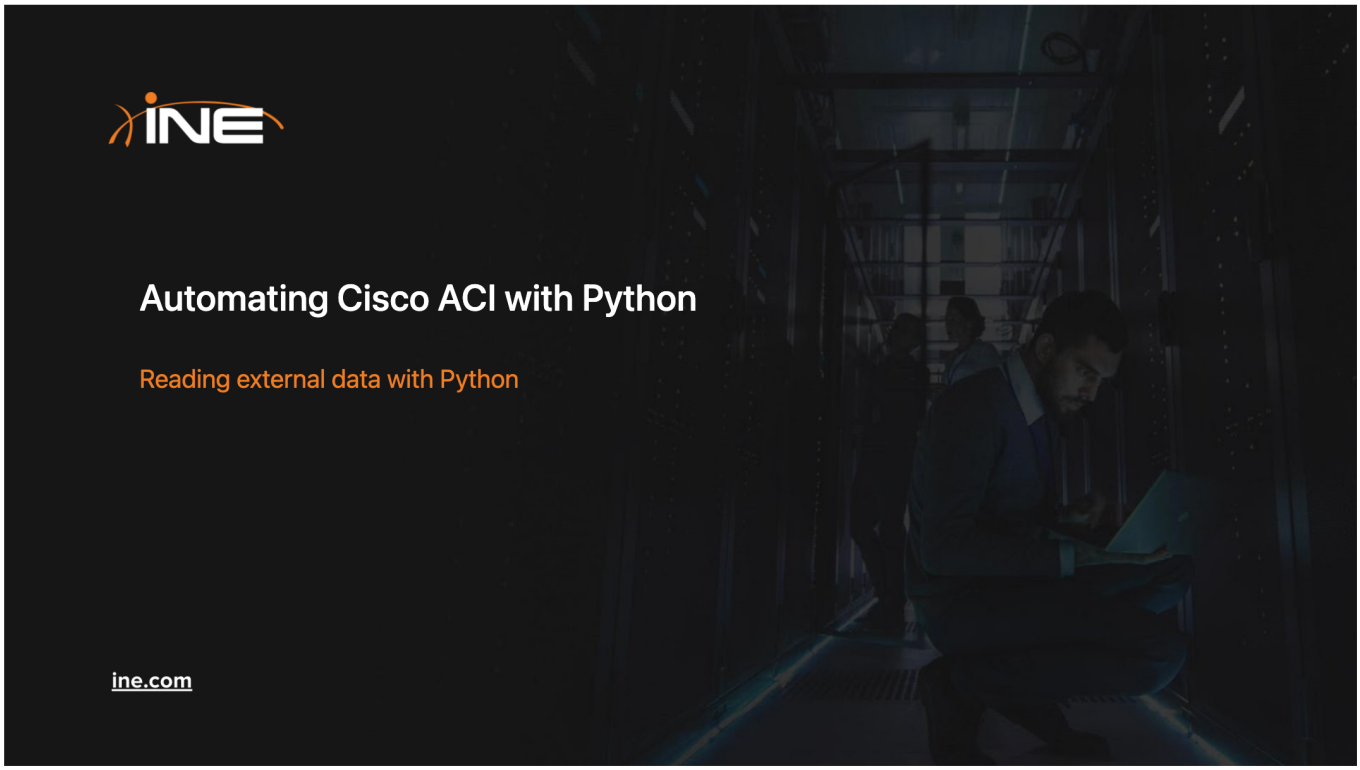


Working with files is crucial to write any meaningful software. The good news is that it's super simple to do file management in Python. For example, this is how you can read a file:

```
file = open("file.csv")

file.read()
```

By default, the function will open the "file object" in the read mode and treat the data in the file as texts.

Let's see an example.

