

Selenium

—

Mouna LEMSAADI
Zineb ELKARKOURI

Selenium

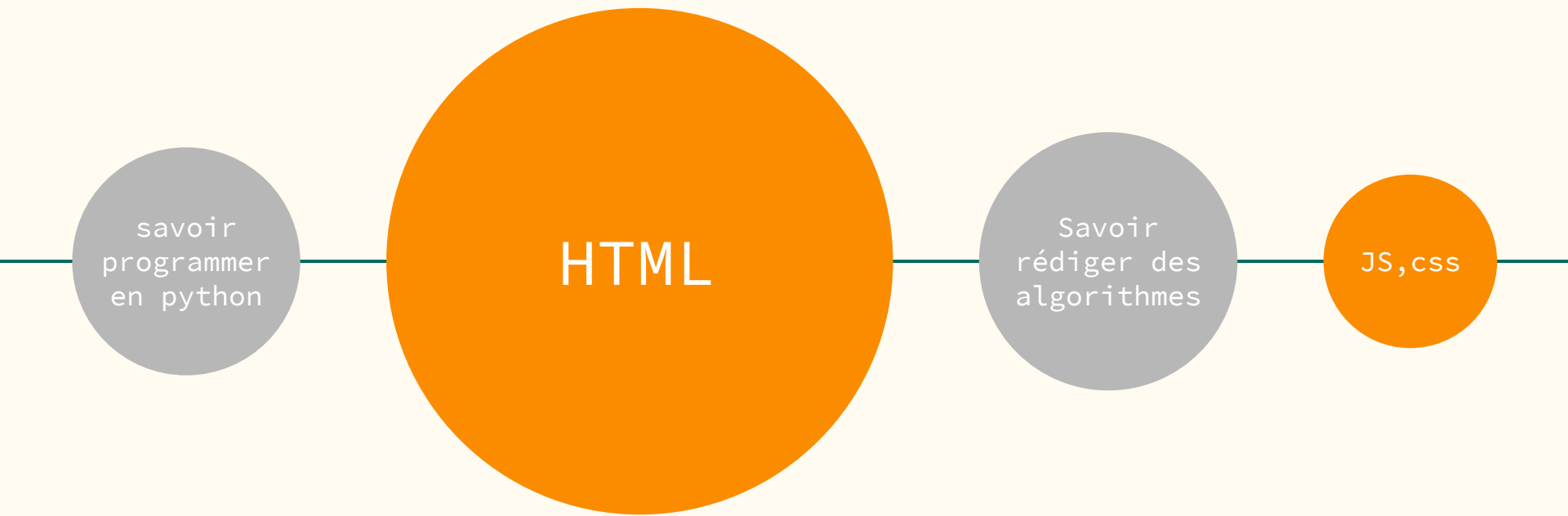
Selenium est un framework de test informatique développé en Java.

Il permet d'interagir avec différents navigateurs web tel que *Google Chrome* grâce au *chromedriver* ou *Mozilla Firefox* avec *Gecko* de même que le ferait un utilisateur de l'application. Il entre ainsi dans la catégorie des outils de test dynamique (à l'inverse des tests statiques qui ne nécessitent pas l'exécution du logiciel) facilitant le test fonctionnel.

