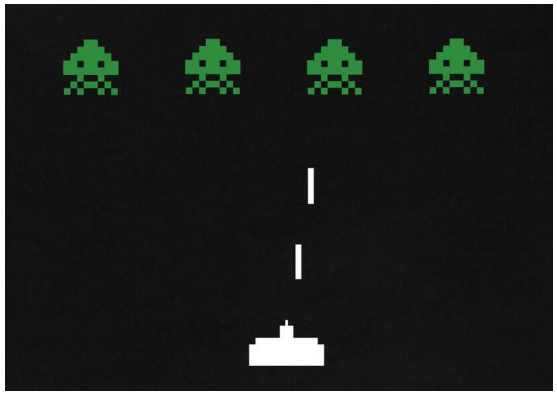
Ball Object

Members:

- position
- speed
- time_to_live
- damage

Methods:

- Update()
- IsOutOfScreen()
- GetDamage()
- IsExpired()



Implémentation d'objets en python

Exemple de codage d'une encyclopédie

```
def chargeDepuisLeDisque():
    …
    return ...gros_dict…

def chercheDansLaBase(mot,base):
    …
    return "une super definition"

definitions = {}
annee_creation = "2002"

definitions = chargeDepuisLeDisque()

defi = \
chercheDansLaBase("arbre",definitions)

print("Voici la définition d'un %s: \
%s" % ("arbre",definition) )
```

```
class Encyclopedie:
    def __init__(self):
        self.dictAll = {}
        self.annee_creation = 2002

    def charge(self):
        …
        self.dictAll = ...gros_dict..

    def cherche(self):
        …

    def printDef(self,mot):
        defi = self.cherche(mot)
        print("Voici la définition d'un %s: %s" % (mot,defi)
)


# creation de l'objet
e = Encyclopedie()

# Appel de ses méthodes
e.charge()  # ~~ charge(e)
e.printDef("arbre") # ~~ printDef(e,"arbre")
```

credits: facebook

Chrono

credits: wikipedia/Joe Haupt

En utilisant la POO, programmer un chronomètre pour mesurer le temps entre 2 appui sur entrée

Fonctions:

- Reset()
- Start()
- Stop()
- GetElapsedTime()

Advanced:

- Pause()
- StartNewLaps()
- GetAllLapsDuration() => liste des temps pour chaque tour
- GetAverageTimePerLaps()

Rappel:

- un objet est défini en utilisant le mot clé “class”
- time.sleep(0.1) => fait une pause de 100ms

chrono.py - Solution

```
import time

class Chrono:
      def __init__( self ):
            self.start_time = None
            self.stop_time = None

      def Start( self ):
            self.start_time = time.time()

      def Stop( self ):
            if self.start_time == None:
                  print("ERR: launch start first")
                  return
            self.stop_time = time.time()

      def GetElapsedTime( self ):
            return self.stop_time - self.start_time

def auto_test()
      c = Chrono()
      c.Start()
      time.sleep(1.)
      c.Stop()
      print(c.GetElapsedTime())
      assert(c.GetElapsedTime()>=1 and c.GetElapsedTime()<1.5)

if __name__ == "__main__":
      auto_test()
```

```
# Réutilisation dans un autre programme

import chrono # ne va pas lancer auto_test
import time

mon_chrono = chrono.Chrono()
mon_chrono.Start()
time.sleep(2)
mon_chrono.Stop()
print(mon_chrono.getElapsedTime())
```

```
# Réutilisation dans un autre programme
vaguement plus utile

import chrono # ne va pas lancer auto_test

mon_chrono1 = chrono.Chrono()
mon_chrono2 = chrono.Chrono()
mon_chrono1.Start()
dummy=input("enter the word 'cheval':")
mon_chrono1.Stop()

mon_chrono2.Start()
print("temps: " + mon_chrono.getElapsedTime())
mon_chrono2.Stop()
```

chrono.py - Solution Advanced

```
import time

class Chrono:
      def __init__( self ):
            self.start_time = None
            self.stop_time = None

      def Start( self ):
            self.start_time = time.time()

      def Stop( self ):
            if self.start_time == None:
                  print("ERR: launch start first")
                  return
            self.stop_time = time.time()

      def GetElapsedTime( self ):
            return self.stop_time - self.start_time

def auto_test()
      c = Chrono()
      c.Start()
      time.sleep(1.)
      c.Stop()
      print(c.GetElapsedTime())
      assert(c.GetElapsedTime()>=1 and c.GetElapsedTime()<1.5)

if __name__ == "__main__":
      auto_test()
```