In [32]:
```
file = open("data/aci_data_2.csv")
```

In [33]:
```
print(file)
```

```
<_io.TextIOWrapper name='data/aci_data_2.csv' mode='r' encoding='UTF-8'>
```

In [34]:
```
print(file.read())
```

```
tenant,bridge_domain_name,address,scope
1-tenant,1-bridge_domain,10.100.1.254/24,shared
1-tenant,2-bridge_domain,10.100.2.254/24,public
1-tenant,3-bridge_domain,10.100.3.254/24,private
2-tenant,1-bridge_domain,10.100.1.254/24,public
```

Important Concepts

There are two important concepts to understand when dealing with files:

- Open modes
- File cursor (or pointer)

Open modes

The "Open Mode" is something you decide when you're opening a file, and basically states if you want to read or to write to a file (or both).

There are different modes you can open a file using the `mode` parameter:

- `r` - reading mode, default mode. Open the file for reading.
- `w` - writing mode. Open the file for writing (**WARNING,** erases the contents of the file).
- `a` - append mode. Open the file for writing, but place the *cursor* at the end to append new data.
- `x` - exclusive creation. Open the file for writing and raise an exception if the file already exists.

> Using the `+` plus mode, will let you read and write at the same time without having to use `r` and `w`.

	r	r+	w	w+	a	a+	x	x+
Read	✓	✓	✘	✓	✘	✓	✘	✓
Write	✘	✓	✓	✓	✓	✓	✓	✓

Creates file	✘	✘	✓	✓	✓	✓	✓	✓
Erases file	✘	✘	✓	✓	✘	✘	✘	✘

In [35]:
```
file = open("data/aci_data_2.csv", mode="r")

file
```

Out[35]:
```
<_io.TextIOWrapper name='data/aci_data_2.csv' mode='r' encoding='UTF-8'>
```

In [36]:
```
file.write("something")
```

```
---------------------------------------------------------------------------
UnsupportedOperation                      Traceback (most recent call last)
<ipython-input-36-e17a54641b4d> in <module>
----> 1 file.write("something")

UnsupportedOperation: not writable
```

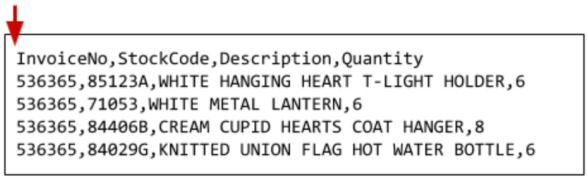
As you can see, an `UnsupportedOperation` exception was raised. We tried to write in a file opened for reading. It was an invalid mode.

You can already guess that the `open` function we introduced before takes a second parameter, which is actually the "open mode".

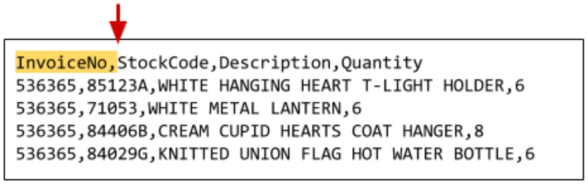
Cursors (or pointers)

We can think a file as a big string of characters (also bytes), suppose we're working with a CSV file with the following content:

We always need to have a notion of "position". That's the idea of the cursor or pointer. You'll open a file and Python will place automatically the cursor at some given position. For example, we open it with mode `'r'`, the pointer is placed at the beginning:



Now we can perform different operations (reading, writing, etc) which will be performed based on the pointer. For example, we can decide to read 10 characters: Python will take the current position of the pointer, read the 10 characters, and place the pointer after the last one read.

