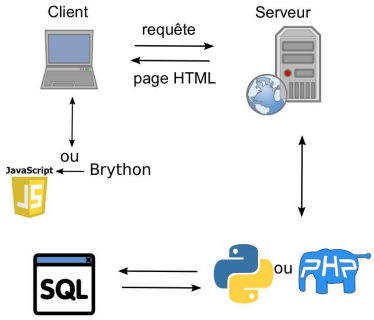
Cours issue de la documentation Brython
Pour débogguer vos script avec le navigateur :

- firefox : F12 ou Ctrl+Maj+k ou Outils-> Développement web->consol web
- chrome : F12

1 Introduction

1.1 Qui fait quoi ?



lorsque vous saisissez une URL dans votre navigateur, que vous validez cette dernière, votre navigateur envoie une **requête** au serveur concerné afin qu'il nous renvoie une page web.
On nomme l'échange de données entre votre navigateur et le serveur qui fournit les pages web un échange ***client / serveur***.
Le client représente votre navigateur.
Habituellement coté client c'est javascript qui est utilisé pour programmer. Il existe des librairies pour facilité la programmation ***javascript*** comme ***jquery, p5js***.
N'ayant pas le temps de vous apprendre à coder en javascript nous utiliseront ***Brython*** qui nous permet de coder en python.
La libraire Brython transforme ce code python en javascript (compréhensible par votre navigateur).

Vous devez bien faire faire la différence le programme exécuté coté client (javascript ou brython) et le programme exécuté coté serveur (php ou python).

1.2 Que peut on faire avec Brython ?

On peut faire la même chose qu'avec javascript

- Accéder aux DOM (Document Object Model) comme lire la couleur, lire un texte etc...
- Modifier le DOM (comme changer la couleur d'un texte, modifier un texte d'un paragraphe, etc...
- Réagir aux évènements de la souris, du clavier etc...

1.3 Installation chez vous avec pyzo.

➔ Lancer ***pyzo***.
On installe les fichiers nécessaires et on prépare le dossier. Dans la console exécutez :

```
pip install brython             # On installe brython
pip install brython --upgrade   # Pour seulement mettre à jour brython
mkdir test-brython              # mkdir pour make directory
cd test-brython                 # cd pour change directory
!python -m brython --install    # On prépare les fichier pour brython.
```

➔ Vous devez disposer alors des fichiers suivants :

- ***brython.js*** : le moteur Brython, à inclure dans la page HTML
- ***brython_stdlib.js*** : regroupe tous les fichiers de la distribution Python standard supportés par Brython

brython.js contient quelques modules très utilisés : ***browser, browser.html, javascript***.
Si votre application utilise des modules de la distribution standard, il faut inclure ***brython_stdlib.js*** en plus de brython.js :

```
<script type="text/javascript" src="./brython.js"></script>
<script type="text/javascript" src="./brython_stdlib.js"></script>
```

➔ Si vous avez un problème j'ai mis à disposition chez moi ces fichiers.
Télécharger les fichiers brython.js et brython_stdlib.js

1.4 Installation depuis le lycée avec pyzo

Évidemment au lycée on a quelques problèmes dû à la configuration.

➔ Lancer ***pyzo***.
On installe les fichiers nécessaires et on prépare le dossier. Dans la console exécutez :

```
pip install brython             # On installe brython
pip install brython --upgrade   # Pour seulement mettre à jour brython
import subprocess               # module pour executer une commande externe
import os                       # module pour connaitre des informations de l'os
user=os.environ['USERNAME']     # C'est votre nom d'utilisateur
diruser=os.path.join('U:\\', user)  # Votre dossier utilisateur
os.chdir(diruser)               # On va dans votre dossier personnel
os.mkdir("test-brython")        # On créer le répertoire test-brython
os.chdir("test-brython")        # On se déplace dans le dossier.
subprocess.run([lienpy, "-m",  "brython","--install"]) # On prépare le dossier pour  brython
```

Vous disposez alors dans le dossier ***test-brython/*** des fichiers suivants :

- ***brython.js*** : le moteur Brython, à inclure dans la page HTML
- ***brython_stdlib.js*** : regroupe tous les fichiers de la distribution Python standard supportés par Brython

2 Création d'un exemple

2.1 Exemple 1

➔ Lancer un éditeur de texte (***notepad++***, pyzo, geany, ...)) et créez le fichier ***test-brython/exo1.html***.

```
<html>
<head>
  <meta charset="utf-8" />
  <script type="text/javascript" src="./brython.js"></script>
  <script type="text/javascript" src="./brython_stdlib.js"></script>
  <title>Clock</title>
</head>
<body onLoad="brython()">
<h1>Exemple 1:</h1>
<div>
  <button id="bouton" type="button">Cliquez!</button>
  <label id="hi">Un texte</label>
</div>
<script type="text/python">
from browser import document
def affiche(ev):
    document["hi"].text="Hello Word"
document["bouton"].bind("click", affiche)
</script>
</body>
</html>
```

Les fichiers HTML peuvent être ouverts directement dans le navigateur, mais il est préférable (surtout avec chrome) de lancer un serveur web dans le répertoire de l'application.

