UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA TRIENNALE
IN INGEGNERIA INFORMATICA

"Data Scraping ed Estrazione Dati Automatizzata dal Web"

Relatore: Prof. Di Nunzio Giorgio Maria

Laureando: Agostini Massimiliano

ANNO ACCADEMICO 2021 – 2022

Data di laurea 19/09/2022

INDICE

Introduzione…… 3

1. IL WEB SCRAPING…… 4
2. PYTHON, SELENIUM E PANDAS…… 5
   2.1. Il Setup dell'Ambiente sul mio computer 5
3. USO DI SELENIUM NELLA NAVIGAZIONE…… 6
   3.1 I moduli importati 7
4. STRUTTURA DEL PROGRAMMA…… 8
5. I SITI ANALIZZATI…… 9
6. CALABRIA…… 10
   6.1 Bandi Europei 10
      6.1.1 Analisi 10
      6.1.2 Script 13
   6.2 Bandi Regionali 17
      6.2.1 Analisi 17
      6.2.2 Script 18
   6.3 Siti esclusi 19
7. LAZIO…… 20
   7.1 Bandi Europei 10
      7.1.1 Analisi 20
      7.1.2 Script 21
   7.2 BUR 24
      7.2.1 Analisi 24
   7.3 Lazio Innova 25
      7.3.1 Analisi 25
      7.3.2 Script 28
8. MOLISE…… 29
9. REVISIONE TESI PRECEDENTI DEI COLLEGHI…… 32
   9.1 Miola Giacomo – Veneto, Friuli Venezia Giulia e Trentino Alto Adige 32
   9.2 Walkuski Pavel – Lombardia, Piemonte e Valle D'Aosta 33
   9.3 Crociani Riccardo – Marche, Toscana e Umbria 33
   9.4 Forzan Riccardo – Basilicata, Puglia, Sardegna e Sicilia
      Spina Lorenzo – Abruzzo Emilia Romagna, Campania e Liguria 33
10. CONCLUSIONI…… 34
11. CODICE…… 35
   11.1 ws_calabria.py – Regione Calabria 35
   11.2 ws_lazio.py – Regione Lazio 38
   11.3 ws_molise.py – Regione Molise 42

INTRODUZIONE

"Per scraping dei dati (più brevemente scraping*), nella sua forma più generale, si intende una tecnica in cui un programma informatico estrae dei dati dall'output generato da un altro programma. Lo scraping si esplicita comunemente nel web scraping, che è il processo nel quale una applicazione estrae informazioni di valore da un sito web."* [1]

La tesi consiste nell'illustrare il processo di creazione di script atti al caricamento di alcune pagine web ed alla successiva estrapolazione di dati d'interesse.

Questo processo, definito per l'appunto Web Scraping, automatizza quello che sarebbe il medesimo processo ma effettuato da un operatore, abbreviandone sensibilmente i tempi di esecuzione ed ovviando a quello che può essere un qualsiasi errore umano.

In questa tesi le pagine prese in considerazione sono quelle che riportano e riassumono i bandi di concorso aperti in tre regioni Italiane, Calabria, Lazio e Molise.
L'insieme dei siti studiati è disomogeneo, in quanto essi appartengono a regioni ed enti differenti e sono nati per mano di differenti programmatori e web designer.
Ogni sito richiede quindi uno sviluppo a se stante e non vi è certezza di essere in grado di estrapolare le medesime informazioni da tutti e nemmeno la sicurezza di portare a termine un'efficace estrazione dei dati per i più disparati motivi che successivamente analizzeremo.

La scelta del linguaggio utilizzato, Python, nasce dalla ricerca di un linguaggio di alto livello, snello e veloce da leggere e scrivere, capace di essere eseguito in più dispositivi
nelle medesime modalità e senza modifiche al codice,
grazie alla semplicità di configurazione dell'ambiente.

Una particolarità che caratterizza il web scraping il suo essere effimero.
È infatti sufficiente una piccola modifica alla struttura del sito, nata da necessità di rinnovamento dell'interfaccia, dall'inserimento di nuovi oggetti e banner,
da alcune modifiche nella legislazione eccetera
per rendere il codice inutilizzabile o deficitario.

Questo aspetto si è evidenziato nel momento di revisione del codice di alcuni colleghi che hanno affrontato precedentemente questa tesi, concentrandosi su altre regioni rispetto alle mie. In qualche caso mi sono trovato semplicemente a sistemare qualche punto dopo un importante aggiornamento della libreria utilizzata, di cui parleremo successivamente, in altri addirittura ad aggiungere o modificare codice per rendere il progetto funzionante.

Nei capitoli a venire parlerò delle librerie utilizzate, dei problemi incontrati e delle possibili soluzioni individuate, quando possibile. Infine un piccolo excursus sulle revisioni delle vecchie tesi.

1. IL WEB SCRAPING

Per web scraping si intende la pratica di creare applicativi (o script) atti ad estrarre dati interessanti da una o più pagine web.
Queste estrapolazioni avvengono per lo più attraverso la simulazione, da parte dei programmi sviluppati, dell'interazione umana sulla pagina. Il web scraping, infatti, è una pratica diffusa ma al contempo molto controversa e, nei confronti della quale, molti siti stanno attrezzando.
Solitamente, infatti, i proprietari dei siti web non gradiscono l'uso dei dati pubblicati sul sito da parte di soggetti terzi, che possono anche scaricarli, rielaborarli e diffonderli.

Si veda per esempio la possibilità di mettere a confronto i prezzi di uno stesso prodotto venduto in più e-commerce. Molte volte la difficoltà di non avere i prezzi a confronto diretto l'uno con l'altro di portano a non acquistare il prodotto desiderato al prezzo migliore. Con il web scraping si potrebbero ricavare, prodotto per prodotto e portale per portale, tutti i prezzi disponibili, oppure si potrebbe riuscire ad individuare automaticamente e senza possibilità d'errore il prezzo migliore.

La presenza di un bot che ricarica le pagine del sito, magari più volte al minuto, o al secondo, potrebbe inoltre inficiarne le prestazioni, anche nei confronti di altri utenti. Un effetto simile a quello di un attacco DOS (Denial Of Service) dove appunto un sito o un servizio vengono interrogati ripetutamente fino al collasso del server.

Solitamente i web server su cui è pubblicata la pagina web tendono a limitare gli accessi al sito da quell'IP pubblico da cui suppongono arrivare troppe richieste.
É successo anche a me sviluppando questa tesi e sono stato costretto a disconnettere e riconnettere il modem per far sì che il mio ISP mi assegnasse un nuovo indirizzo IP pubblico.

Famosa è anche la finestrella di verifica di Google che ci invita a mettere la spunta sull'opzione "Non sono un robot" e a provarlo successivamente con piccoli giochini difficilmente risolvibili da un computer.

Diffusa è anche la tecnica di utilizzare script lato client organizzati in un DOM tree, più difficilmente navigabile da programmi che riconoscono in automatico le caratteristiche di una pagina web, necessitando in questo caso di uno sviluppo di codice ad hoc.

Il web scraping si pone quindi l'obiettivo di aggirare queste limitazioni imitando il più possibile l'attività umana per rendersi trasparente e difficilmente distinguibile dall'utente reale.

2. PYTHON, SELENIUM E PANDAS

Come anticipato nell'introduzione, il linguaggio che ho utilizzato è Python.

Molteplici sono le librerie per svariati linguaggi, ma il suo essere ad alto livello, veloce ed immediato nella lettura e nella scrittura lo ha reso la scelta ideale.

Molteplici prodotti sono disponibili e atti allo scopo; i più famosi sono Beautiful Soup e Selenium. Il primo si presenta con metodi molto intuitivi e semplici da utilizzare ed implementare mentre la seconda consente una lavorazione più precisa e puntuale, a scapito della semplicità di utilizzo. Il componente che utilizzerò prende il nome di WebDriver. Nasce principalmente per eseguire dei test sui siti web emulando l'interazione umana, il che lo rende lo strumento ideale al nostro scopo.

Ho deciso di utilizzare il WebDriver di Chrome, che utilizza le API fornite direttamente da Google per il suo browser, essendo il più utilizzato e, di conseguenza, quello compatibile con la maggior parte di siti e applicativi.

Per la rielaborazione dei dati estratti mi sono affidato a Pandas, libreria per Python specifica per l'analisi e la manipolazione dei dati, che consente di esportare i dati in un foglio di calcolo (es .xlsx, il formato di Microsoft Excel) rendendoli facili da leggere ed elaborare anche per l'utente medio.

2.1. Il Setup dell'Ambiente sul mio computer.

La configurazione di Python, Selenium e chromewebdriver sono estremamente semplici ed alla portata.

Ho sviluppato la tesi su un PC con Ubuntu 22.04, che presenta già installati Python e Pip, il componente che consente di installare moduli e librerie.

Con Pip procedo all'installazione di Selenium e Pandas, aprendo il terminale Linux e digitando "pip install selenium" e "pip install pandas" seguiti da un invio.
Diversa è la questione del webdriver, scaricabile da internet.
Per utilizzarlo vi sono varie opzioni:
- inserire il file nella cartella stessa contenente il programma
- dichiarare una variabile PATH nel programma contenente l'indirizzo del file ed usarla per dichiarare il webdriver
- Dichiarare la cartella tra i percorsi delle variabili d'ambiente
- Inserire il driver in un percorso già presente tra le variabili d'ambiente.

Personalmente ho scelto l'ultima opzione, ritenendola la strada più facile e veloce.

3. USO DI SELENIUM NELLA NAVIGAZIONE

La prima cosa da fare è importare il pacchetto contenente il codice inerente al WebDriver inserendo, in testa allo script la dicitura

```
from selenium import webdriver
```

Una volta fatto, occorre creare un oggetto di tipo WebDriver che in tutto il resto del mio elaborato chiamerò, per comodità "browser"

```
browser = webdriver.Chrome()
```

Come specificato in precedenza non vado ad indicare in alcun modo il precorso contenente il chromewebdriver essendo inserito in una cartella già dichiarata tra le variabili d'ambiente.

Il caricamento di una pagina web avviene attraverso il comando "get" inserendo il link desiderato

```
browser.get(https://www.google.com)
```

Questo aprirà una finestra di browser controllata dal driver che caricherà la pagina all'url indicato. Una volta completato il caricamento della stessa, il driver ridarà il controllo allo script.

Paragrafo 3.1: Individuazione degli elementi web

Nel browser verrà caricata la pagina richiesta in linguaggio HTML. Esso può essere esplorato aprendo la stessa pagina in una normale finestra di browser; una volta premuto il tasto F12 verrà mostrato tutto il codice che va a comporre la pagina.

Ci si potrà riferire ai vari elementi nella pagina con diversi identificatori, attraverso i metodi messi a disposizione. Essi sono:

- ID
- TAG NAME
- CLASS NAME
- CSS SELECTOR
- XPATH
- LINK TEXT e PARTIAL LINK TEXT

Questi identificatori potranno essere usati per individuare singoli elementi:

```
browser.find_element(By.CLASS_NAME, 'nome')
```

Oppure per identificare una molteplicità di elementi, restituiti sottoforma di lista

```
browser.find_elements(By.CSS_SELECTOR, 'css')
```

Una volta identificato l'elemento vi si potranno compiere svariate azioni, click, compilazione del campo di testo, apertura in una nuova scheda e così via.

Una nota importante da sottolineare è che la chiamata ai metodi che ho presentato sopra è figlia di un aggiornamento della libreria Selenium. Tempo addietro la sintassi era differente. Fino alla versione 4.2 della libreria, era tollerata la vecchia sintassi e suggerita la nuova in fase di esecuzione dello script. Dalla versione 4.3 in poi la vecchia sintassi risulta non essere più funzionante.

3.1: Moduli importati

A supporto della programmazione ho importato i seguenti moduli che vado a presentare.

Nel caso, ne prosieguo dell'elaborato, si necessiti di altri moduli, ve li andrò a presentare singolarmente.

```
import pandas as pd
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from datetime import date, datetime
from selenium.webdriver.common.keys import Keys
```

Pandas per l'esportazione dei dati elaborati in un foglio di calcolo attraverso la creazione di una tabella a partire, nel mio caso, da una lista di dizionari che presentano, come chiave, gli attributi che sono andato a ricercare.

Webdriver, di cui abbiamo ampiamente discusso in precedenza

ActionChains che ho utilizzato per operazioni più complesse del semplice click, solitamente per spostare a pagina in modo da rendere visibile un elemento per poi cliccarvi sopra. Ricordo che WebDriver agisce come un utente umano e può quindi cliccare solo ciò che vede.