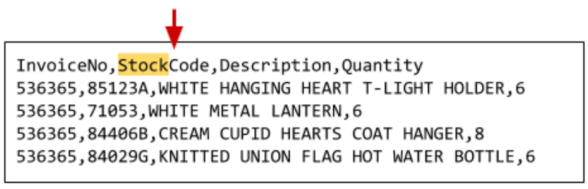


In [38]:
```
file = open("data/aci_data_2.csv", mode="r")

file.read(10)
```

Out[38]:
```
'tenant,bri'
```

If we then decide to read 5 more characters, Python will start from that previous position of the pointer and after reading, it'll keep the pointer at the last character again:



In [39]:
```
file.read(5)
```

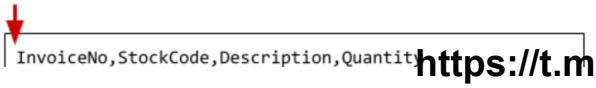
Out[39]:
```
'dge_d'
```

There are two methods to manage the pointer: `tell` will tell you the position of the pointer, and `seek` will let you move that pointer to whatever position you want. For example, after reading 10 characters first, and 5 characters in our second operation, we should expect this file's pointer position to be `15`:

In [40]:
```
file.tell()
```

Out[40]:
```
15
```

I could now "reset" the pointer and move it back at the beginning (similarly as how it started when we just opened the file)

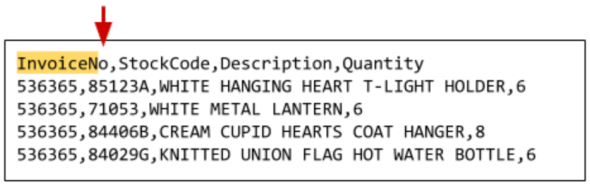


```
536365,85123A,WHITE HANGING HEART T-LIGHT HOLDER,6
536365,71053,WHITE METAL LANTERN,6
536365,84406B,CREAM CUPID HEARTS COAT HANGER,8
536365,84029G,KNITTED UNION FLAG HOT WATER BOTTLE,6
```

In [41]:
```
file.seek(0)
```

Out[41]:
```
0
```

And then try to read 8 characters:



In [42]:
```
file.read(8)
```

Out[42]:
```
'tenant,b'
```

Reading line by line

To avoid this, you can read by lines dividing the file by the newline char (`'\n'`).

In [44]:
```
file = open("data/aci_data_2.csv", mode="r")

line = file.readline()
print(line)
```

```
tenant,bridge_domain_name,address,scope
```

In [46]:
```
line.split(",")
```

Out[46]:
```
['tenant', 'bridge_domain_name', 'address', 'scope\n']
```

File objects and the iterator pattern

When you open a file with the function `open` , the result is a "file object". It's a full featured Python object that has a few methods and attributes (`read` , `readline` , etc). But as many other Python objects, it also holds the concept of "iteration" (as for example, lists or dicts). That means that you can just "iterate" over that object with a regular for loop. Each "pass" of the iterator will give you a new line (similar to the `readline()` method).

In [49]:
```
file = open("data/aci_data_2.csv", mode="r")

for line_num, line in enumerate(file):
    if (line_num < 2):
        print(line_num, line)
```

```
0 tenant,bridge_domain_name,address,scope

1 1-tenant,1-bridge_domain,10.100.1.254/24,shared
```

Closing the file

The operating system needs to allocate resources to keep up with the files you open. Remember that the OS needs to "monitor" all our files-related activities; so every time you open a file, the OS needs to allocate resources for that "monitoring" task.

Closing a file is simple, just use the `close()` method of the file object returned by `open()` :

In [50]:
```
file.closed
```

Out[50]:
```
False
```

In [51]:
```
file.close()
```

In [52]:
```
file.closed
```

Out[52]:
```
True
```

Any successive operation that we try to perform with a closed file will raise an exception:

In [53]:
```
file.read(5)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-53-99e24f97803f> in <module>
```

```
----> 1 file.read(5)

ValueError: I/O operation on closed file.
```

To avoid forgetting to close the file everytime you use it, it is recommended to use **the *with* context manager**:

In [54]:
```
with open("data/aci_data_2.csv", "r") as file:
    print(file.read())
```