➔ Il faut donc lancer un serveur. Dans la console pyzo tapez :

- Chez vous il suffit de taper

```
!python -m http.server 8080
```

- Au lycée on utilisera

```
import subprocess                #module pour executer une commande externe
import os                        #pour connaitre des informations de l'os
user=os.environ['USERNAME']      #C'est votre nom d'utilisateur
diruser=os.path.join('U:\\', user)  # Votre dossier
os.chdir(diruser)                # On va dans votre dossier personnel
subprocess.run([lienpy, "-m",  "http.server","8080"])  # On lance le serveur
```

➔ Lancer un navigateur de votre choix à l'adresse http://localhost:8080 et lancer le fichier ***exo1.html***.

Exemple 1:

Cliquez! Un texte

Exemple 1:

Cliquez! Hello Word

document["bouton"] est l'élément qui a pour ***id*** "bouton", donc on définit l'évènement ***click*** sur le button d'id "bouton" qui déclenche la fonction ***affiche***.
Remarque : C'est la même syntaxe que pour tkinter.

2.2 Passer des arguments dans la fonction

Il est aussi possible d'envoyer des arguments à la fonction ***affiche*** en utilisant les fonctions ***lamda***.
Modifier le code comme ci-dessous.

```
<script>
from browser import document
def affiche(message,ev):
    document['hi'].text=message

message="Hello World"
document["bouton"].bind("click", lambda ev: affiche(message,ev))
</script>
```

La syntaxe avec la fonction lambda sera identique avec tkinter.

2.3 Interagir avec le clavier.

```
<div>
  <label id='hi'>Appyer sur la touche esc</label>
</div>
<script type="text/python">
from browser import document
def affiche(message,ev):
    if int(ev.which)==27 :
        document['hi'].text=message

document.bind('keydown', lambda ev :  affiche("Hello World", ev))
document.bind('keyup',   lambda ev :  affiche("Appyer sur la touche esc", ev))
</script>
```

Remarque : La touche ***Esc*** a pour ***keycode 27*** (***ev.Keycode*** est déprécié, on utilise à la place ***ev.which***).
On lie (bind) l'évènement ***'keydown'*** à la fonction affiche avec le paramètre ***ev*** (ev signifie ***event (évènement)***).
➔ Liste des évènements que l'on peut utiliser.

- Événements souris
  - ***mouseenter*** la souris entre dans la zone couverte par l'élément, ou un de ses descendants
  - ***mouseleave*** la souris sort de la zone couverte par l'élément et par ses descendants
  - ***mouseover*** la souris entre dans la zone couverte par l'élément
  - ***mouseout*** la souris quitte la zone couverte par l'élément
  - ***mousemove*** la souris se déplace sur l'élément
  - ***mousedown*** appui sur le bouton gauche de la souris
  - ***mouseup*** relâchement du bouton gauche de la souris
  - ***click*** clic : appui puis relâchement du bouton gauche de la souris
  - ***dblclick*** double clic
- Événements clavier
  - ***input*** déclenché quand la valeur d'un élément <input> ou <textarea> est modifié, ou quand le contenu d'un élément contenteditable est modifié
  - ***keydown*** appui sur une touche quelconque du clavier
  - ***keypress*** appui sur une touche du clavier qui produit un caractère. Par exemple, quand on entre *Ctrl+C* au clavier, l'événement keypress n'est déclenché qu'au moment où on appuie sur C, alors que keydown est déclenché dès l'appui sur Ctrl
  - ***keyup*** relâchement d'une touche enfoncée
- Focus events
  - ***blur*** un élément a perdu le focus
  - ***focus*** un élément a reçu le focus

➔ Liste des ***KeyCodes*** que l'on peut utiliser.

backspace : 8	insert : 45	g : 71	y : 89	decimal point : 110	comma : 188
tab : 9	delete : 46	h : 72	z : 90	divide : 111	dash : 189
enter : 13	0 : 48	i : 73	left window key : 91	f1 : 112	period : 190
shift : 16	1 : 49	j : 74	right window key : 92	f2 : 113	forward slash : 191
ctrl : 17	2 : 50	k : 75	select key : 93	f3 : 114	grave accent : 192
alt : 18	3 : 51	l : 76	numpad 0 : 96	f4 : 115	open bracket : 219
pause/break : 19	4 : 52	m : 77	numpad 1 : 97	f5 : 116	back slash : 220
caps lock : 20	5 : 53	n : 78	numpad 2 : 98	f6 : 117	close braket : 221
escape : 27	6 : 54	o : 79	numpad 3 : 99	f7 : 118	single quote : 222
(space) : 32	7 : 55	p : 80	numpad 4 : 100	f8 : 119	
page up : 33	8 : 56	q : 81	numpad 5 : 101	f9 : 120	
page down : 34	9 : 57	r : 82	numpad 6 : 102	f10 : 121	
end : 35	a : 65	s : 83	numpad 7 : 103	f11 : 122	
home : 36	b : 66	t : 84	numpad 8 : 104	f12 : 123	
left arrow : 37	c : 67	u : 85	numpad 9 : 105	num lock : 144	
up arrow : 38	d : 68	v : 86	multiply : 106	scroll lock : 145	
right arrow : 39	e : 69	w : 87	add : 107	semi-colon : 186	
down arrow : 40	f : 70	x : 88	subtract : 109	equal sign : 187	