Bpm

En utilisant la POO et en réutilisant la classe chrono de chrono.py de l'exercice précédent y compris la fonction GetAverageTimePerLaps.
Faire un programme bpm.py qui donne l'estimation du bpm (beat per minute) d'une musique en tapant le rythme sur la touche entrée.

Recette:

1. Je lance une chanson avec un autre logiciel
2. Je lance mon programme
3. Je tape le rythme avec mon doigt sur la touche entrée.
4. Au bout de 4 appuis, je demande a chrono le temp moyen entre 2 appuis.
5. Je programme une formule mathématique compliquée, mais pas trop en fait, pour convertir en bpm.

credits: wikipedia/Joe Haupt

chrono.py - Solution

```
import time

class Chrono:

	def __init__( self ):
		self.reset()

	def reset( self ):
		self.start_time = None
		self.stop_time = None
		self.start_laps_time = None
		self.list_laps_time = []

	def start( self ):
		self.start_time = time.time()
		self.start_laps_time = self.start_time

	def stop( self ):
		if self.start_time == None:
			print("ERR: launch start \
				before calling stop !")
			return
		self.start_new_laps()
		self.stop_time = time.time()

	def get_elapsed_time( self ):
		return self.stop_time - self.start_time
```

```
	def start_new_laps( self ):
		"""
		will measure the time since start
		or since last call of this function
		"""
		laps_time = time.time() - self.start_laps_time
		if laps_time >= 0.0001: # prevent case stopped just
					after start_new_laps
			self.list_laps_time.append(laps_time)
			self.start_laps_time = time.time()

	def get_all_laps_duration(self):
		return self.list_laps_time

	def get_average_time_per_laps( self ):
		sum = 0
		for t in self.list_laps_time:
		sum += t
		return sum/len(self.list_laps_time)
		# or simplier:
		# get_elapsed_time()/len(self.list_laps_time)

def auto_test():
		c = Chrono()
		c.start()
		for i in range(4):
			time.sleep(1.)
			c.start_new_laps()
		c.stop()
		print(c.get_all_laps_duration())
		print(c.get_average_time_per_laps())
		assert(c.get_average_time_per_laps()>=1 and
		c.get_average_time_per_laps()<1.5)
if __name__ == "__main__":
		auto_test()
```

bpm.py - Solution

```
import chrono

def measure_bpm():
    c = chrono.Chrono()
    input("appui sur entree pour commencer...")
    c.start()
    for i in range(4):
        input("appui sur entree encore %d fois:" % (4-i))
        c.start_new_laps()
    c.stop()
    print(c.get_all_laps_duration())
    print(c.get_average_time_per_laps())
    print("bpm: %5.2f" % (60/c.get_average_time_per_laps()))


measure_bpm()
```