By che, come ho illustrato precedentemente, si utilizza nel cercare elementi con i metodi find_element e find_elements al momento di definire l'identificatore desiderato

Options consente di definire alcune opzioni del webdriver per migliorare l'esperienza di rielaborazione

Date e datetime si sono resi necessari per completare alcuni campi di testo con la data attuale, per ricercare i bandi ancora attivi.

Infine, Keys consente di emulare la pressione dei tasti della tastiera all'interno del browser.

4. STRUTTURA DEL PROGRAMMA

Nel mio modesto trascorso da programmatore ho appreso che, per la comprensibilità del codice da parte mia e di terzi, è essenziale standardizzare e semplificare il più possibile la scrittura.

Questo avviene attraverso alcune accortezze:

- Commentare i passaggi meno chiari ed immediati del codice
- Creare codice snello, scrivendo metodi dal nome chiaro e con poche operazioni al loro interno
- Creando delle classi standard in tutti gli script, che eseguano la medesima procedura seppur adattata alle esigenze del momento.

Analizzando il mio codice incontreremo questi quattro metodi principali:

- Il metodo che inizializza l'esplorazione della pagina e ne raccoglie i frutti
- "crawler", ovvero il metodo che si occupa di esplorare le sezioni per noi interessanti della pagina, estrarre eventuali collegamenti ed aprire i collegamenti individuati
- "scraper" ovvero il metodo che effettivamente estrae i dati, in questo caso dei bandi per restituire i dati
- Il "main", inizializzato con `if __name__=='__main__':` mi consente di lanciare il codice contenuto nello script ivi dichiarandovi i metodi necessari solo se lo stesso è eseguito singolarmente e restando inattivo nel caso in cui lo script o qualche metodo vengano richiamati da altri script.

5. I SITI ANALIZZATI

Insieme a due colleghi, coinvolti anche loro nel progetto di tesi, abbiamo provveduto a reperire alcuni collegamenti ai principali portali delle regioni interessate per la pubblicazione dei bandi pubblici di concorso.

Ho scelto di concentrare i miei sforzi sui bandi di concorso aperti o in prossima apertura poiché un domani, se si dovesse applicare questa procedura, immagino che l'interesse sia rivolto a quei bandi.

I portali recuperati per le regioni a me assegnate sono i seguenti:

Calabria

- http://calabriaeuropa.regione.calabria.it/website//bandi/
- http://www.consiglioregionale.calabria.it/portale/BancheDati/BURC/BURC_I_II_Form (Fino 2019)
- https://burc.regione.calabria.it/ricerca.jsp
- https://www.regione.calabria.it/website/bandiregione/

Lazio

- http://www.lazioeuropa.it/bandi/
- http://pubbur.ised.it/PublicBur/burlazio/FrontEnd/RicercaEdiz

Molise

- http://www3.regione.molise.it/flex/cm/pages/ServeBLOB.php/L/IT/IDPagina/3 (aree tematiche)
- http://www3.regione.molise.it/flex/cm/pages/ServeBLOB.php/L/IT/IDPagina/14700 (POR e altri, manca PSR)
- http://psr.regione.molise.it/aperti (solo PSR)
- http://www3.regione.molise.it/flex/cm/pages/ServeBLOB.php/L/IT/IDPagina/18

Prima di procedere con la parte di progettazione e realizzazione dello script ho effettuato l'analisi del sito per capire quali dati fosse possibile estrarre e in che modo farlo. Questo mi ha portato ad escludere alcuni siti dal progetto perché l'estrazione sarebbe risultata fallimentare o estremamente controproducente.
In alcuni casi invece, nonostante il quantitativo irrisorio di dati che si sarebbero potuti raccogliere, ho proceduto egualmente all'estrazione.

Puntualizzo che il codice che inserirò in fotografia non è la sua totalità, che inserirò in fondo a questo elaborato e renderò disponibile alla consultazione su GitHub, bensì un estratto delle parti salienti che identificheranno le peculiarità implementate per i singoli siti.

Mi troverò spesso ad incollare schermate del mio computer per illustrare passaggi e, soprattutto, risultati. Questi ultimi, per una migliore lettura saranno pubblicati su GitHub insieme al codice.

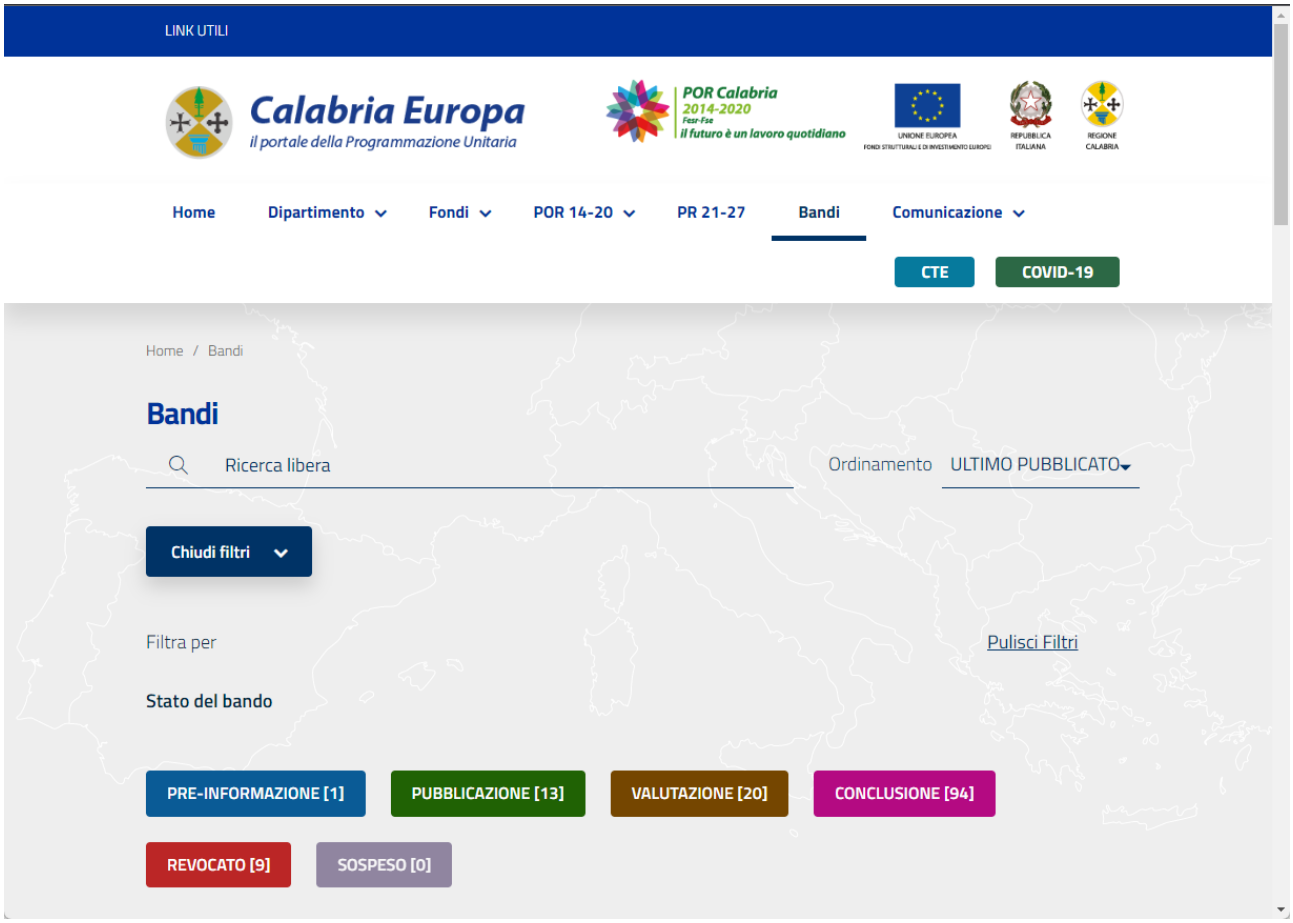
Nella prima sezione sarò più specifico per evidenziare i passaggi fondamentali che poi darò come scontati.

6. CALABRIA

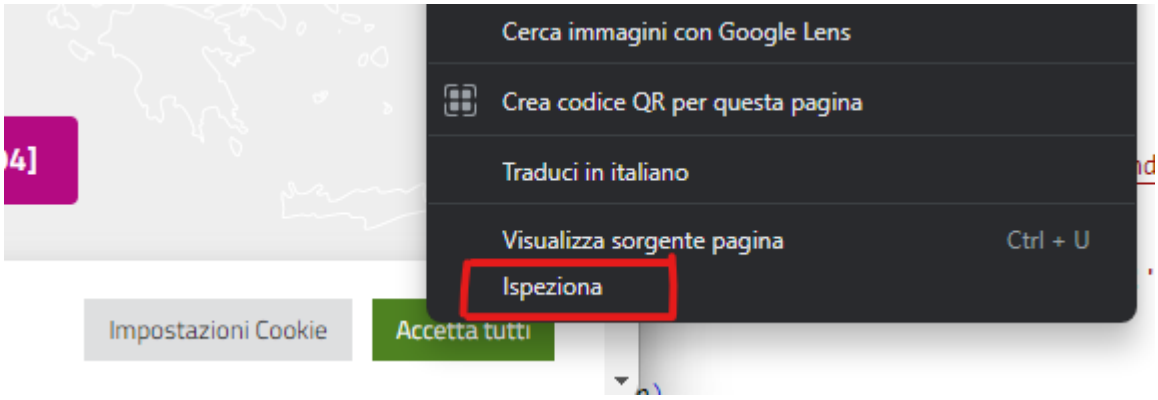
6.1. Bandi Europei

6.1.1. Analisi

La pagina si apre subito evidenziando uno spazio dedicato alla ricerca selettiva dei bandi dando la possibilità di effettuare subito una grande scrematura selezionando quelli la cui apertura è prossima e quelli già aperti, escludendo già in fase di ricerca quelli chiusi o in altro stato.



Essendo questo il primo sito analizzato, per familiarizzare con i moduli e le possibilità di Selenium, ho utilizzato qualche piccolo script per le mie prove. Mi sono subito accorto dell'importanza di togliere dalla visuale il banner nella zona inferiore riguardo il consenso all'uso dei Cookies, che nascondeva alcuni oggetti, impedendomi di interagirvi.

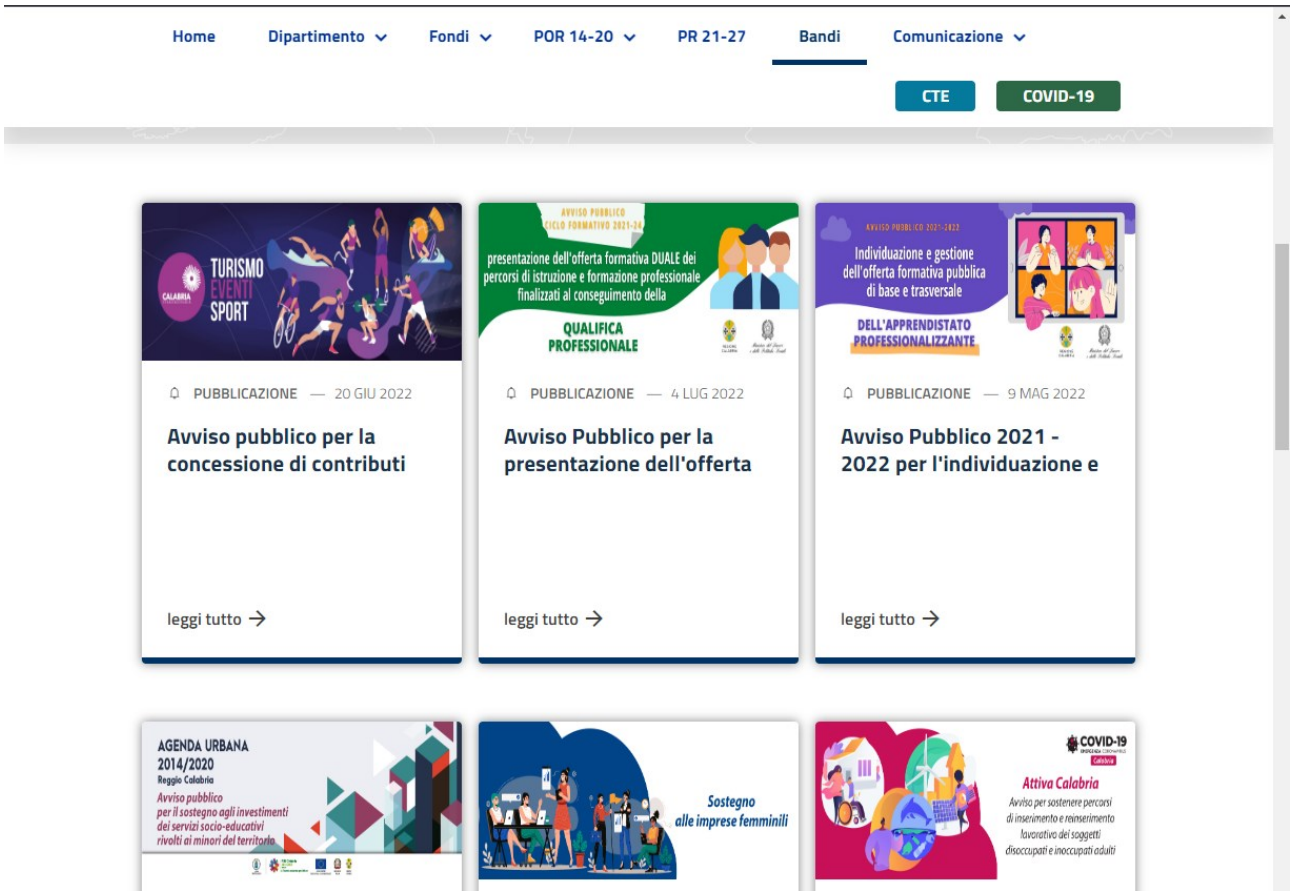


Un metodo rapido per individuare gli identificatori di un dato elemento è cliccarvi sopra col tasto destro del mouse e selezionare la voce "Ispeziona"