2.4 Utilisation de la balise *<input>*.

La balise ***<input>*** est utilisée pour créer un contrôle interactif dans un formulaire web qui permet à l'utilisateur de saisir des données. Les saisies possibles et le comportement de l'élément ***<input>*** dépend fortement de la valeur indiquée dans son attribut type.
Regardez les différentes balises ***<input>*** http://www.startyourdev.com/html/tag-html-balise-input

Modifier le code comme ci-dessous.

➔ Pour la partie html :

```
<div>
<label for="nombre">Entrer un nombre</label>
<input type="number" min="0" max="100" id="nombre" /><br />
<button id="bouton" type="button" >Cliquez!</button>
<label id="reponse"></label>
</div>
```

➔ Pour la partie Brython :

```
<script type="text/python">
from browser import document
def affiche(ev):
    n=int(document["nombre"].value)
    document["reponse"].text = 'Le double de {0} est {1}'.format(n,2*n)

document["bouton"].bind("click", affiche)
</script>
```

3 Accéder aux éléments de la page

Pour accéder à un élément, on peut utiliser plusieurs méthodes.
La plus courante est de se servir de son identifiant, c'est-à-dire de son attribut ***id*** :

➔ si on a une zone de saisie définie par `<input id="data">` on peut obtenir une référence à ce champ par

```
    from browser import document
    data = document["data"]
```

➔ On peut aussi récupérer tous les éléments d'un certain type, par exemple tous les liens hypertexte (balise HTML A), en utilisant la syntaxe

```
from browser import html
links = document[html.A]
```

➔ Enfin, tous les éléments de la page possèdent une méthode ***get()*** qui permet de rechercher des éléments de plusieurs façons :

- `elt.get(name=N)` retourne une liste avec tous les éléments descendant de *elt* dont l'attribut name est N.
- `elt.get(selector=S)` retourne une liste avec tous les éléments descendant de *elt* dont le sélecteur CSS correspond à S.

Quelques exemples :

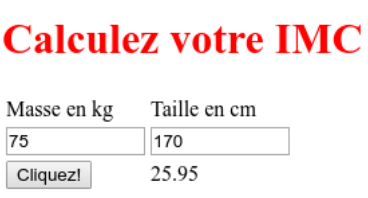
```
document.get(selector='.foo')       # éléments avec la classe "foo"
document.get(selector='form')       # liste des balises "<form>"
document.get(selector='H1.bar')     # balises H1 avec la classe "bar"
document.get(selector='#container') # liste avec l'élément dont l'id vaut "container",
    similaire à [document["container"]]
document.get(selector='a[title]')   # balises A avec un attribut "title"
```

4 TP

4.1 IMC

Nous allons créer une page html, un calculateur d'indice de masse corporelle (IMC).
On utilisera

- deux labels (masse et taille),
- deux balises ***<input>*** de type number,
- un bouton et un label pour afficher l'imc.
- On placera le tout avec un tableau pour faire simple.



$IMC = \frac{masse}{taille^2}$ où la masse est en kg et la taille en mètre.
Solution sur mon site

4.2 Dimension de l'écran

- Faire une recherche sur internet avec les mots clés : *dimension ecran javascript*
- Comment connait on en javascript les dimensions de l'ecran ? ***window.*_____** et ***window.*_____**
- Donc avec ***Brython*** on utilisera ***window.*_____**
- Créer une page html qui affiche les dimensions de *l'écran.*

4.3 Savoir si on utilise un mobile ou un pc

- Faire une recherche sur internet avec les mots clés : *mobile ou pc en javascript*
- Comment connait on en javascript les informations de son navigateur ? ***navigator.*_____**
- Donc avec ***Brython*** on utilisera ***window.navigator.*_____**
- Créer une page html qui affiche le nom du navigateur.
- Créer une fonction python retourne ***True*** si le navigateur est su pc ou ***False*** si on est sur un mobile.

```
from browser  import window
def ismobile():
    Agent=window_navigator_________
    Liste_mobile=['Phone','iPod','Android','opera mini','blackberry','palm os','palm','
    hiptop','avantgo','plucker','xiino','blazer','elaine','iris','3g_t','windows ce','opera
    mobi','windows ce; smartphone;','windows ce']
    if ..................... :
        return True
    else:
        return False
```

- Compléter le fichier html pour faire apparaitre une image seulement si le navigateur n'est pas sur un mobile.