```
def test_mfa_page_with_email_method_and_imap(monkeypatch, mocker):
    # Prepare dummy elements
    class DummyInput:
        def clear(self): pass
        def send_keys(self, x): self.sent = x
        def click(self): self.clicked = True
    driver = mocker.Mock()
    token_input = DummyInput()
    token_button = DummyInput()
    # Patch search_mfa_method to simulate email MFA page
    monkeypatch.setattr('mintapi.signIn.search_mfa_method', lambda d: (token_input, token_button, 'EMAIL'))
    # Patch get_email_code to return a fixed code
    monkeypatch.setattr('mintapi.signIn.get_email_code', lambda a,b,c,d: '654321')
    called = {}
    def mfa_input_callback(prompt):
        called['prompted'] = True
        return '999999'
    mfa_page(driver, mfa_method=None, mfa_token=None, mfa_input_callback=mfa_input_callback,
             imap_account='ia', imap_password='ip', imap_server='is', imap_folder='INBOX')
    assert token_input.sent == '654321'
    assert hasattr(token_button, 'clicked')
```

---Extra Details---
Filename: signIn.py
Description: Tests the mfa_page function with EMAIL MFA using IMAP, ensuring that the get_email_code function provides the ex
Score: 98%
Alignment: 95%
Validation Notes: Directly exercises the IMAP/email MFA branch, with dummy input and code patch.