Ecco cosa viene visualizzato in Chrome:

```
<a id="wt-cli-accept-all-btn" tabindex="0" role="button" data-cli_action="accept_all" class="wt-cli-element m
edium cli-plugin-button wt-cli-accept-all-btn cookie_action_close_header cli_action_button" style="color: rgb
(255, 255, 255); background-color: rgb(97, 162, 41);">Accetta tutti</a> == $0
```

Una volta effettuata la ricerca la pagina presenta un'anteprima di ogni bando ma è solo cliccandovi sopra che possiamo accedere alla pagina con tutte le informazioni di cui necessitiamo.



All'interno del codice della pagina troviamo una serie di DIV annidati. I DIV di nostro interesse, quelli che identificano le nostre anteprime cliccabili, sono quelli che presentano uguale "class".

```
▼<div class="container">
  ▼<div class="widget_category_mansory">
    ▼<div class="_card-columns row"> flex
      ▶<div class="col-lg-4 col-lg-4 col-12 p-0 p-md-2 mb-4">…</div>
      ▶<div class="col-lg-4 col-lg-4 col-12 p-0 p-md-2 mb-4">…</div>
      ▶<div class="col-lg-4 col-lg-4 col-12 p-0 p-md-2 mb-4">…</div>
      ▶<div class="col-lg-4 col-lg-4 col-12 p-0 p-md-2 mb-4">…</div>
      ▶<div class="col-lg-4 col-lg-4 col-12 p-0 p-md-2 mb-4">…</div>
      ▶<div class="col-lg-4 col-lg-4 col-12 p-0 p-md-2 mb-4">…</div>
      ▶<div class="col-lg-4 col-lg-4 col-12 p-0 p-md-2 mb-4">…</div>
      ▶<div class="col-lg-4 col-lg-4 col-12 p-0 p-md-2 mb-4">…</div>
      ▶<div class="col-lg-4 col-lg-4 col-12 p-0 p-md-2 mb-4">…</div>
      ▶<div class="col-lg-4 col-lg-4 col-12 p-0 p-md-2 mb-4">…</div>
      ▶<div class="col-lg-4 col-lg-4 col-12 p-0 p-md-2 mb-4">…</div>
      ▶<div class="col-lg-4 col-lg-4 col-12 p-0 p-md-2 mb-4">…</div>
      ▶<div class="col-lg-4 col-lg-4 col-12 p-0 p-md-2 mb-4">…</div>
    </div>
  </div>
</div>
```

Esplorando le pagine dettagliate dei bandi noto con piacere come esse siano standardizzate e sintetiche consentendomi quindi una proficua estrazione.

Ogni sezione della pagina è contenuta in un DIV (foto sotto) al cui interno il titolo è segnalato dal tag "h3" mentre il resto del testo è contenuto in un DIV annidato.

```
<br>
▶ <div class="pl-md-3 pl-0">…</div> == $0
▶ <div class="pl-md-3 pl-0">…</div>
▶ <div class="pl-md-3 pl-0">…</div>
▶ <div class="pl-md-3 pl-0">…</div>
▶ <div class="pl-md-3 pl-0">…</div>
▶ <div class="pl-md-3 pl-0">…</div>
▶ <div class="pl-md-3 pl-0">…</div>
▶ <div class="pl-md-3 pl-0">…</div>
```

6.1.2. Script

In questa prima parte dichiaro la variabile browser di tipo webdriver e apro il sito con il comando "get"

Passo poi alla fase di accettazione dei cookies. Utilizzo il costrutto try-except per gestire l'eccezione ElementNotFoundException nel caso in cui il banner dei cookies non fosse presente e il driver non trovasse il pulsante da cliccare.

```
browser = webdriver.Chrome()
browser.get('http://calabriaeuropa.regione.calabria.it/website//bandi/')
try:
    browser.find_element(By.ID, 'wt-cli-accept-all-btn').click() #cookies
except:
    pass
```

Esaurita questa parte passo alla successiva, chiamando il mio metodo "crawler1" chiedendogli di trovare i bandi pre-pubblicati in una prima iterazione e quelli pubblicati nella seconda.
Uso il numero uno e seguenti se, all'interno dello stesso script ho inserito più metodi per pagine diverse.

```
def crowler1(browser, type):
    actions = ActionChains(browser)
    bandi=[]
    catselector = browser.find_element(By.ID, type)
    actions.move_to_element(catselector).click(catselector).perform()
    try:
        lista = browser.find_elements(By.XPATH, "//div[@class='col-lg-4 col-lg-4 col-12
    except:
        print('bandi non trovati/pagina irraggiungibile')
    for ii in lista:
        actions.move_to_element(ii).click(ii).perform()
        tmp_bando = scraper1(browser)
        bandi.append(tmp_bando)
        browser.back()
    return(bandi)
```

Nell'ordine inizializzo actions per potermi muovere nella pagina e cliccare, inizializzo bandi, la mia lista che conterrà un dizionario per ogni bando analizzato.

Il metodo move_to_element mi consente di spostare la pagina in modo da rendere visibile e cliccabile l'elemento desiderato.

Chiamo catselector l'oggetto restituitomi dal metodo, cioè il tasto che mi permette di selezionare il tipo di bando. All'interno della variabile "type" viene passato il valore string dell'id del tasto, scoperto consultando l'HTML della pagina.

Successivamente vado a ricavarmi la lista di anteprime cliccabili visualizzate cercando l'oggetto con il suo tag "div" e la sua classe. Molte volte mi troverò ad usare la ricerca per xpath, che trovo molto flessibile e che consente di adattarsi in molte occasioni.

Dopodiché procedo ad aprire pagina per pagina tutte le anteprime e a far elaborare la pagina aperta allo scraper.

Una volta finita l'elaborazione con browser.back() torno alla pagina precedente.

```
def scraper1(browser):
    bando = {}
    bando['title'] = browser.find_element(By.XPATH, "//h1[@class='entry-title ']").text
    bando['url'] = browser.current_url
    specs_h = browser.find_elements(By.XPATH, "//h3[@class='cem-description-title']")
    specs_b = browser.find_elements(By.XPATH, "//div[@class='cem-text-description cem-ove

    for ii in range(len(specs_h)):
        bando[specs_h[ii].text] = specs_b[ii].text

    return bando
```

Nello scraper ricevo il browser che ha caricato la pagina da cui estrarre i dati.
Riesco ad estrarre il titolo e tutti i sottotitoli ed il loro testo. Compongo il tutto all'interno di un dizionario che ritornerò alla fine del metodo.

```
def excelOut(bandi, nomeFile):
    excel = pd.DataFrame(bandi) #generate dataframe from list
    excel.to_excel(nomeFile+".xlsx", index = False) #export dataframe in excel
```

Con questo metodo, che resterà uguale per tutti i siti, creo il mio file di output, un foglio di calcolo grazie a Pandas.

Creo il dataframe inserendo direttamente la mia lista di dizionari che ho assemblato nei vari metodi, dopodiché creo l'output con il metodo .toExcel()
In questo modo otterrò un foglio di calcolo che presenta, come nomi colonna, i nomi delle

caratteristiche del bando elencate nel sito. Essendo standardizzate presentano anche lo stesso nome.

Al momento del test, il file di output presenta un'enormità di informazioni tanto da essere di difficile comprensione se inserito all'interno di questo elaborato.
Si può ovviamente procedere ad una selezione delle categorie da inserire per una maggiore immediatezza.

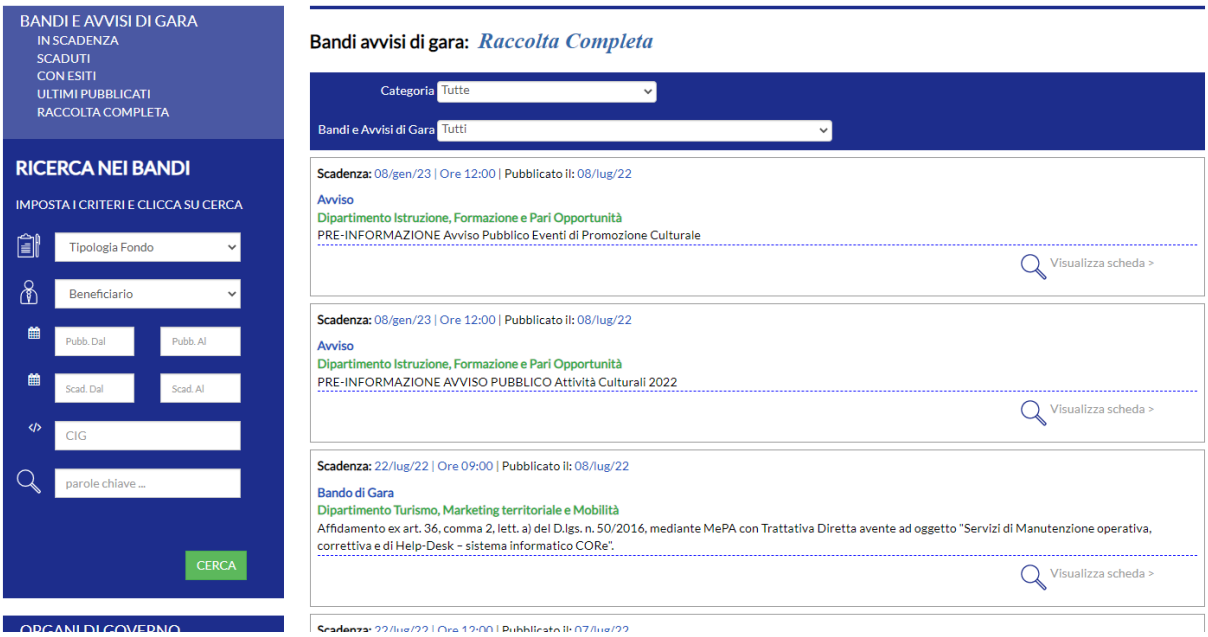
A riprova de risultato inserirò alcune delle colonne estratte dal sito.

Lo script viene correttamente eseguito senza errori, naviga in tutte le pagine del sito ed estrae tutti i valori di che desideriamo, esportandoli correttamente.