Languages

- Java
- Python
- Ruby
- .NET
- Perl
- Rust
- Go

Pré-requis



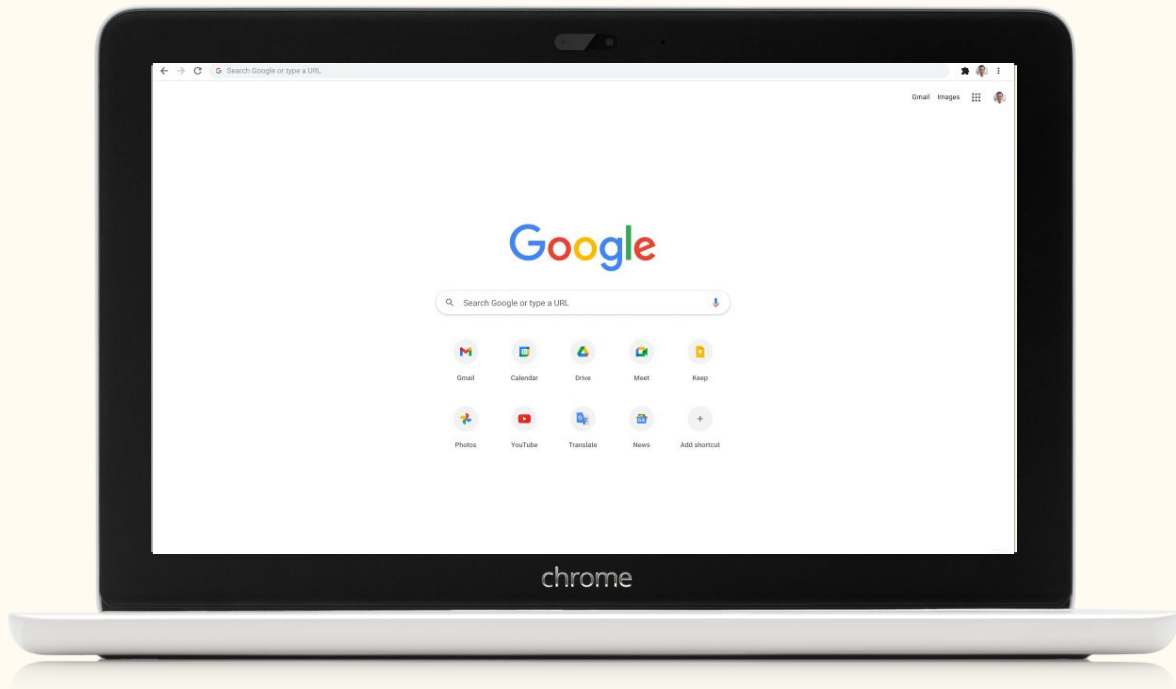
Installer Selenium

—

ChromeDriver

Google Chrome

Pour utiliser Selenium, il faut installer un navigateur comme Google Chrome, ainsi que ChromeDriver.



Vérifier la version du navigateur

Avant d'installer "Google driver", il faut s'assurer de télécharger la bonne version.(Veuillez appuyer sur "à propos" pour vérifier la version du navigateur chrome installé)

Lien d'installation:

<https://sites.google.com/a/chromium.org/chromedriver/downloads>



Google Chrome



Google Chrome est à jour

Version 91.0.4472.77 (Build officiel) (x86_64)

Mettre à jour Chrome automatiquement pour tous les utilisateurs [En savoir plus](#)

Obtenir de l'aide avec Chrome

Signaler un problème

ChromeDriver - WebDriver for Chrome

CHROMEDRIVER

CAPABILITIES & CHROME OPTIONS

CHROME EXTENSIONS

CHROMEDRIVER CANARY

CONTRIBUTING

▼ DOWNLOADS

VERSION SELECTION

▼ GETTING STARTED

ANDROID

CHROME OS

▼ LOGGING

PERFORMANCE LOG

MOBILE EMULATION

▼ NEED HELP?

CHROME DOESN'T START OR CRASHES
IMMEDIATELY

CHROMEDRIVER CRASHES

Downloads

Current Releases

- If you are using Chrome version 91, please download [ChromeDriver 91.0.4472.19](#)
- If you are using Chrome version 90, please download [ChromeDriver 90.0.4430.24](#)
- If you are using Chrome version 89, please download [ChromeDriver 89.0.4389.23](#)
- If you are using Chrome version 88, please download [ChromeDriver 88.0.4324.96](#)
- For older version of Chrome, please see below for the version of ChromeDriver that supports it.







If you are using Chrome from Dev or Canary channel, please following instructions on the [ChromeDriver Canary](#) page.

For more information on selecting the right version of ChromeDriver, please see the [Version Selection](#) page.

ChromeDriver 91.0.4472.19

Supports Chrome version 91

Index of /90.0.4430.24/

	<u>Name</u>	Last modified	Size	ETag
	Parent Directory		-	
	chromedriver linux64.zip	2021-03-15 16:49:46	5.53MB	ff32297377308392f3e5b44cf282f77a
	chromedriver mac64.zip	2021-03-15 16:49:48	7.68MB	01378f44ca91150771859e254809fb66
	chromedriver mac64 ml.zip	2021-03-15 16:49:50	7.01MB	9cd97b08730a9d395610d051b4aa2c05
	chromedriver win32.zip	2021-03-15 16:49:51	5.67MB	eeb5e37fc4d4b21337a46576137a2053
	notes.txt	2021-03-15 16:49:56	0.00MB	a79b03d7895fbb145c4d3d0a63ba0d41

