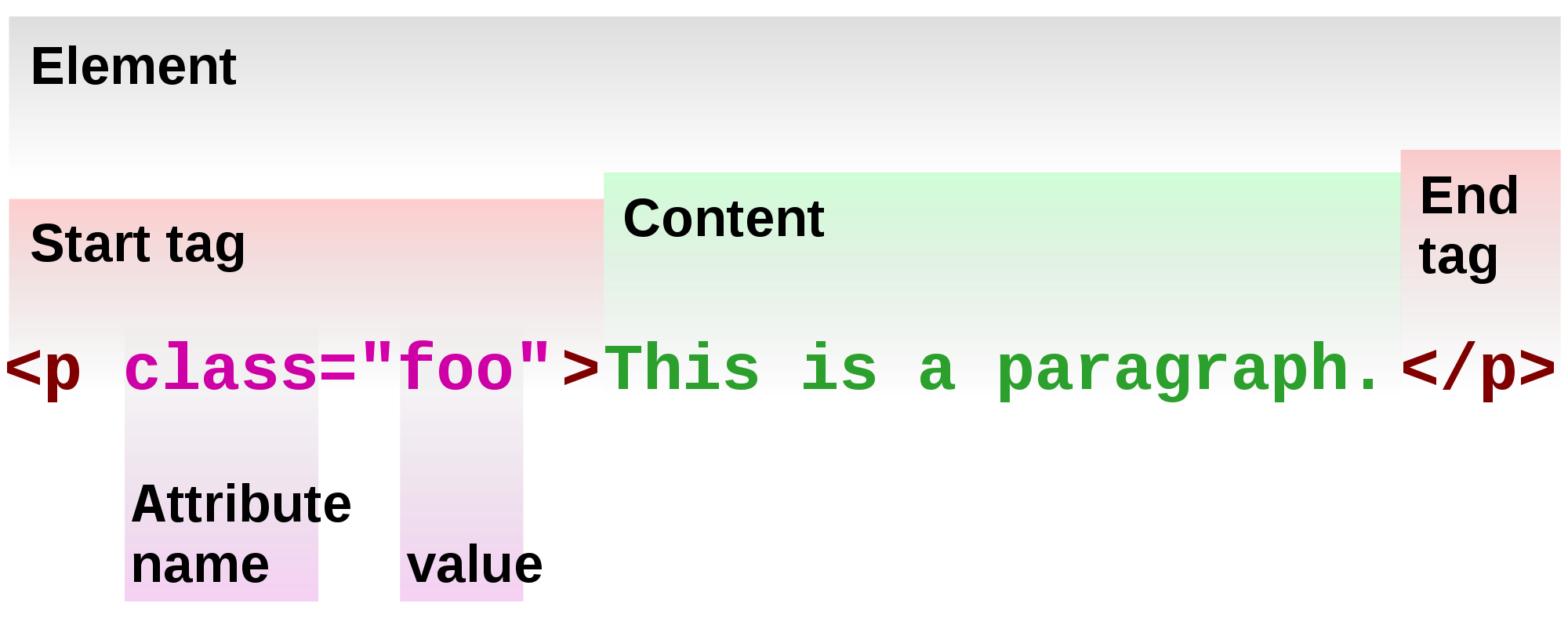
code review do selenium básico

1. from selenium import webdriver
2. from selenium.webdriver.common.by import By
3. from selenium.webdriver.support.wait import WebDriverWait
4. from selenium.webdriver.support import expected\_conditions
6. # Instanciando o firefox driver.
7. # Passamos o caminho do driver, no meu caso eh linux a barra / eh no windows eh o contrario.
8. driver = webdriver.Firefox(executable\_path="/media/dev/DATA/testes/drivers/geckodriver")
9. # Abre a url.
10. driver.get("http://www.w3schools.com/html/tryit.asp?filename=tryhtml\_basic\_link")
11. # Espera de forma implicita os elementos carregar na pagina.
12. driver.implicitly\_wait(10)
13. # Muda para o frame correto.
14. driver.switch\_to.frame("iframeResult")

17. def espera\_explicita(driver, selector, delay=30):
18. # Espera Explicita > Melhor forma de fazer.
19. # An expectation for checking that an element is present on the DOM of a page and visible.
20. return WebDriverWait(driver, delay).until(
21. expected\_conditions.visibility\_of\_element\_located((By.CSS\_SELECTOR, selector)))

24. espera\_explicita(driver, "a")
26. # Procura o elemento na tela via CSS Selector.
27. elemento = driver.find\_element(By.CSS\_SELECTOR, "a")
29. # Armazenar o elemento texto da tag a
30. variavel\_texto = elemento.text
32. # Armazenar o atributo href da tag a
33. variavel\_url = elemento.get\_attribute('href')
35. # imprimindo no console os valores texto e url.
36. print(variavel\_texto, variavel\_url)
38. # Fazendo os assert, se os valores sao diferentes do esperado o assert vai falhar ou
39. # seja nosso teste falhou.
41. assert "This is a link" == variavel\_texto
42. assert "http://www.w3schools.com/" == variavel\_url
44. # fecha o browser
45. driver.quit()

**Anatomia de um elemento HTML**

Um Elemento HTML é definido através de uma tag, exemplo um paragrafo "<p class="foo">Paragrafo</p> ", a tag deve ter um inicio e fim, como no exemplo abaixo , as tags possui ou não atributos, atributos prove informações adicionais aos elementos.

**Em python para pegar o elemento HTML, no nosso caso o elemento parágrafo:**

1. # Procura o elemento na tela pela tag <p> via CSS Selector.
2. elemento = driver.find\_element(By.CSS\_SELECTOR, "p")

**Em python para ter acesso as atributos:**

1. # Armazenar o atributo class da tag p
2. variavel\_class = elemento.get\_attribute('class')
3. print(variavel\_class)
4. # Resultado: foo

**Em python para ter acesso ao conteúdo da tag:**

1. # Armazenar o elemento texto da tag p
2. variavel\_texto = elemento.text
3. print(variavel\_texto)
4. # Resultado: This is a paragraph.

Python é uma linguagem que foi criada inicialmente com o paradigma funcional, com o tempo foi colocado suporte para código orientado a objeto, que é outro paradigma. Na época os autores decidiram que deveria ficar explicito a passagem do objeto a que pertence a função que está sendo chamada quando a função pertencer a uma classe porque o foco da linguagem eram as funções. Assim convencionaram que seria o primeiro parâmetro e o nome seria self. Em outras linguagens é semelhante ao this.

Se você seguir o [Zen do Python](https://www.python.org/dev/peps/pep-0020/#id3) você verá que em Python: *Explícito é melhor que implícito*. E isso é levado muito a sério em toda a linguagem!

1. # Criamos uma classe com a palavra reservada "class", em seguida o nome da classe, e ":" no final

4. class NomeDaClasse:
5. propriedade = 0
7. def metodo(self):
8. print(self.propriedade) # se tiver só propriedade aqui seria uma variavel local

11. objeto = NomeDaClasse() # Realiza a Instancia a classe, para ter acesso aos métodos.
12. objeto.propriedade = 10 # Mudando o valor da variavel da classe.
13. objeto.metodo() # Realizando acesso a funcao/metodo da classe.
15. # resultado: 10

https://www.python.org/dev/peps/pep-0020/#id3