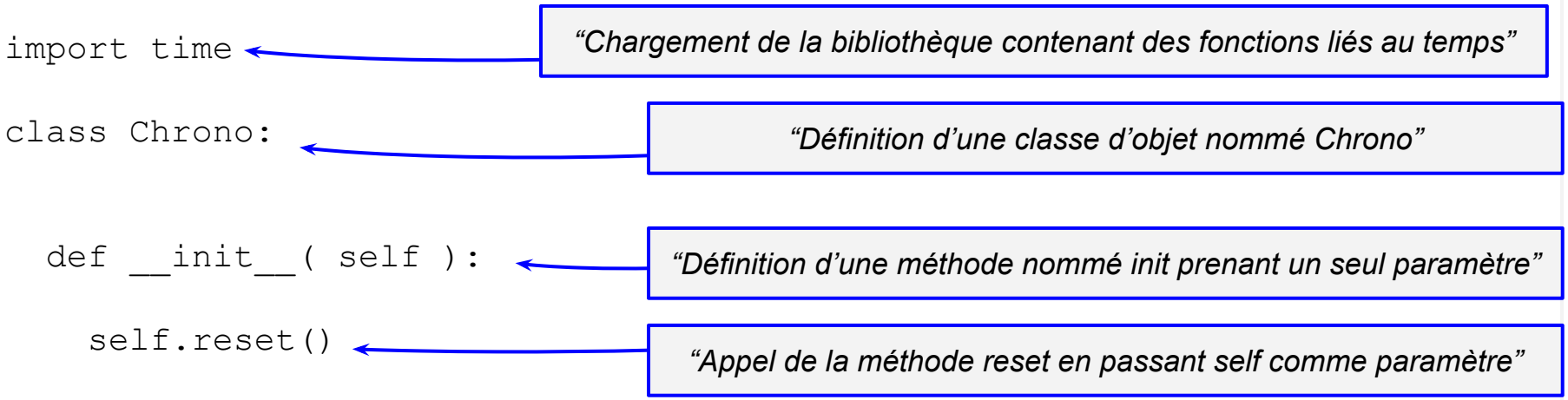
Révision

Revoir la solution précédente et chaque élève à tour de rôle explique en francais ce que fait chaque ligne.

```
import time

class Chrono:


  def __init__( self ):

    self.reset()

  ...
```

Révision

Revoir la solution précédente et chaque élève à tour de rôle explique en francais ce que fait chaque ligne.



Révision 2

Ecrire une fonction tous ensemble de manière interactive.

Création d'une fonction qui prend une chaîne de caractère en paramètre

- Affichage d'un caractère sur deux.
- Compter le nombre de e.
- E devient une lettre quelconque en paramètre.
- Rendre le programme insensible à la casse avec un paramètre d'option.
    - Différents manière de faire le test (sur une ligne, en plusieurs bouts…)

Entrainement au réflexe

credits: permisecole.com

On veut entraîner un conducteur à aiguiser ses réflexes.

Réutiliser la classe chrono pour faire un programme qui attend entre 0 et 4s puis demande au conducteur de freiner en appuyant sur la touche entrée.

On affiche ensuite le temps de réaction.

Advanced:
Pour éviter les tricheries en remplissant le buffer du clavier de touches entrée, utiliser les méthodes kbhit et getch de la bibliothéque windows msvcrt.

(j'espère qu'elle est installée sur votre machine)

car_game.py

```
import chrono
import msvcrt
import random
import time

def game_car():
     c = chrono.Chrono()
     time.sleep(random.random()*4)
     while msvcrt.kbhit():
          # Only if there's a keypress waiting do we get it with getch()
          print("Key hit: %s" % (msvcrt.getch()))
     c.start()
     input("La voiture de devant freine, vite freine toi aussi, en pressant enter!")
     c.stop()
     print("Temps pour freiner: %.2fs" % c.get_elapsed_time())


while 1:
     game_car()
```

Entrainement au réflexe v2

En continuant d'utiliser msvcrt.getch varier les touches selon le type d'obstacles

- g => gauche,
- d => droite,
- f => freine

credits: permisecole.com

Des questions ?

???

Tu peux me télécharger ici:

http://engrenage.studio/oia/OIA_2021_2022_Cycle2.pdf

En bonus: qrcode.py

```
import qrcode # pip3 install qrcode

data = "http://engrenage.studio/oia/OIA_2021_2022_Cycle2.pdf"

filename = "cyclex.png"

img = qrcode.make(data)

img.save(filename)
print("SUCCESS: QRCode written to file " + filename )

# et si on l'affichait juste pour voir, ca serait plus poli.

import cv2

img = cv2.imread(filename)

cv2.imshow("result: " + filename,img)

cv2.waitKey(0)
```