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time
PATH="Downloads/chromedriver"
driver=webdriver.Chrome(PATH)
driver.get("https://fr.wikipedia.org/wiki/Wikip%C3%A9dia:Accueil_principal")
link=driver.find_element_by_link_text("Sommaire de l'aide")
link.click()
try:
    element=WebDriverWait(driver,10).until(
        EC.presence_of_element_located((By.LINK_TEXT,"Espace Aide"))
    )
    element.click()
    element1=WebDriverWait(driver,10).until(
        EC.presence_of_element_located((By.ID,"n-thema"))
    ).click()
except:
    driver.quit()
```

TESTCASE: Tester le site (python.org) si ça marche

main.py:

setUp(): Cette méthode permet de lancer le chromeDriver ainsi que l'accès à au site du python

```
def setUp(self):
    self.driver = webdriver.Chrome(ChromeDriverManager().install())
    self.driver.get("http://www.python.org")
```

test_search_python(): Cette méthode de test est lancée après setUp et qui retourne une valeur indiquant si les éléments testés fonctionnent bien ou pas

```
def test_search_python(self):
    print("Test succée")
    mainPage = page.MainPage(self.driver)
    assert mainPage.is_title_matches()
    mainPage.serach_text_element = "pycon"
    mainPage.click_go_button()
    search_result_page = page.SearchResultsPage(self.driver)
    assert search_result_page.is_results_found()
```

tearDown(): cette méthode est lancée dès que le test est terminé (on quitte le navigateur Chrome)

```
def tearDown(self):
    self.driver.close()
```

TESTCASE: Tester le site (python.org) si ça marche

page.py:

Visé à créer un objet pour chaque partie d'une page web. Cette technique permet de créer une séparation entre le code de test et le code réel qui interagit avec la page web.

```
class SearchTextElement(BasePageElement):  
    locator = 'q'
```

is_title_matches(): cette méthode va chercher dans le titre du site est-ce qu'il contient le mot "python"

```
def is_title_matches(self):  
    return "Python" in self.driver.title  
  
def click_go_button(self):
```

click_go_button(): cette fonction va chercher le bouton "GO" puis va cliquer dessus et il va écrire aussi dans la barre de recherche

```
def click_go_button(self):  
    s=self.driver.find_element_by_name(SearchTextElement.locator)  
    s.send_keys("Python")  
    s.send_keys(Keys.RETURN)  
    element = self.driver.find_element(*MainPageLocators.GO_BUTTON)  
    element.click()
```

TESTCASE: Tester le site (python.org) si ça marche

element.py:

```
from selenium.webdriver.support.ui import WebDriverWait
class BasePageElement(object):

    def __set__(self, obj, value):

        driver = obj.driver
        WebDriverWait(driver, 100).until(
            lambda driver: driver.find_element_by_name(self.locator))
        driver.find_element_by_name(self.locator).clear()
        driver.find_element_by_name(self.locator).send_keys(value)

    def __get__(self, obj, owner):

        driver = obj.driver
        WebDriverWait(driver, 100).until(
            lambda driver: driver.find_element_by_name(self.locator))
        element = driver.find_element_by_name(self.locator)
        return element.get_attribute("value")
```


TESTCASE: Tester le site (python.org) si ça marche

Locator.py:


L'idée derrière ce programme est de stocker les ID des éléments du site pour les récupérer facilement

```
class MainPageLocators(object):  
    GO_BUTTON = (By.ID, 'submit')  
  