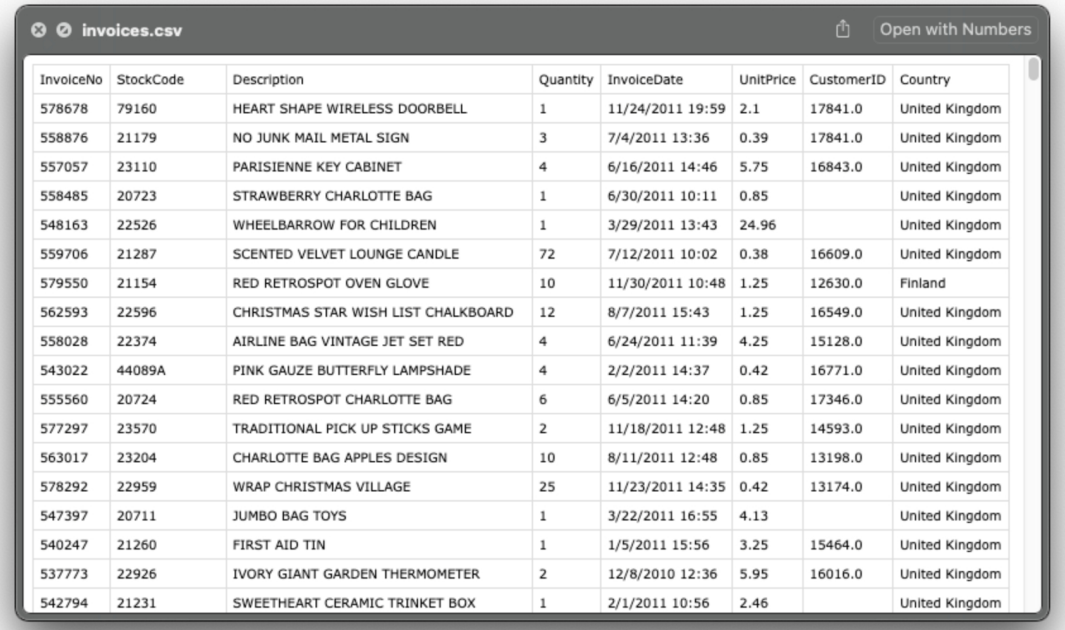
```
tenant,bridge_domain_name,address,scope
1-tenant,1-bridge_domain,10.100.1.254/24,shared
1-tenant,2-bridge_domain,10.100.2.254/24,public
1-tenant,3-bridge_domain,10.100.3.254/24,private
2-tenant,1-bridge_domain,10.100.1.254/24,public
```

In [55]:
```
file.closed
```

Out[55]:
```
True
```

Reading CSV files

If you're doing any meaningful work with CSV files, you'll probably be using pandas or csvkit. But nevertheless, it's a good use case for traditional file management. Python has the csv module which is already builtin. It's just about importing it and using it.



InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
578678	79160	HEART SHAPE WIRELESS DOORBELL	1	11/24/2011 19:59	2.1	17841.0	United Kingdom
558876	21179	NO JUNK MAIL METAL SIGN	3	7/4/2011 13:36	0.39	17841.0	United Kingdom
557057	23110	PARISIENNE KEY CABINET	4	6/16/2011 14:46	5.75	16843.0	United Kingdom
558485	20723	STRAWBERRY CHARLOTTE BAG	1	6/30/2011 10:11	0.85		United Kingdom
548163	22526	WHEELBARROW FOR CHILDREN	1	3/29/2011 13:43	24.96		United Kingdom
559706	21287	SCENTED VELVET LOUNGE CANDLE	72	7/12/2011 10:02	0.38	16609.0	United Kingdom
579550	21154	RED RETROSPOT OVEN GLOVE	10	11/30/2011 10:48	1.25	12630.0	Finland
562593	22596	CHRISTMAS STAR WISH LIST CHALKBOARD	12	8/7/2011 15:43	1.25	16549.0	United Kingdom
558028	22374	AIRLINE BAG VINTAGE JET SET RED	4	6/24/2011 11:39	4.25	15128.0	United Kingdom
543022	44089A	PINK GAUZE BUTTERFLY LAMPSHADE	4	2/2/2011 14:37	0.42	16771.0	United Kingdom
555560	20724	RED RETROSPOT CHARLOTTE BAG	6	6/5/2011 14:20	0.85	17346.0	United Kingdom
577297	23570	TRADITIONAL PICK UP STICKS GAME	2	11/18/2011 12:48	1.25	14593.0	United Kingdom
563017	23204	CHARLOTTE BAG APPLES DESIGN	10	8/11/2011 12:48	0.85	13198.0	United Kingdom
578292	22959	WRAP CHRISTMAS VILLAGE	25	11/23/2011 14:35	0.42	13174.0	United Kingdom
547397	20711	JUMBO BAG TOYS	1	3/22/2011 16:55	4.13		United Kingdom
540247	21260	FIRST AID TIN	1	1/5/2011 15:56	3.25	15464.0	United Kingdom
537773	22926	IVORY GIANT GARDEN THERMOMETER	2	12/8/2010 12:36	5.95	16016.0	United Kingdom
542794	21231	SWEETHEART CERAMIC TRINKET BOX	1	2/1/2011 10:56	2.46		United Kingdom

A CSV file is just like a table

Let's start processing it...

CSV reading

We now import the `csv` module and initialize a csv `reader`:

In [15]:
```
import csv
```

In [16]:
```
!cat data/aci_data_2.csv
```

```
tenant,bridge_domain_name,address,scope
1-tenant,1-bridge_domain,10.100.1.254/24,shared
1-tenant,2-bridge_domain,10.100.2.254/24,public
1-tenant,3-bridge_domain,10.100.3.254/24,private
2-tenant,1-bridge_domain,10.100.1.254/24,public
```

In [21]:
```
with open("data/aci_data_2.csv", 'r') as file:
    reader = csv.reader(file)

    print(reader)
```

<_csv.reader object at 0x10a0ec80>

In [23]:
```
with open("data/aci_data_2.csv", 'r') as file:
    reader = csv.reader(file)

    for i, row in enumerate(reader):
        print(i, row)
```

```
0 ['\ufefftenant', 'bridge_domain_name', 'address', 'scope']
1 ['1-tenant', '1-bridge_domain', '10.100.1.254/24', 'shared']
2 ['1-tenant', '2-bridge_domain', '10.100.2.254/24', 'public']
3 ['1-tenant', '3-bridge_domain', '10.100.3.254/24', 'private']
4 ['2-tenant', '1-bridge_domain', '10.100.1.254/24', 'public']
```

We want to transform the CSV data to a useful Python data structure.

In this case using a `dict` with be the best option:

```
{
        "1-tenant": [
            {
                "bd": "1-bridge_domain",
                "address": "10.100.1.254/24",
                "scope": "shared"
            },
            {
                "bd": "1-bridge_domain",
                ...
            }
        ],
        "2-tenant": [
            ...
        ]
    }
```

> **Warning**: we don't want to include the CSV header as data!

In [26]:
```
with open("data/aci_data_2.csv", 'r') as file:
    reader = csv.reader(file)

    data = {}
    for i, row in enumerate(reader):
        if (i != 0):
            tenant = row[0]
            bd_dict = {
                "bd": row[1],
                "address": row[2],
                "scope": row[3]
            }

            if (tenant in data.keys()):
                data[tenant].append(bd_dict)
            else:
                data[tenant] = [bd_dict]

print(data)
```

```
{'1-tenant': [{'bd': '1-bridge_domain', 'address': '10.100.1.254/24', 'scope': 'shared'}, {'bd': '2-bridge_domain', 'address': '10.100.2.254/24', 'scope': 'public'}, {'bd': '3-bridge_domain', 'address': '10.100.3.254/24', 'scope': 'private'}], '2-tenant': [{'bd': '1-bridge_domain', 'address': '10.100.1.254/24', 'scope': 'public'}]}
```