TITOLO	STANZIAMENTO	DOMANDA
Avviso Pubblico per la presentazione dell'offerta formativa duale dei percorsi di istruzione e formazione professionale finalizzati al conseguimento della qualifica professionale - Ciclo Formativo 2021/2024	Le risorse finanziarie disponibili ammontano complessivamente ad euro 4.108.851,00 (quattro milioni cento ottomila ottocento cinquantuno/00) a valere sulle risorse statali assegnate alla Regione Calabria dal Ministero del Lavoro e delle Politiche Sociali, previsti dall'art. 68, co. 4, della Legge n. 144/1999, come integrate dall'art. 32, co. 3, del D. Lgs. 14 settembre 2015, n. 150	L'Avviso, con la relativa modulistica per la presentazione della Domanda di finanziamento del progetto formativo, è visionabile sul sito della Regione Calabria
Avviso Pubblico 2021 - 2022 per l'individuazione e la gestione dell'offerta formativa pubblica di base e trasversale dell'apprendistato professionalizzante	Le risorse disponibili per il finanziamento della misura, oggetto del presente provvedimento, ammontano complessivamente ad € 248.589,00 a valere sulle risorse assegnate – in quota parte – alla Regione Calabria, dal Ministero del Lavoro e delle Politiche sociali.	L'Avviso, con la relativa modulistica per la presentazione della Domanda di finanziamento del progetto formativo, è visionabile sul sito della Regione Calabria
Fondo Artigianato	Lo stanziamento complessivo è di € 5.900.410,26 di cui: € 1.509.508,68 a valere su fondi del Bilancio Regionale; € 4.390.901,58 a valere sull'Azione 3.3.1 – Obiettivo specifico 3.3 – Asse 3 del Piano di Azione e Coesione (PAC) 2014/2020.	La domanda di ammissione alle agevolazioni, deve essere compilata dal Soggetto presentatore (Istituti di Credito/Associazioni di Categoria), utilizzando la modulistica messa a disposizione Soggetto gestore sul proprio sito internet e su quello della Regione Calabria, e dallo stesso caricata e inviata telematicamente. Lo sportello per la presentazione delle domande è aperto, per i soggetti abilitati all'inserimento ed in possesso delle relative credenziali, a decorrere dal 20 gennaio 2022 ore 12.00.

6.2 Bandi Regionali

6.2.1 Analisi



Il sito si presenta con tutti i bandi visibili già inizialmente, con la possibilità di esplorarli per sezioni con i menu cosiddetti “a tendina” in alto e con la possibilità di una ricerca più approfondita sulla sinistra.

I bandi si presentano nel seguente modo nel codice della pagina, una serie di div con lo stesso attributo “class style”

```
▸<div style="background-color:#1D2D8C; color:#fff; padding:10px" class>…</div>
▸<div class style="  border:1px solid #999;padding:8px; margin-top:8px; ">…</div>
▸<div class style="  border:1px solid #999;padding:8px; margin-top:8px; ">…</div>
▸<div class style="  border:1px solid #999;padding:8px; margin-top:8px; ">…</div>
▸<div class style="  border:1px solid #999;padding:8px; margin-top:8px; ">…</div>
▸<div class style="  border:1px solid #999;padding:8px; margin-top:8px; ">…</div>
▸<div class style="  border:1px solid #999;padding:8px; margin-top:8px; ">…</div>
▸<div class style="  border:1px solid #999;padding:8px; margin-top:8px; ">…</div>
▸<div class style="  border:1px solid #999;padding:8px; margin-top:8px; ">…</div>
▸<div class style="  border:1px solid #999;padding:8px; margin-top:8px; ">…</div>
▸<div class style="  border:1px solid #999;padding:8px; margin-top:8px; ">…</div>
```

Scelgo di usare la ricerca sulla sinistra ed inserire la data odierna in “Scad Dal” così da visualizzare unicamente i bandi aperti.

```
<input class="form-control hasDatepicker" placeholder="Scad. Dal" name="data_scad_dal" style="font
-size:11px; border-radius: 0px;" value id="dp1657336382801"> == $0
```

Una volta aperto il dettaglio del bando scopro che le informazioni comuni sono molto ridotte, mentre le altre sono varie, non uniformi e impossibili quindi da estrarre in maniera omogenea.

Si presentano come una serie di paragrafi con all'interno degli "span"

```
<p>
  <span class="txt_testo_bold">Scadenza</span>
  " | "
  <span class="txt_red"> 22/lug/22 | Ore 09:00 </span>
  <span class="txt_testo"> | Pubblicato il</span>
  <span class="txt_red"> 08/lug/22 </span>
</p>
<p>
  <span class="txt_red_bold">Bando di Gara</span>
  <br>
  <span class="regcolorbold">Dipartimento Turismo, Marketing territoriale e Mobilità</span>
  <br>
  " Affidamento ex art. 36, comma 2, lett. a) del D.lgs. n. 50/2016, mediante MePA con Trattativa Diretta
  avente ad oggetto "Servizi di Manutenzione operativa, correttiva e di Help-Desk - sistema informatico
  CORe"."
  <br>
</p>
```

6.2.2. Script

Il tratto saliente di questa parte di codice è la sperimentazione di un metodo diverso. Invece di cliccare volta per volta sui collegamenti, aprire la pagina, elaborarla e tornare indietro, ho deciso di raccogliere tutti i link dei pulsanti "Visualizza Dettagli", raccoglierli in un vettore ed aprirli uno per uno con "get".
Per tenere pulito il codice mi sono aiutato con la funzione linkCollector per recuperare i link e restituirli in un vettore

Trovo questo metodo ancora più rapido del precedente, tuttavia va fatta attenzione se si usa il browser per qualcos'altro, perché non tornerà alla pagina iniziale.

```
for ii in links:
    browser.get(ii)
    bando_tmp = scraper2(browser)
    if bando_tmp != None:
     bandi.append(bando_tmp)
return bandi
```

Un altro punto saliente è l'uso, per la prima volta, della compilazione di dei campi di testo nella ricerca.

```
links = linkCollector2(browser, 'Visualizza ')
```

Una volta individuato l'identificatore per individuare il campo dove inserire la data, ho inserito la data odierna ricavata da date e avviato il processo di crowling e scraping dei risultati.

Come prima il tutto è caricato in una lista di dizionari e elaborata da Pandas per la creazione di un Excel.

Riporto il risultato ottenuto in fase di test quando l'unico bando aperto è quello esplorato.

Scadenza	Pubblicazione	Descrizione	Ente appaltante	Nome Procedura	Tipo Procedura	Stato	RUP	N. Identificativo	Base Asta	E-Mail
12/lug/22 | Ore 12:00	22/giu/22	Bando di GaraSTAZIONE UNICA APPALTANTEProcedura aperta, con r	STAZIONE UNICA APPALTANTE	Gara_vaccini_2022_2023	Multilotto	In Corso	Vincenzo	8597420	€ 6.862.500,00	v.ragusa@regione.calabria.it

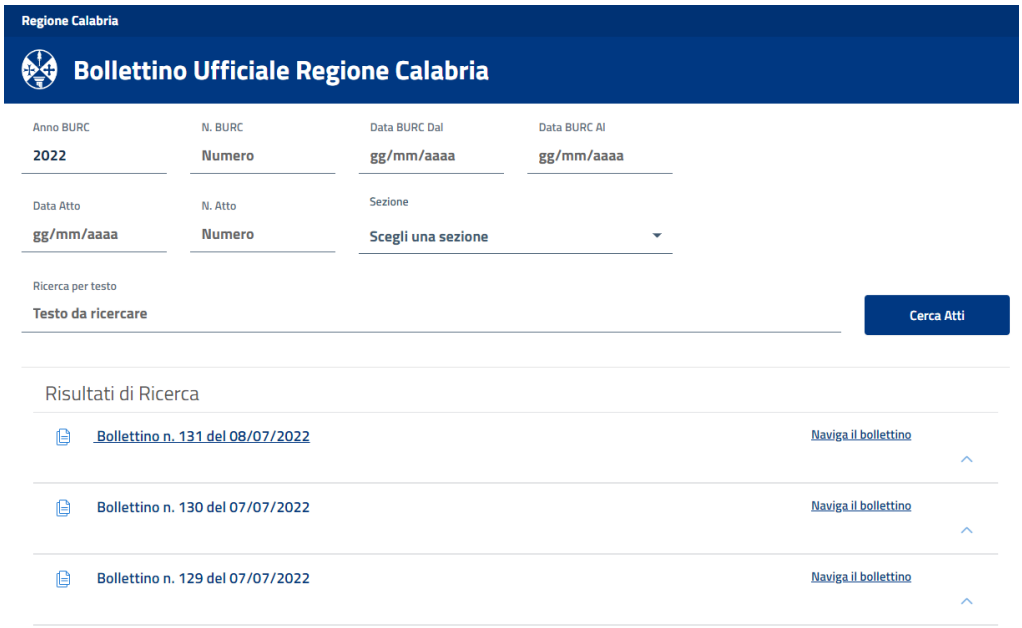
6.3. Siti esclusi.

Come si sarà notato ho fatto la scelta di escludere due siti dopo aver studiati e analizzati.

Il primo degli esclusi contiene i bollettini del BURC fino al 2019. Vista la decisione nel dare la precedenza ai bandi recenti ed attivi ho deciso di escludere questo sito dal lavoro di estrazione dati.

Il secondo sito invece effettua una ricerca molto estesa restituendo però solo documenti in PDF senza presentare nemmeno date o importi in vista.

Numerosi sono i metodi di fare uno scraping anche dei file PDF ma non è il tema di questo lavoro.



7. LAZIO

7.1. Bandi Europei

7.1.1. Analisi

Ad un primo sguardo il sito assomiglia molto al sito dei bandi europei della regione Calabria.

Anche in questo sito sono presenti i bandi prossimi all'apertura e i bandi aperti e, come nell'occasione precedente, li esploreremo tutti e due.

La differenza qui è che non abbiamo nessun pulsante da cliccare per la scelta dello stato, ma un menu "a tendina" da cui scegliere le opzioni.

```
▼<select name="stati" data-placeholder="Stato" class="ricerca-bandi-form-el-select select2-hidden-ac
 id="filtro-stati" data-select2-id="filtro-stati" tabindex="-1" aria-hidden="true">
   <option value data-select2-id="2"></option>
   <option value="1" data-select2-id="14"> Aperto </option>
   <option value="0" data-select2-id="15"> Chiuso </option>
   <option value="2" data-select2-id="16"> Prossima Apertura </option>
 </select>
▼<span class="select2 select2-container select2-container--default select2-container--below select2-
 --focus" dir="ltr" data-select2-id="1" style="width: 286.688px;">
  ▶<span class="selection">…</span> == $0
```

Le sue fattezze sono un po' particolari in quanto le opzioni selezionabili sono dentro il "select" mentre l'oggetto con cui si va ad interagire è dentro lo span.

Come nel sito della Calabria le varie anteprime portano alla pagina con il dettaglio del bando. Tuttavia mi accorgo, semplicemente passando sopra il mouse, che non tutta la superficie è cliccabile e che alcuni link sono fuorvianti rispetto ai nostri obiettivi.
I link che vogliamo sono nella foto e nel titolo il grassetto blu. Tra i due, la scelta su cosa usare è indifferente.

Ogni riquadro è un "article", l'immagine è all'interno di un div e così anche tutte le sezioni di testo del riquadro.

```
▼<div class="container-bandi">  grid  == $0
  ▶<article class="box-bandi">…</article>  grid
  ▶<article class="box-bandi">…</article>  grid
  ▶<article class="box-bandi">…</article>  grid
  ▶<article class="box-bandi">…</article>  grid
  ▶<article class="box-bandi">…</article>  grid
  ▶<article class="box-bandi">…</article>  grid
  ▶<article class="box-bandi">…</article>  grid
  ▶<article class="box-bandi">…</article>  grid
  ▶<article class="box-bandi">…</article>  grid
  ▶<article class="box-bandi">…</article>  grid
  ▶<article class="box-bandi">…</article>  grid
  ▶<article class="box-bandi">…</article>  grid
  ▶<article class="box-bandi">…</article>  grid
  </div>
```

Il cospicuo numero di risultati fa sì che siano disposti su più pagine. Si può passare alla pagina successiva premendo il bottone "SUCCESSIVI"

Il testo all'interno dei bandi in dettaglio, purtroppo, è fortemente disomogeneo, per quantità, per disposizione e anche per implementazione nella pagina.

In questo caso è impossibile creare uno script generico che collezioni dati e si dovrebbe intervenire pagina per pagina, ma la cosa in questo caso diventa assolutamente sconveniente.

Ho tuttavia raccolto i bandi aperti e di prossima apertura e alcune tra le informazioni che erano disponibili nella pagina.

7.1.2. Script

Creo un unico crawler sia per i bandi aperti che in prossima apertura. Vado ad osservare la variazione di identificativo che mi permetterà di distinguere i due. Nel menu a tendina è conveniente selezionare l'opzione attraverso il testo visualizzato o parte di esso, se non vi sono doppioni.

Alla prima esecuzione del crawler passerò quindi lo stato ad 'Aperto' mentre la seconda volta lo stato sarà 'Prossima Apertura' ovvero i due testi nella selezione a tendina che mi interessano.

```
selector = Select(browser.find_element(By.ID, "filtro-stati"))
selector.select_by_visible_text(stato)
browser.find_element(By.NAME, 'action').click()
```

Nell'ultima riga sto cliccando il tasto "Cerca" per confermare la mia scelta.