class SearchResultsPageLocators(object):  
    pass
```

Lancer la recherche “ensias” sur google

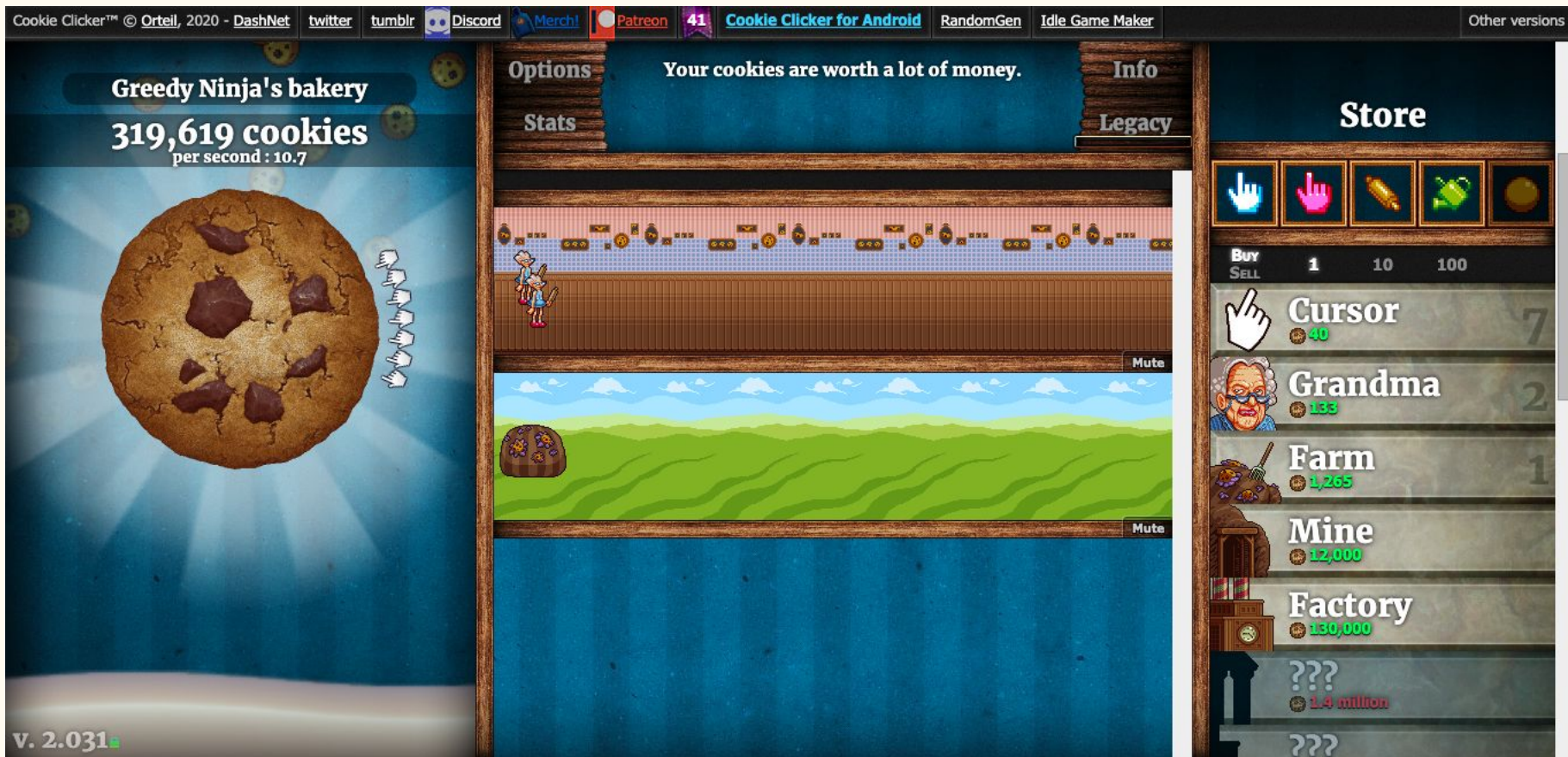
```
2  from selenium import webdriver
3  from selenium.webdriver.common.keys import Keys
4  import time
5
6
7  PATH="/Users/mounalemsaadi/desktop/ensias-miola/cours/Scrum/Selenium/chromedriver"
8  driver=webdriver.Chrome(PATH)
9
10
11
12  driver.get("https://google.com")
13  print(driver.title)
14  search= driver.find_element_by_name("q")
15  search.send_keys("ensias")
16  search.send_keys(Keys.RETURN)
17
18  driver.quit()
```

Atelier



Écrire un programme
python, qui clique d'une
façon continue sur un
cookie sur le jeu
cookieclicker et achète un
item à chaque fois que son
score en est suffisant.

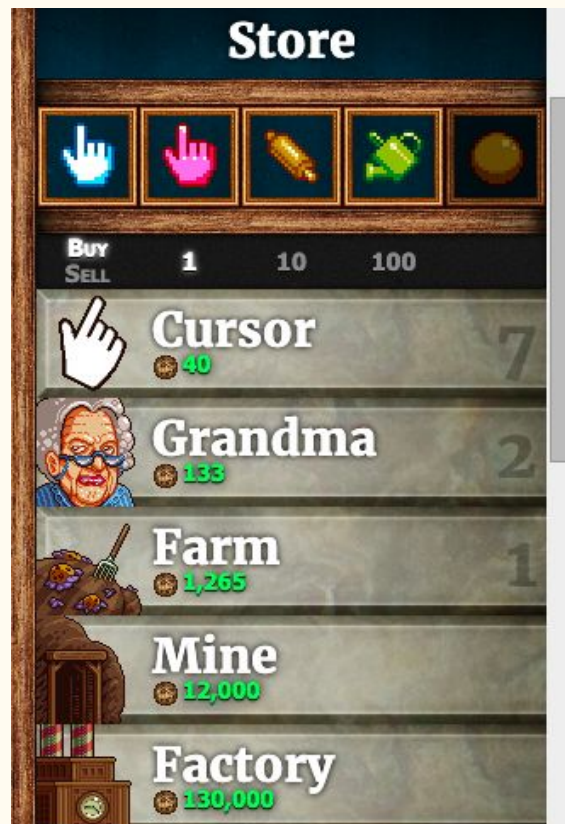
Le jeux cookieclicker



Le cookie

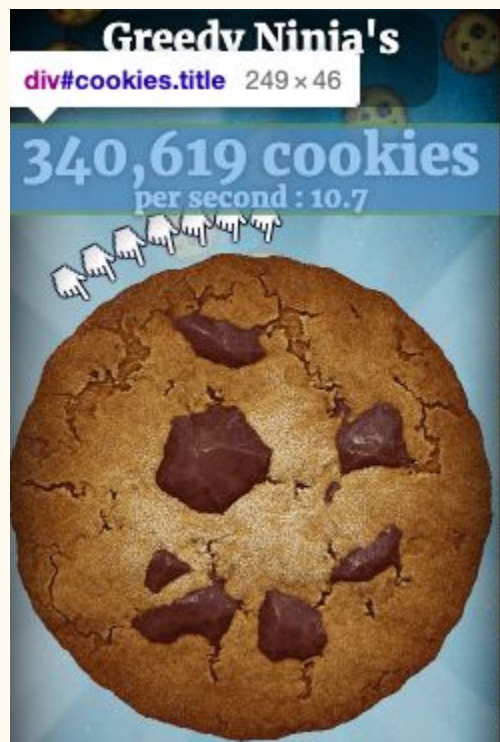


Store des items



Solution

Le cookie