In [27]:
```
tenants = data.keys()

tenants
```

Out[27]:
```
dict_keys(['1-tenant', '2-tenant'])
```

In [31]:
```
for t in tenants:
    print(t)

    for bd in data[t]:
        print(f"--- {bd['bd']} {bd['address']}")
```

```
1-tenant
--- 1-bridge_domain 10.100.1.254/24
--- 2-bridge_domain 10.100.2.254/24
--- 3-bridge_domain 10.100.3.254/24
2-tenant
--- 1-bridge_domain 10.100.1.254/24
```

In [ ]:
```
# Create and push Tenant and BDs
tenants = data.keys()
for t in tenants:
    ten = aci.Tenant(t)

    for bd_data in data[t]:
        bd = aci.BridgeDomain(bd_data["bd"], ten)
        subnet = aci.Subnet('', bd)
        subnet.addr = bd_data["address"]
        subnet.set_scope(bd_data["scope"])

    resp = ten.push_to_apic(session)
    if resp.ok:
        print(f'OK {t}')
    else:
        print('%% Error: Could not push configuration to APIC')
        print(resp.text)
```

Reading YAML files

In [7]:
```
import yaml
```

In [8]:
```
!cat data/aci_data_1.yml
```

```
---
tenants:
- name: "1-tenant"
  desc: "Automated tenant using Python"
  vrf: "1-vrf"
  bridge_domains:
    - bd: "1-bd"
      gateway: "10.16.110.1"
      mask: "24"
      scope: "shared"
    - bd: "2-bd"
      gateway: "10.17.110.1"
      mask: "24"
      scope: "public"
    - bd: "3-bd"
      gateway: "10.18.110.1"
      mask: "24"
      scope: "private"
- name: "2-tenant"
  desc: "Automated tenant using Python"
  vrf: "1-vrf"
  bridge_domains:
  - bd: "1-bd"
    gateway: "10.16.110.1"
    mask: "24"
    scope: "public"
```

In [12]:
```
# Read YAML configuration
yml_file = open("data/aci_data_1.yml").read()

yml_file
```

Out[12]:
```
'---\ntenants:\n- name: "1-tenant"\n  desc: "Automated tenant using Python"\n  vrf: "1-vrf"\n  bridge_domains:\n    - bd: "1-bd"\n      ga
teway: "10.16.110.1"\n      mask: "24"\n      scope: "shared"\n    - bd: "2-bd"\n      gateway: "10.17.110.1"\n      mask: "24"\n      sco
pe: "public"\n    - bd: "3-bd"\n      gateway: "10.18.110.1"\n      mask: "24"\n      scope: "private"\n- name: "2-tenant"\n  desc: "Autom
ated tenant using Python"\n  vrf: "1-vrf"\n  bridge_domains:\n  - bd: "1-bd"\n    gateway: "10.16.110.1"\n    mask: "24"\n    scope: "publ
ic"'
```

In [13]:
```
yml_dict = yaml.load(yml_file, yaml.SafeLoader)

yml_dict
```

Out[13]:
```
{'tenants': [{'name': '1-tenant',
   'desc': 'Automated tenant using Python',
   'vrf': '1-vrf',
   'bridge_domains': [{'bd': '1-bd',
     'gateway': '10.16.110.1',
     'mask': '24',
     'scope': 'shared'},
    {'bd': '2-bd', 'gateway': '10.17.110.1', 'mask': '24', 'scope': 'public'},
    {'bd': '3-bd',
     'gateway': '10.18.110.1',
     'mask': '24',
     'scope': 'private'}]},
  {'name': '2-tenant',
   'desc': 'Automated tenant using Python',
   'vrf': '1-vrf',
   'bridge_domains': [{'bd': '1-bd',
     'gateway': '10.16.110.1',
     'mask': '24',
     'scope': 'public'}]}]}
```