Una volta esauriti tutti i collegamenti nella pagina è ora di passare alla successiva. Per generalizzare, non sapendo quante pagine potrebbero esserci un giorno, creo un ciclo for perpetuo e inserisco come condizione di uscita forzata il momento in cui non ci sarà più nessun pulsante "SUCCESSIVI".

```
while True:
    try:
        browser.find_element(By.PARTIAL_LINK_TEXT, 'SUCCESSIVI').click()
        links_tmp = linkCollector1(browser)
        for ii in links_tmp:
            links.append(ii)
    except:
        break
```

Quando la ricerca dell'elemento andrà in errore, al posto di lanciare l'eccezione il comando break interrompe il loop.

Come già fatto in precedenza raccolto tutti i link per risparmiare tempo nell'esecuzione. Per la raccolta degli stessi ho preferito prelevarli dall'immagine, cercando gli elementi con xpath.

```
def linkCollector1(browser):
    listaEl = browser.find_elements(By.XPATH, "//div[@class='box-bandi-immagine']//a")
    links = []
    for ii in listaEl:
        links.append(ii.get_attribute('href'))
    return links
```

In questa occasione si nota un nuovo aspetto di xpath, ovvero la possibilità di definire un oggetto anche in base alla posizione nel codice, alla gerarchia.
In questo caso sto cercando un collegamento, a, entro il div di classe "box-bandi-immagine".

Questo aspetto aiuta in caso di elementi uguali con lo stesso id ma con oggetti diversi all'interno dando due o più elementi di riconoscimento.

```
def linkCollector1(browser):
    listaEl = browser.find_elements(By.XPATH, "//div[@class='box-bandi-immagine']//a")
    links = []
    for ii in listaEl:
        links.append(ii.get_attribute('href'))
    return links
```

Per estrarre i tag a fondo pagina, ho di nuovo usato la ricerca con xpath, che mi ha concesso di diversificare i vari gruppi.

I tag a fondo pagina sono stati recuperati con l'utilizzo dell'XPath completo, che descrive l'intero percorso attraverso i vari annidamenti del codice.
Questo poiché non si riuscivano a recuperare gli elementi in questione con un XPath relativo.

```
tipoAg = browser.find_elements(By.XPATH, "/html/body/div[1]/main/article/footer/span[1]/div")
tipoAg = toTxtList(tipoAg)
```

Il codice funziona e riesce ad analizzare il sito in maniera soddisfacente, generando un output di cui riporto una parte.

Titolo	categoria	Tipo Agevolazioni	Destinatari	Enti	url	Stato
MANIFESTAZIONE I	INTERNAZIONALIZZAZIONE	AIUTO	PICCOLE E MEDIE IMPRESE, STARTUP	CCIAA, LAZIO INNOVA S.P.A., REGIONE LAZIO	https://w	Aperto
CONTRIBUTI PER LA	CREATIVITÀ, CINEMA, TURISMO E CULTURA	FONDO PERDUTO	ASSOCIAZIONI, COOPERATIVE, ENTI, E	REGIONE LAZIO	https://w	Aperto
PON GOVERNANCE	GOVERNANCE	ALTRO	COMUNI, FROSINONE, LATINA, RIETI, F	REGIONE LAZIO	https://w	Aperto
CONTRIBUTO DI LIB	LAVORO, FORMAZIONE E INCLUSIONE	AIUTO	DONNE, FROSINONE, LATINA, RIETI, R(	LAZIO INNOVA S.P.A., REGIONE LAZIO	https://w	Aperto
PNRR – NEXT GENE	CREATIVITÀ, CINEMA, TURISMO E CULTURA	FONDO PERDUTO	ALTRE IMPRESE, ASSOCIAZIONI, ASSO	REGIONE LAZIO	https://w	Aperto
PO FEAMP, PROTEZ	IMPRESE AGRICOLE, DELLA PESCA E DELL'ACQUACOLTURA	ENTIIMPRESE E LIBERI PROFESSIONISTILATINAPESCATORIROMAVITERBO	REGIONE LAZIO		https://w	Aperto
CONTRIBUTI PER LA	COMMERCIO	AIUTO	ENTI LOCALI, FROSINONE, LATINA, RIE	LAZIO INNOVA S.P.A., REGIONE LAZIO	https://w	Aperto
PR FSE+ 2021-2027:	LAVORO, FORMAZIONE E INCLUSIONE	FONDO PERDUTO	- DI 18 ANNI, ASSOCIAZIONI, CITTADIN	REGIONE LAZIO	https://w	Aperto
SOVVENZIONI PER	CREATIVITÀ, CINEMA, TURISMO E CULTURA	AIUTO	FROSINONE, IMPRESE E LIBERI PROFESSIONISTI, LATINA, RIETI, ROMA, VITERBO		https://w	Aperto
PR FESR 2021-2027,	CREATIVITÀ, CINEMA, TURISMO E CULTURA	AIUTO	FROSINONE, LATINA, PICCOLE E MEDIE	LAZIO INNOVA S.P.A.	https://w	Aperto
PO FEAMP, CONTRI	IMPRESE AGRICOLE, DELLA PESCA E DELL'ACQUACOLTURA	AIUTO	IMPRESE E LIBERI PROFESSIONISTI, LAT	REGIONE LAZIO	https://w	Aperto
CATALOGO DELLE B	CREATIVITÀ, CINEMA, TURISMO E CULTURA	ALTRO	ENTI, FROSINONE, LATINA, RIETI, ROM	REGIONE LAZIO	https://w	Aperto
ATELIER ARTE BELLE	CREATIVITÀ, CINEMA, TURISMO E CULTURA	ALTRO	ENTI LOCALI	LAZIO INNOVA S.P.A., REGIONE LAZIO	https://w	Aperto
POR FSE, CONTRIBU	LAVORO, FORMAZIONE E INCLUSIONE	CONTRIBUTI UNA TANTUM	FROSINONE, LATINA, LAVORATORI DIP	REGIONE LAZIO	https://w	Aperto
LAZIODISCO, BAND	LAVORO, FORMAZIONE E INCLUSIONE	BORSE	FROSINONE, LATINA, RIETI, ROMA, STU	DISCO	https://w	Aperto
PO FEAMP 2014-20	IMPRESE AGRICOLE, DELLA PESCA E DELL'ACQUACOLTURA	ENTIFROSINONEIMPRESE E LIBERI PROFESSIONISTILATINAMICROIMPRESEPESCAT	REGIONE LAZIO		https://w	Aperto
BANDO DELLE IDEE	AMBIENTE E SOSTENIBILITÀCREATIVITÀ, CINEMA, TURISM	FONDO PERDUTO	ASSOCIAZIONI, ENTI, FROSINONE, LAT	LAZIOCREA S.P.A., REGIONE LAZIO	https://w	Aperto
PR FSE+, AVVISO PU	BENESSERE E SALUTE	VOUCHER	18-29 ANNI, CITTADINI, FROSINONE, L	ORGANISMO INTERMEDIO	https://w	Aperto
EMERGENZA COVID	BENESSERE E SALUTE	RIMBORSO	ASSOCIAZIONI DI PROMOZIONE SOCIA	LAZIOCREA S.P.A.	https://w	Aperto

In questa immagine si può notare come alla riga evidenziata, alle colonna Tipo Agevolazioni, Destinatari ed Enti vi siano degli errori.
Questo perché la pagina di quel singolo bando aveva una formattazione diversa rispetto alle altre su cui mi sono basato per sviluppare il codice.

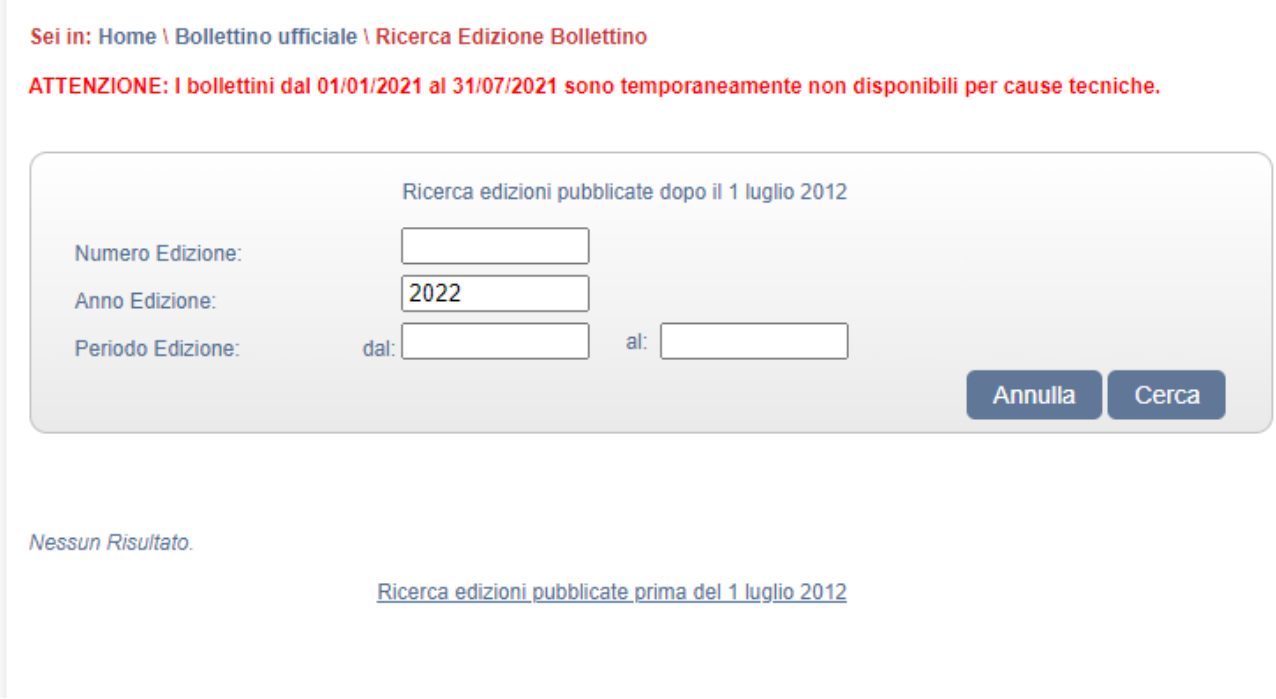
Questo è il limite maggiore di questa procedura, poiché non avendo un codice "intelligente" non è in grado di adattarsi e comprendere la variazione o l'errore.

7.2. BUR

7.2.1. Analisi

Dopo una veloce analisi del sito del BUR della regione Lazio noto con dispiacere che si profila una situazione speculare alla precedente, ovvero di una serie di pubblicazioni PDF la cui estrapolazione dati esime dalla consegna del lavoro.

Il sito, inoltre, non presenta alcun bando o avviso del 2022 e anzi, ci sono problemi con la visualizzazione dei bandi per l'anno 2021.