```
<div class="blackGradient"></div>
<div id="sectionLeftInfo"></div>
** ▶ <div id="cookies" class="title" >...</div> == $0
    ▶ <div id="bakeryNameAnchor">...</div>
    <div id="specialPopup" class="framed prompt offScreen"></div>
    <div id="huffe" class="crateBox"></div>
```

L'item: Cursor



```
▼ <div class="content">
  <div class="lockedTitle">???
```

L'item : Grandma



```
▼<div class="content">
  <div class="lockedTitle">???
```

cookieclicker.py

appeler la méthode `get()` avec le lien du site

```
6  #Le chemin vers le chromedriver téléchargé
7  PATH="/Users/mounalemsaadi/desktop/ensias-miola/cours/Scrum/Selenium/chromedriver"
8  driver=webdriver.Chrome(PATH)
9
10 #Le lien vers le site cookieclicker
11 driver.get("https://orteil.dashnet.org/cookieclicker/")
```

cookieclicker.py

Affecter les éléments correspondants aux variables cookie (Le cookie à cliquer) et cookie_count (Le score gagné)

```
33  #Le cookie à cliquer dessus
34  cookie=driver.find_element_by_id("bigCookie")
35  #Le score
36  cookie_count=driver.find_element_by_id("cookies")
```

cookieclicker.py

Déclarer le tableau des item à acheter, et actions, le click sur le cookie

```
38  #L'item à acheter avec le score
39  items = [driver.find_element_by_id("productPrice"+str(i)) for i in range(1, -1, -1)]
40
41  actions = ActionChains(driver)
42  actions.click(cookie)
```

cookieclicker.py

- Cliquer sur le cookie
- comparer le score au prix de l'item
- Si le prix de l'item est inférieur au score gagné, le programme achète l'item

```
46 for i in range(5000):
47     actions.perform() #Cliquer sur le cookie
48     count = int(cookie_count.text.split(" ")[0])
49     print(count)
50     for item in items:
51         value=int (item.text)
52         if value<=count : # Si le prix de l'item est inférieur au score, acheter l'item
53             upgrade_actions = ActionChains(driver)
54             upgrade_actions.move_to_element(item)
55             upgrade_actions.click()
56             upgrade_actions.perform()
```

MERCI POUR VOTRE
ATTENTION