In [16]:
```
tenants = yml_dict["tenants"]

for t in tenants:
    print(f"{t['name']} - {t['desc']}")
```

```
1-tenant - Automated tenant using Python
2-tenant - Automated tenant using Python
```

In [21]:
```
tenants = yml_dict["tenants"]

for t in tenants:
    print(f"{t['name']} - {t['desc']}")

    for bd in t['bridge_domains']:
        print(f"--> {bd['bd']} {bd['gateway']}")
```

```
1-tenant - Automated tenant using Python
--> 1-bd 10.16.110.1
--> 2-bd 10.17.110.1
--> 3-bd 10.18.110.1
2-tenant - Automated tenant using Python
--> 1-bd 10.16.110.1
```

JSON files

In [56]:
```
json_string = '''
  {
    "id": "5fce1c837cb3e52a804d1b50",
    "isActive": false,
    "balance": "$2,730.70",
    "age": 36,
    "name": {
      "first": "Meadows",
      "last": "Barber"
    },
    "company": "TECHTRIX",
    "email": "meadows.barber@techtrix.tv",
    "phone": "+1 (980) 509-3159",
    "address": "841 Boerum Place, Bladensburg, Idaho, 5282",
    "tags": [
      {"id": 1, "name": "aliquip"},
      {"id": 2, "name": "reprehenderit"},
      {"id": 3, "name": "eu"},
      {"id": 4, "name": "eiusmod"},
      {"id": 5, "name": "officia"}
    ]
  }
'''
```

In [57]:
```
import json
```

In [58]:
```
json_file = json.loads(json_string)
```

In [59]:
```
json_file
```

Out[59]:
```
{'id': '5fce1c837cb3e52a804d1b50',
 'isActive': False,
 'balance': '$2,730.70',
 'age': 36,
 'name': {'first': 'Meadows', 'last': 'Barber'},
 'company': 'TECHTRIX',
 'email': 'meadows.barber@techtrix.tv',
 'phone': '+1 (980) 509-3159',
 'address': '841 Boerum Place, Bladensburg, Idaho, 5282',
 'tags': [{'id': 1, 'name': 'aliquip'},
  {'id': 2, 'name': 'reprehenderit'},
  {'id': 3, 'name': 'eu'},
  {'id': 4, 'name': 'eiusmod'},
  {'id': 5, 'name': 'officia'}]}
```

In [60]:
```
type(json_file)
```

Out[60]:
```
dict
```

In [61]:
```
json_file["tags"]
```

Out[61]:
```
[{'id': 1, 'name': 'aliquip'},
 {'id': 2, 'name': 'reprehenderit'},
 {'id': 3, 'name': 'eu'},
 {'id': 4, 'name': 'eiusmod'},
 {'id': 5, 'name': 'officia'}]
```

In [62]:
```
json_file["tags"][3]
```

Out[62]:
```
{'id': 4, 'name': 'eiusmod'}
```

In [63]:
```
json_file["tags"][3]["name"]
```

Out[63]:
```
'eiusmod'
```

In [64]:
```
for tag in json_file["tags"]:
    print(tag["name"])
```

```
aliquip
reprehenderit
eu
eiusmod
officia
```

Reading user input

Python user input from the keyboard can be read using the `input()` built-in function.

In [9]:
```
value = input("Please enter a string:\n")
```

```
print(f'You entered {value}')
```

```
You entered
```

In [10]:
```
import data.read_input as ri
```

In [11]:
```
ri.__dir__()
```

Out[11]:
```
['__name__',
 '__doc__',
 '__package__',
 '__loader__',
 '__spec__',
 '__file__',
 '__cached__',
 '__builtins__',
 'read_user_input']
```

In [12]:
```
ri.read_user_input()
```

```
You entered asfasf
```