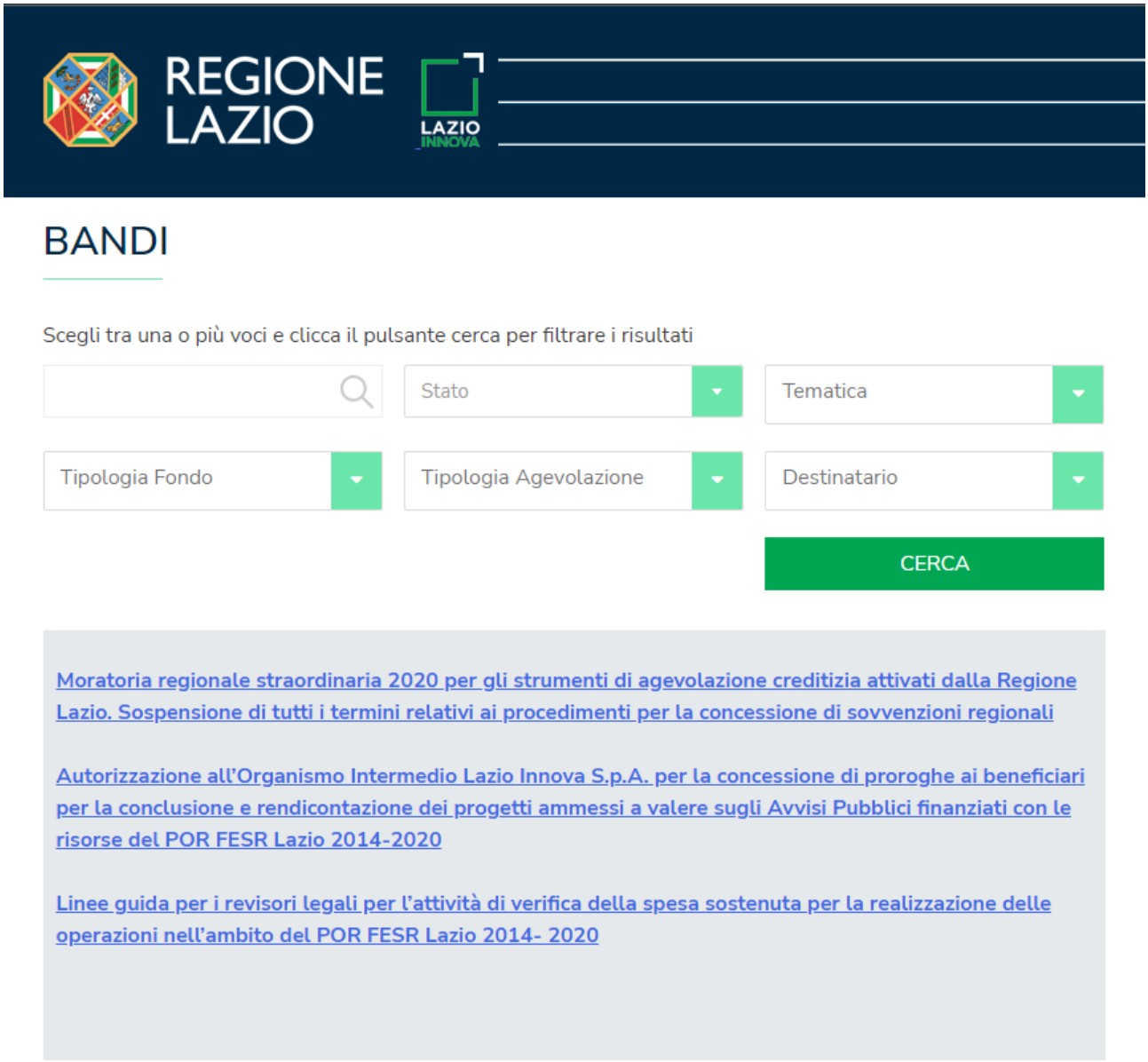


Come prima, non eseguirò alcun tentativo su questo sito, poiché non collima ne con i bandi che ritengo interessanti, ne mi consente una visualizzazione web degli stessi.

7.3 Lazio Innova

Una ricerca di altri siti da cui trarre informazioni sui bandi mi sono imbattuto in questo che non era nel progetto originale.

7.3.1 Analisi



Il sito si apre con questo form di ricerca identico quello presente nel sito precedente (7.1).

Nel selezionare lo stato mi presenta le anteprime dei bandi una al di sotto della precedente, con evidenziati nient'altro che i tag che saranno riportati poi anche in fondo alla pagina del bando completo.

In basso troviamo un pulsante che consente di cambiare pagina per visualizzare altri bandi individuati.

APERTO lunedì 18 Settembre 2017

Voucher di Garanzia

TEMATICHE	ACCESSO AL CREDITO
TIPOLOGIE FONDI	EUROPEI
TIPOLOGIE AGEVOLAZIONI	A FONDO PERDUTO, GARANZIE
DESTINATARI	MEDIE IMPRESE, MICRO-IMPRESE, PERSONE FISICHE, PICCOLE IMPRESE

APERTO martedì 31 Luglio 2018

Innova Venture

TEMATICHE	AEROSPAZIO, AGRIFOOD, AUDIOVISIVO, CINEMA, CREAZIONE DI IMPRESA, CULTURA, ECONOMIA CIRCOLARE, ECONOMIA DEL MARE, EDITORIA, GAMING, GREEN ECONOMY, ICT, INNOVAZIONE, MOBILITÀ, MODA, PRE-SEED, RICERCA / R&S, S3, SCALE-UP, SCIENZE DELLA VITA
TIPOLOGIE FONDI	EUROPEI
TIPOLOGIE AGEVOLAZIONI	EQUITY CROWDFUNDING, VENTURE CAPITAL
DESTINATARI	IMPRESE COSTITUENDE, MEDIE IMPRESE, MICRO-IMPRESE, PICCOLE IMPRESE, STARTUP E SPINOFF

← BANDI MENO RECENTI

Di nuovo, come nel sito lazioeuropa.it, mi trovo nella situazione in cui il bando è presentato in maniera molto discorsiva e poco schematica.

Noto poi che, come nel sito "lazioeuropa" ogni pagina è gestita ed ha una composizione ed impaginazione a sé stante.
Questo va unito al fatto che nemmeno date e importi in euro hanno la stessa formattazione.

Si trova scritta un importo completamente in cifre, ad esempio €456.000 (valore fittizio per esemplificare) come si può trovare scritta come "465 mila euro".
Il discorso è lo stesso per le date, vengono presentate in formato gg/mm/yyyy oppure come gg, mese in lettere e yyyy. Esempi di ciò sono evidenziati nell'immagine estratta dal sito.

L'insieme di questi fattori rende difficile o impossibile una corretta estrazione dati con i mezzi a mia conoscenza, pertanto, come nel sito 7.1 mi limiterò ad estrarre titolo, stato, tag e url.

CARATTERISTICHE

- Dotazione del fondo **€ 21,8 milioni**
- Target: **Imprese Ammissibili con sede operativa nel Lazio** (esistente o prevista)
- Settori ammissibili: **Tutti** (salvo specifiche e limitate esclusioni)
- Fasi di investimento: **seed, start-up. scale-up, espansione**
- limiti di investimento:
  - Investimento Iniziale **minimo € 150.000 e massimo € 1,4 milioni**
  - Investimento massimo (Iniziale + Follow on): **€ 2,5 milioni**
- Durata dell'investimento prevista: **fino a 8 anni**

PRESENTAZIONE DELLE PROPOSTE

La versione aggiornata dell'Avviso Pubblico di INNOVA Venture è stata pubblicata sul **BURL n. 117 del 16 dicembre 2021** e la presentazione delle relative Proposte è a partire **dalla medesima data fino alle ore 12.00 del 31 luglio 2023** (salvo esaurimento precedente dei fondi).

La presentazione delle Proposte di investimento deve avvenire **esclusivamente attraverso la piattaforma GeCoWEB Plus** (gecoplus.lazioinnova.it) compilando l'apposito formulario di INNOVA Venture ivi presente **e caricando tassativamente sulla medesima, prima della finalizzazione del formulario e della firma digitale, i seguenti documenti**:

1. il business plan completo a 5 anni (inteso come **documento unitario e completo comprensivo della parte descrittiva e numerica del progetto imprenditoriale**, come da Appendice 3 dell'Invito);
2. **le manifestazioni di interesse da parte di Co-investitori Privati indipendenti** interessati a investire insieme a Innova Venture (redatte con il modello 1, se non vincolanti, o modello 2, se vincolanti, di cui all'avviso).

7.3.2 Script

Script similare nelle fattezze a quello di “lazioeuropa” per il fatto che i due siti hanno molti elementi caratteristici comuni.

Riporto alcune parti di codice riguardanti il recupero dei tag che, come prima, utilizza lo XPath completo in sostituzione a quello relativo per riuscire a definire con precisione gli elementi di interesse.

Metto in evidenza la differenza tra XPath relativo, il primo, che individua l’elemento utilizzando solo l’ultima parte della sequenza dei tag annidati e XPath complessivo che effettua la ricerca con tutta la sequenza completa.

```
bando['Titolo'] = browser.find_element(By.XPATH, "//header[@class='entry-header']").text
```

```
tematiche = browser.find_elements(By.XPATH, "html/body/div[1]/main/article/footer/span[1]/div/a")
```

Riporto inoltre l’estratto del codice che ho usato per girare le pagine.
Anche in questo sito ho preferito estrarre tutti i link dei bandi presenti in una pagina con il metodo linkExtractor da me creato su specifiche della pagina interessata, per poi visitarli uno per uno.

Per girare le pagine, non sapendo quante ve ne siano, creo un loop perpetuo con un ciclo while True che, in ogni pagina di risultati aperta, cerca di premere il pulsante “BANDI MENO RECENTI”. Se riesce si effettua la raccolta di link nella nuova pagina, se non riesce si origina l’eccezione NoSuchElementException, catturata dal blocco try-except, che porta alla terminazione del while ed al proseguire del codice.

```
while True:
    try:
        browser.find_element(By.PARTIAL_LINK_TEXT, 'BANDI MENO RECENTI').click()
        links.append(linkCollector2(browser))
    except:
        break
```

Il codice funziona correttamente ed estrapola tutti i dati richiesti senza cadere in errore.
Riporto alcuni dei risultati qui.

Titolo	url	Tematiche	Tipologie Fondi	Tipo Agevolazioni	Destinatari	Stato
Lazio Cinema International 2022	https://w	CINEMA	EUROPEI	A FONDO PERDUTO	MEDIE IMPRESE, PICCOLE IMPRESE	Aperto
Zagarolo Game House	https://w	GAMING, ICT	EUROPEI	PREMI IN DENARO E SERVIZI	GIOVANI, PERSONE FISICHE, SCUOLE E STUDENTI, STARTUP E SPINOFF	Aperto
Start your Energy Revolution	https://w	DIGITALIZZAZIONE, GREEN ECONOMY, INNOVAZIONE, SOSTENIBILITÀ	EUROPEI, PRIVATI	PREMI IN DENARO E SERVIZI	MEDIE IMPRESE, MICRO-IMPRESE, PICCOLE IMPRESE, STARTUP E SPINOFF	Aperto
POR FESR, Pre Seed Plus	https://w	INNOVAZIONE	EUROPEI	A FONDO PERDUTO	STARTUP E SPINOFF	Aperto
Boost your Ideas 2022 – Dai la spinta g	https://w	CULTURA, DIGITALIZZAZIONE, TRANSIZIONE ECOLOGICA, TURISMO	EUROPEI	PREMI IN DENARO E SERVIZI, SERVIZI	GIOVANI, IMPRESE, PERSONE FISICHE, STARTUP E SPINOFF	Aperto
Start Up DTC Lazio	https://w	CREAZIONE DI IMPRESA, CULTURA, ICT, INNOVAZIONE, PARI OPPORTU	NAZIONALI, REGIONALI	A FONDO PERDUTO	MICRO-IMPRESE, PICCOLE IMPRESE, STARTUP E SPINOFF	Aperto
Fondo rotativo regionale per il recupe	https://w	ACCESSO AL CREDITO, COOPERAZIONE, CREAZIONE DI IMPRESA, SOSTE	REGIONALI	FINANZIAMENTI AGEVOLATI	IMPRESE COSTITUENDE, MEDIE IMPRESE, MICRO-IMPRESE, PICCOLE IMPRESE	Aperto
ESA BIC (Business Incubation Centre) I	https://w	AEROSPAZIO, INNOVAZIONE	EUROPEI	SERVIZI	IMPRESE COSTITUENDE, STARTUP E SPINOFF, UNIVERSITÀ E RICERCA	Aperto
Voucher di Garanzia	https://w	ACCESSO AL CREDITO	EUROPEI	A FONDO PERDUTO, GARANZIE	MEDIE IMPRESE, MICRO-IMPRESE, PERSONE FISICHE, PICCOLE IMPRESE	Aperto
Innova Venture	https://w	AEROSPAZIO, AGRIFOOD, AUDIOVISIVO, CINEMA, CREAZIONE DI IMPRE	EUROPEI	EQUITY CROWDFUNDING, VENTURE CAPITAL	IMPRESE COSTITUENDE, MEDIE IMPRESE, MICRO-IMPRESE, PICCOLE IMPRESE, STARTUP E SPINOFF	Aperto

8 MOLISE

Al netto di tutti i tentativi eseguiti per estrapolare bandi dai vari siti della regione, ho rilevato come la regione Molise abbia prodotto i risultati più deludenti tra tutte le regioni analizzate.
Ho trovato siti che non venivano aggiornati da anni, realizzati male e, soprattutto con pochi contenuti.

Questo è probabilmente dovuto a dimensione e popolazione del Molise, molto ridotte rispetto alle altre regioni studiate. Questo porta ad un impoverimento anche e soprattutto a livello di iniziative sociali, bandi e concorsi.

8.1 Bandi europei

Sul sito dei bandi europei ho trovato solo bandi chiusi, il cui tempo per la partecipazione è quindi terminato. Ho comunque analizzato il sito per capire se fosse possibile, nell'eventualità di riapertura di alcuni bandi, eseguirne lo scraping.

La pagina si presenta con un elenco delle anteprime dei bandi, con link che apre la pagina interessata all'interno del titolo.

AVVISI E BANDI

POC MOLISE 2014-2020 ASSE 8 APPROVAZIONE "BANDO DI CONCORSO INTEGRATIVO PER L'ASSEGNAZIONE DI BORSE DI STUDIO PER IL DIRITTO ALLO STUDIO UNIVERSITARIO A.A. 2021/2022.

