Review

1. from selenium import webdriver
2. from selenium.webdriver.support.wait import WebDriverWait
3. from selenium.webdriver.support import expected\_conditions
4. from selenium.webdriver.common.action\_chains import ActionChains
5. import re
7. """
8. Vamos levantar um http server com o comando abaixo:
9. python -m http.server --bind 127.0.0.1 8083
10. referencia: http://www.geradorcpf.com/mascara-cpf-com-jquery.htm
11. """
13. # Instanciando o firefox driver.
14. driver = webdriver.Firefox()
15. # Abre a url.
16. driver.get("http://127.0.0.1:8083/html/cpf.html")
18. # Espera de forma implicita os elementos carregar na pagina.
19. driver.implicitly\_wait(30)

22. def find(driver, selector, delay=30):
23. # Espera Explicita > Melhor forma de fazer.
24. # An expectation for checking that an element is present on the DOM of a page and visible.
25. return WebDriverWait(driver, delay).until(
26. expected\_conditions.visibility\_of\_element\_located(("css selector", selector)))
28. elemento = find(driver, "input[id=cpf]")
29. print(elemento)
31. cpf = "52540430066"
32. # elemento.send\_keys(cpf)
33. ActionChains(driver).move\_to\_element(elemento).click().send\_keys(cpf).perform()
35. elemento = find(driver, "input[id=cpf]")
36. value = elemento.get\_attribute('value')
37. value = re.sub(r'(\D+)', '', value)
39. assert cpf in value
41. # fecha o browser
42. driver.quit()

Resolvemos o problema do CPF via ActionChains o X da questão foi o click() para deixar o elemento como ativo e assim o send\_keys do ActionChains funcionou bem, se você remover o click vai ver que o send\_keys não funciona como a função base do selenium.

\*\* Um curiosidade do Selenium que muita das vezes você manda o comando uma vez e não funciona e você manda ele duas vezes e funciona como se fosse mágica, na verdade é que o Selenium se perdeu na primeira vez, mas ele conseguiu manter o foco no elemento e quando você manda o segundo ele não se perde pois já tem o foco.