Vai al bando

Chiuso

AZIONE 6.1.1 - AVVISO PUBBLICO PER LA PRESENTAZIONE DI OPERAZIONI FORMATIVE FINALIZZATE A FAVORIRE L'INSERIMENTO E REINSERIMENTO NEL MERCATO DEL LAVORO

A partire dalle ore 17,00 del giorno 13 marzo 2021 sono sopspesi i termini di presentazione delle istanze di partecipazione all'avvisio. Tutti i dettagli nella DD 20 del 103.21

Chiuso

COVID 19 - AZIONE 3.1.1 - AVVISO PUBBLICO PER CONCEDERE AGEVOLAZIONI PER SUPPORTARE LE IMPRESE E FAVORIRE LA RIPRESA PRODUTTIVA

A piè di pagina è presente il selettore della pagina desiderata. Si possono scorrere le pagine dalla prima all'ultima come eseguito in precedenza, ovvero cercando di cliccare il pulsante "Seguente" all'interno di un while perpetuo, interrotto dall'eccezione in caso di incapacità di individuare il pulsante sulla falsariga degli script usati in precedenza.

All’interno del dettaglio dei vari bandi noto con dispiacere che, seppur quasi tutte le pagine siano standardizzate nei contenuti e nei paragrafi, alcune sono diverse.
Sicuramente procedendo e adattando il codice alle pagine più comuni si avranno quindi delle eccezioni da gestire o delle inesattezze al momento dell’output dei risultati prodotti.

AZIONE 6.1.1 - AVVISO PUBBLICO PER LA PRESENTAZIONE DI OPERAZIONI FORMATIVE FINALIZZATE A FAVORIRE L’INSERIMENTO E REINSERIMENTO NEL MERCATO DEL LAVORO

- A partire dalle ore 17,00 del giorno 13 marzo 2021 sono sopspesi i termini di presentazione delle istanze di partecipazione all’avvisio. Tutti i dettagli nella DD 20 del 103.21

- **A partire dalle ore 10,30 del giorno 16 febbraio 2021 è possibile avviare le attività di precaricamento per la predisposizione della candidatura e di tutta la documentazione da allegare**. Il sistema MoSEM consentirà ai soggetti proponenti di entrare nel proprio profilo, caricare la proposta progettuale e consolidare le informazioni (il consolidamento avviene con il cambiamento dello stato della proposta, passando dallo stato di "bozza" a quello di "consolidata"). Restano confermeti i termini di aprtura dello sportello: **dalle ore 8.00 del 25 febbraio 2021 e fino alle ore 10.00 del 30 arile 2021.**

- L’Avviso pubblico è rivolto agli organismi di formazione accreditati dalla Regione Molise ed è finalizzato a favorire l’impiego o il reimpiego di persone in cerca di lavoro mediante interventi formativi e Work Experience presso le imprese del territorio che evidenzieranno il loro fabbisogno professionale e si dichiareranno disponibili ad ospitare gli allievi in formazione, creando un collegamento tra formazione e azienda. **Le candidature potranno essere presentate a partire dalle ore 8.00 del 25 febbraio 2021 e fino alle ore 10.00 del 30 arile 2021.**

OBIETTIVO:

Favorire l’impiego o il reimpiego di persone in cerca di lavoro mediante interventi formativi e un’esperienza di lavoro guidata (Work Experience) presso imprese del territorio che evidenzino il loro fabbisogno professionale e si dichiarino disponibili ad ospitare gli allievi in formazione, creando un collegamento tra formazione e azienda.

CHI PUO' PARTECIPARE:

Gli Organismi di formazione accreditati dalla Regione Molise, ai sensi della Deliberazione di Giunta regionale n. 533 del 08/08/2012 e ss .mm. e ii. alla data di presentazione dell’istanza di partecipazione, in forma singola (non sono ammesse ATI/ATS).

DESTINATARI:

I destinatari delle azioni a valere sul presente Avviso sono le persone disoccupate/inoccupate, beneficiari e non di prestazioni di sostegno al reddito, di età compresa tra i 18 e i 65 anni, risidenti o domiciliate nella Regione Molise.

COME PARTECIPARE:

Le domande di partecipazione al presente Avviso devono essere compilate, a pena la loro inammissibilità, esclusivamente per via telematica tramite la procedura informatica MoSEM messa

Questo è come si presenta la tipologia più comune di pagina. Il titolo è un elemento con tag h1, tag che contraddistingue i titoli.

```
▼<h1 class="title" id="page-title">
   "AZIONE 6.1.1 - AVVISO PUBBLICO PER LA PRESENTAZIONE DI OPERAZIONI FORMATIVE FINALIZZATE A FAVORIRE
   L'INSERIMENTO E REINSERIMENTO NEL MERCATO DEL LAVORO"
 </h1>
```

Il resto della pagina è contenuto in una serie di div annidati uno dentro l'altro fino alla parte inerente alle sezioni in cui è suddiviso il testo, che presentano i dettagli salienti del bando in maniera precisa e sintetica.

Seppur somigli decisamente all'impaginazione realizzata nell'equivalente sito della regione Calabria, a livello di struttura di HTML è molto diverso, e più difficile da analizzare.

Le sezioni sono suddivise in una serie di paragrafi, identificati da tag "p", come si vede in foto. Tuttavia il problema si presenta in quanto titolo del paragrafo e paragrafo stesso vengono trattati come due paragrafi distinti.

```
▶<p class="rtejustify">…</p>
▶<p class="rtejustify">…</p>
▶<p class="rtejustify">…</p>
▶<p class="rtejustify">…</p>
▶<p class="rtejustify">…</p>
▶<p class="rtejustify">…</p>
▶<p class="rtejustify">…</p>
▶<p class="rtejustify">…</p>
▶<p class="rtejustify">…</p>
▶<p class="rtejustify">…</p>
▶<p class="rtejustify">…</p>
▶<p class="rtejustify">…</p>
▶<p class="rtejustify">…</p>
▶<p class="rtejustify">…</p>
▶<p class="rtejustify">…</p>
 <p class="rtejustify">30 aprile 2021 - ore 10.00</p>
▶<p class="rtejustify">…</p>
 <p class="rtejustify">Help Desk: 08744291</p>
▶<p class="rtejustify">…</p>
```

Realizzare un codice che unisca titolo e paragrafo insieme rischia di essere molto vulnerabile ad errori, soprattutto in caso di lievi modifiche da parte del webmaster.

9. REVISIONE TESI PRECEDENTI DEI COLLEGHI

Rispetto al periodo in cui i miei colleghi hanno sviluppato le loro tesi vi è stato un importante aggiornamento della libreria “selenium”; i metodi che consentono di individuare elementi nella pagina sono stati modificati. La chiamata è stata unificata e si differenzia per il parametro che viene loro passato. La dicitura find_element_by_* non è più riconosciuta e va sostituita con find_element(By.*, *elemento_da_individuare*). Ometterò quindi che, in ogni tesi, sono andato a sostituire le chiamate al metodo ogni volta che esso viene invocato per rendere il codice nuovamente eseguibile.

9.1 Miola Giacomo – Veneto, Friuli-Venezia Giulia e Trentino.

Nella revisione il codice funziona egregiamente. Ho trovato solo un intoppo dato da una modifica alla legge sulla privacy. Dal 9 gennaio 2022 infatti la finestra o popup di accettazione dei cookies devono obbligatoriamente rendere la pagina non navigabile fino all’accettazione degli stessi ed alla conseguente chiusura della pagina. Com’è noto molte pagine devono ancora adeguarsi, infatti ho riscontrato questo inconveniente solo nella pagina dei bandi del Trentino disponibile al seguente link: https://trentinosviluppo.it/it/Principale/Bandi_e_Appalti

Ho risolto la questione nel modo seguente, implementando un piccolo script che premesse il tasto di accettazione. La difficoltà è dovuta al fatto che il popup in questione viene caricato al di sopra della pagina e, per individuarlo, va analizzata la sezione di codice definita “shadowRoot”, ombra, ovvero una sezione che contiene codice di popup e finestre destinate ad apparire in un livello sovrapposto al livello della pagina e, successivamente, a scomparire.
Eseguito questo script il popup scompare ed il codice gira correttamente.

```
#shadow-root (open) == $0
  <div data-nosnippet="1" data-testid="uc-app-container">
    <style data-styled="active" data-styled-version="5.3.3"></style>
    <div class="sc-bdvvtL 1jqtvs">
      <div id="focus-lock-id">
        <div id="focus-lock-id">
          <div data-testid="uc-overlay" class="sc-AjmGg jqYUbp"></div>
          <div class="sc-pVTFL gsAnRF"> flex
            <div lang="it" role="dialog" aria-modal="true" aria-labelledby="uc-heading-title" tabindex="0" id="uc-ce
            nter-container" data-testid="uc-default-wall" class="sc-jrQzAO dAiVdG"> flex
              <div tabindex="-1" data-testid="uc-container" class="sc-ksdxgE LHaBT">…</div> flex
              <div class="sc-bYoBSM egarKh">
                <div data-testid="uc-footer" class="sc-kLwhqv fslbnd">
                  <div class="sc-dlVxhl bEDIID"> flex
                    <div data-testid="uc-buttons-container" class="sc-kfPuZi jQTRpX" style="margin: 0px -6px;"> flex
                      <button role="button" data-testid="uc-save-button" class="sc-gsDKAQ ewMBCj" style="margin: 0px
                      6px;">Salva impostazioni</button> flex
                      <button role="button" data-testid="uc-deny-all-button" class="sc-gsDKAQ jrAeAK" style="margin:
                      0px 6px;">Rifiuta</button> flex
                      <button role="button" data-testid="uc-accept-all-button" class="sc-gsDKAQ IIehq" style="margin:
                      0px 6px;">Accetta tutti</button> flex
                    </div>
                  </div>
                <div class="sc-jgrJnh cDLcRi">…</div> flex
```

```
#AGGIUNTE
time.sleep(2) #Attesa caricamento popup dei cookies
try: #accettazione cookies
    chrome.execute_script('''return document.querySelector('div#usercentrics-root').shadowRoot.
    querySelector('button[data-testid="uc-accept-all-button"]')''').click()
except:
    pass
```

9.2 Walkuski Pavel – Lombardia, Piemonte e Valle D'Aosta

Un problema si presenta nella parte di codice che analizza il sito https://bandi.regione.piemonte.it/ in quanto vi è la variazione nel nome di una classe, da "cardBandiStato" a "cardRp cardBandiStato" che ho corretto.

9.3 Crociani Riccardo – Marche, Toscana e Umbria

Con questa tesi ho incontrato un problema differente, il collega ha infatti usato una diversa libreria e ho dovuto apprenderne quantomeno le basi per comprendere il codice.
Non è stato tuttavia troppo difficile essendo ormai addentro al modus operandi del web scraping, mi sono trovato in leggera difficoltà per quanto riguarda l'apprendere il modo di ragionare ed impostare il programma diverso dal mio.

Nel sito https://www.regione.marche.it/Entra-in-Regione/Bandi ho dovuto apportare una modifica al metodo di collezione dei link alle pagine dei bandi poiché restituiva una lista vuota.

```
pagine = soup.find_all('div', {"class":"row table-hover"})
#print(pagine)   #Sezione con tutti i link alle pagine
#lista_tag_a = pagine.find_all("a")
for div in pagine:
    a = div.find("a")
    lista_pagine.append(a.get('href'))    #Non ci sono controlli da fare perché tutti i link in quel ta
```

Una successiva modifica si è resa necessaria nella fase di richiesta delle pagine in quanto risultava mancante la prima parte del link

```
for pagina_i in lista_pagine:
    html_pagina_i = requests.get('https://regione.marche.it'+pagina_i).text
    soup_pagina = bs(html_pagina_i, 'html.parser')
```

Il codice scritto per le altre due regioni si è rivelato essere funzionante.

9.4 Forzan Riccardo – Basilicata, Puglia, Sardegna e Sicilia e Spina Lorenzo – Abruzzo, Emilia-Romagna, Campania e Liguria.

Ho avuto il piacere di suddividermi il lavoro con questi due colleghi all'inizio del mio percorso di tesi.
Essendo le loro tesi più recenti di quelle prese in esame precedentemente, il loro codice non risulta essere affetto da particolari problemi dovuti alla variazione dei siti. Come per gli altri è affetto dal cambiamento nel modo di invocazione dei metodi.

10. CONCLUSIONI

Ho molto apprezzato questo percorso di tesi, che mi ha posto di fronte numerose volte alla necessità di escogitare soluzioni per rimediare ai diversi problemi che mi si sono posti davanti. Non tutto è andato a buon fine, alcune estrazioni si sono rivelate difficoltose o addirittura impossibili e questo mi è stato di esempio nel vedere come un lavoro non sempre porti ai risultati sperati.

Mi sono inoltre scontrato con la difficoltà nell'analizzare codice non scritto da me, nell'entrare nel modo di ragionare di alcuni colleghi, marginalmente diverso dal mio, ma che comunque richiede tempo e applicazione.

Rendo disponibile il codice della tesi online su GitHub oltre che nelle successive pagine.

https://github.com/massiagostini/Tesi/

11 CODICE

11.1 ws_calabria.py – Regione Calabria

```
import time
import pandas as pd
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from selenium.common.exceptions import NoSuchElementException
from datetime import date, datetime
from selenium.webdriver.common.keys import Keys
def scraper1(browser):
    bando = {}
    bando['title'] = browser.find_element(By.XPATH, "//h1[@class='entry-
title ']").text
    bando['url'] = browser.current_url
    specs_h = browser.find_elements(By.XPATH, "//h3[@class='cem-description-
title']")
    specs_b = browser.find_elements(By.XPATH, "//div[@class='cem-text-
description cem-overflow-anywhere']")
    for ii in range(len(specs_h)):
        bando[specs_h[ii].text] = specs_b[ii].text

    return bando
def scraper2(browser):
    bando = {}
    type = browser.find_element(By.XPATH,
"//span[@class='txt_red_bold']").text
    if type != 'Bando di Gara':
        return
    tmp = browser.find_elements(By.TAG_NAME, 'p')
    date_tmp = browser.find_elements(By.XPATH, "//span[@class='txt_red']")
    bando['Scadenza'] = date_tmp[0].text
    bando['Pubblicazione'] = date_tmp[1].text
    bando['Descrizione']=tmp[2].text
    for ii in range(len(tmp)-3):
        splitted = tmp[ii+3].text.split(': ')
        bando[splitted[0]] = splitted[1]
    return bando

def crawler1(browser, type):
    actions = ActionChains(browser)
    bandi=[]
    catselector = browser.find_element(By.ID, type)
    actions.move_to_element(catselector).click(catselector).perform()
    try:
        lista = browser.find_elements(By.XPATH, "//div[@class='col-lg-4 col-
lg-4 col-12 p-0 p-md-2 mb-4']")
    except:
        print('bandi non trovati/pagina irraggiungibile')
    for ii in lista:
        actions.move_to_element(ii).click(ii).perform()
        tmp_bando = scraper1(browser)
        bandi.append(tmp_bando)
        browser.back()
    return(bandi)
def crawler2(browser):
```

```
    actions = ActionChains(browser)
    bandi = []
    date = browser.find_element(By.NAME, "data_scad_dal")
    date.send_keys(datetime.today().strftime('%d/%m/%Y'))
    date.send_keys(Keys.ENTER)
    links = linkCollector2(browser, 'Visualizza ')
    while True:
        try:
            nextPage = browser.find_element(By.PARTIAL_LINK_TEXT,
'Precedenti')
            print(nextPage.text)
            actions.move_to_element(nextPage).perform()
            nextPage.click()
            for ii in linkCollector2(browser, 'Visualizza '):
                links.append(ii)
        except:
            break
    for ii in links:
        browser.get(ii)
        bando_tmp = scraper2(browser)
        if bando_tmp != None:
         bandi.append(bando_tmp)
    return bandi


def linkCollector2(browser, partLinkText):
    list_tmp = browser.find_elements(By.PARTIAL_LINK_TEXT, 'Visualizza ')
    links = []
    for ii in list_tmp:
        links.append(ii.get_attribute('href'))
    return links

def calEuropa():
    bandi = []
    browser = webdriver.Chrome()
    browser.get('http://calabriaeuropa.regione.calabria.it/website//bandi/')
    try:
        browser.find_element(By.ID, 'wt-cli-accept-all-btn').click()
#cookies
    except:
        pass
    prepublished = crawler1(browser, 'prepublished')
    browser.get('http://calabriaeuropa.regione.calabria.it/website//bandi/')
    try:
        browser.find_element(By.ID, 'wt-cli-accept-all-btn').click()
#cookies
    except:
        pass
    published = crawler1(browser, 'published')
    for ii in prepublished:
        bandi.append(ii)
    for ii in published:
        bandi.append(ii)
    browser.quit()
    return bandi
def calReg():
    browser = webdriver.Chrome()
    browser.get('https://www.regione.calabria.it/website/bandiregione/')
    time.sleep(2)
```

```
    try:
        browser.find_element(By.PARTIAL_LINK_TEXT, 'Accetto!').click()
    except:
        pass
    bandi = crawler2(browser)
    return bandi
def excelOut(bandi, nomeFile):
    excel = pd.DataFrame(bandi) #generate dataframe from list
    excel.to_excel(nomeFile+".xlsx", index = False) #export dataframe in
excel
if __name__=='__main__':
    bandiCalEu = calEuropa()
    bandiCalReg = calReg()
    excelOut(bandiCalReg, 'regione.calabria.it')
    excelOut(bandiCalEu, 'calabriaeuropa')
```

11.2 ws_lazio.py – Regione Lazio

```
from cmd import PROMPT
import pandas as pd
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import Select
def toTxtList(lista):
    output = ''
    for ii in lista:
        if ii != lista[-1]:
            output = output + ii.text + ', '
        else:
            output = output + ii.text
    return output
def linkCollector1(browser):
    listaEl = browser.find_elements(By.XPATH, "//div[@class='box-bandi-
immagine']//a")
    links = []
    for ii in listaEl:
        links.append(ii.get_attribute('href'))
    return links
def excelOut(bandi, nomeFile):
    excel = pd.DataFrame(bandi) #generate dataframe from list
    excel.to_excel(nomeFile+".xlsx", index = False) #export dataframe in
excel
def scraper1(browser):
    bando = {}
    bando['title'] = browser.find_element(By.XPATH, "//div[@class='single-
bandi-titolo']").text
    bando['categoria'] = browser.find_element(By.XPATH,
"//div[@class='single-bandi-categoria']").text
    tipoAg = browser.find_elements(By.XPATH,
"/html/body/div[1]/main/article/footer/span[1]/div")
    tipoAg = toTxtList(tipoAg)
    bando['Tipo Agevolazioni'] = tipoAg
    dest = browser.find_elements(By.XPATH,
"/html/body/div[1]/main/article/footer/span[2]/div/a")
    dest = toTxtList(dest)
    bando['Destinatari'] = dest
    enti = browser.find_elements(By.XPATH,
"html/body/div[1]/main/article/footer/span[3]/div/a")
    enti = toTxtList(enti)
    bando['Enti'] = enti
    bando['url'] = browser.current_url
    return bando

def crawler1(browser, stato):
    actions = ActionChains(browser)
    bandi=[]
    selector = Select(browser.find_element(By.ID, "filtro-stati"))
    selector.select_by_visible_text(stato)
    browser.find_element(By.NAME, 'action').click()
    links = linkCollector1(browser)
    while True:
```

```
        try:
            browser.find_element(By.PARTIAL_LINK_TEXT, 'SUCCESSIVI').click()
            links_tmp = linkCollector1(browser)
            for ii in links_tmp:
                links.append(ii)
        except:
            break
    for ii in links:
        browser.get(ii)
        bandi.append(scraper1(browser))
    return bandi
def lazEuropa():
    bandi = []
    browser = webdriver.Chrome()
    browser.get('http://www.lazioeuropa.it/bandi/')
    try:
        browser.find_element(By.PARTIAL_LINK_TEXT, 'Accetta tutto').click()
    except:
        pass
    aperti = crawler1(browser, 'Aperto')
    browser.get('http://www.lazioeuropa.it/bandi/')
    try:
        browser.find_element(By.PARTIAL_LINK_TEXT, 'Accetta tutto').click()
    except:
        pass
    proxApertura = crawler1(browser, 'Prossima Apertura')
    bandi = []
    for ii in aperti:
        ii['Stato']='Aperto'
        bandi.append(ii)
    for jj in proxApertura:
        jj['Stato']='Prossima Apertura'
        bandi.append(jj)
    browser.quit()
    return bandi
def linkCollector2(browser):
    lista_tmp =  browser.find_elements(By.XPATH, "//header//h2//a")
    output = []
    for ii in lista_tmp:
        output.append(ii.get_attribute('href'))
    return output
def scraper2(browser):
    bando = {}
    bando['Titolo'] = browser.find_element(By.XPATH,
"//header[@class='entry-header']").text
    bando['url'] = browser.current_url
    tematiche = browser.find_elements(By.XPATH,
"html/body/div[1]/main/article/footer/span[1]/div/a")
    tematiche = toTxtList(tematiche)
    bando['Tematiche'] = tematiche
    tipoFondi = browser.find_elements(By.XPATH,
"html/body/div[1]/main/article/footer/span[2]/div/a")
    tipoFondi = toTxtList(tipoFondi)
    bando['Tipologie Fondi'] = tipoFondi
    tipoAg = browser.find_elements(By.XPATH,
"html/body/div[1]/main/article/footer/span[3]/div/a")
    tipoAg = toTxtList(tipoAg)
    bando['Tipo Agevolazioni'] = tipoAg
```

```
    dest = browser.find_elements(By.XPATH,
"html/body/div[1]/main/article/footer/span[4]/div/a")
    dest = toTxtList(dest)
    bando['Destinatari'] = dest
    return bando

def crawler2(browser, stato):
    actions = ActionChains(browser)
    bandi=[]
    selector = Select(browser.find_element(By.ID, "filtro-stati"))
    selector.select_by_visible_text(stato)
    browser.find_element(By.NAME, 'action').click()
    links = linkCollector2(browser)
    while True:
        try:
            browser.find_element(By.PARTIAL_LINK_TEXT, 'BANDI MENO
RECENTI').click()
            links.append(linkCollector2(browser))
        except:
            break
    for ii in links:
        browser.get(ii)
        bandi.append(scraper2(browser))
    return bandi
def lazioInnova():
    browser = webdriver.Chrome()
    browser.get('http://www.lazioinnova.it/bandi/')
    aperti = crawler2(browser, 'Aperto')
    browser.get('http://www.lazioinnova.it/bandi/')
    proxApertura = crawler2(browser, 'Prossima Apertura')
    bandi = []
    for ii in aperti:
        ii['Stato']='Aperto'
        bandi.append(ii)
    for jj in proxApertura:
        jj['Stato']='Prossima Apertura'
        bandi.append(jj)
    browser.quit()
    return bandi
if __name__=='__main__':
    #lazEu = lazEuropa()
    #excelOut(lazEu, 'lazioeuropa')
    lazInn = lazioInnova()
    excelOut(lazInn, 'lazioinnova')
```

11.3 ws_molise.py – Regione Molise

```
from cmd import PROMPT
from lib2to3.pgen2 import driver
import time
import pandas as pd
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.keys import Keys
def scraper(browser):
    bando = {}
    titolo = browser.find_element(By.XPATH, "//section[@id='post-
content']//h1").text
    if titolo == 'TEST':
        return
    bando['Titolo'] = titolo
    bando['url'] = browser.current_url
    bando['Descrizione'] = browser.find_element(By.XPATH,
"//div[@class='field field-name-body field-type-text-with-summary field-
label-hidden']")
    bando['Misura'] = browser.find_element(By.XPATH, "//div[@class='field
field-name-field-relazione-misura field-type-taxonomy-term-reference field-
label-above']//div[@class=field-item even']")
    print(bando)
    return bando
def crawler(browser):
    actions = ActionChains(browser)
    bandi = []
    lista = browser.find_elements(By.XPATH, "//div[@class='field-item
even']//h3//a")
    links = []
    for ii in lista:
        #time.sleep(3)
        tmp = ii.get_attribute("href")
        if tmp != None:
            links.append(tmp)
    for ii in links:
        print(ii)
        browser.get('https://psr.regione.molise.it/aperti')
        time.sleep(3)
        browser.get(ii+'/')
        time.sleep(2)
        tmp_list = browser.find_elements(By.PARTIAL_LINK_TEXT, 'Leggi
tutto')
        for jj in tmp_list:
            browser.get(jj.get_attribute('href'))
            b_tmp = scraper(browser)
            time.sleep(2)
            if b_tmp != None:
                bandi.append(b_tmp)
    return bandi

def psrMolise():
    browser = webdriver.Chrome()
    browser.get('https://psr.regione.molise.it/aperti')
```

```
    bandi = crawler(browser)
if __name__=='__main__':
    psrMolise()
```