INTRODUCTION TO DATA SCIENCE

TA class I – Jupyter Notebook

TA : Lee Chi-Hsuan

Before starting…

Anaconda

- Data science toolkit
- Tons of packages

ANACONDA®

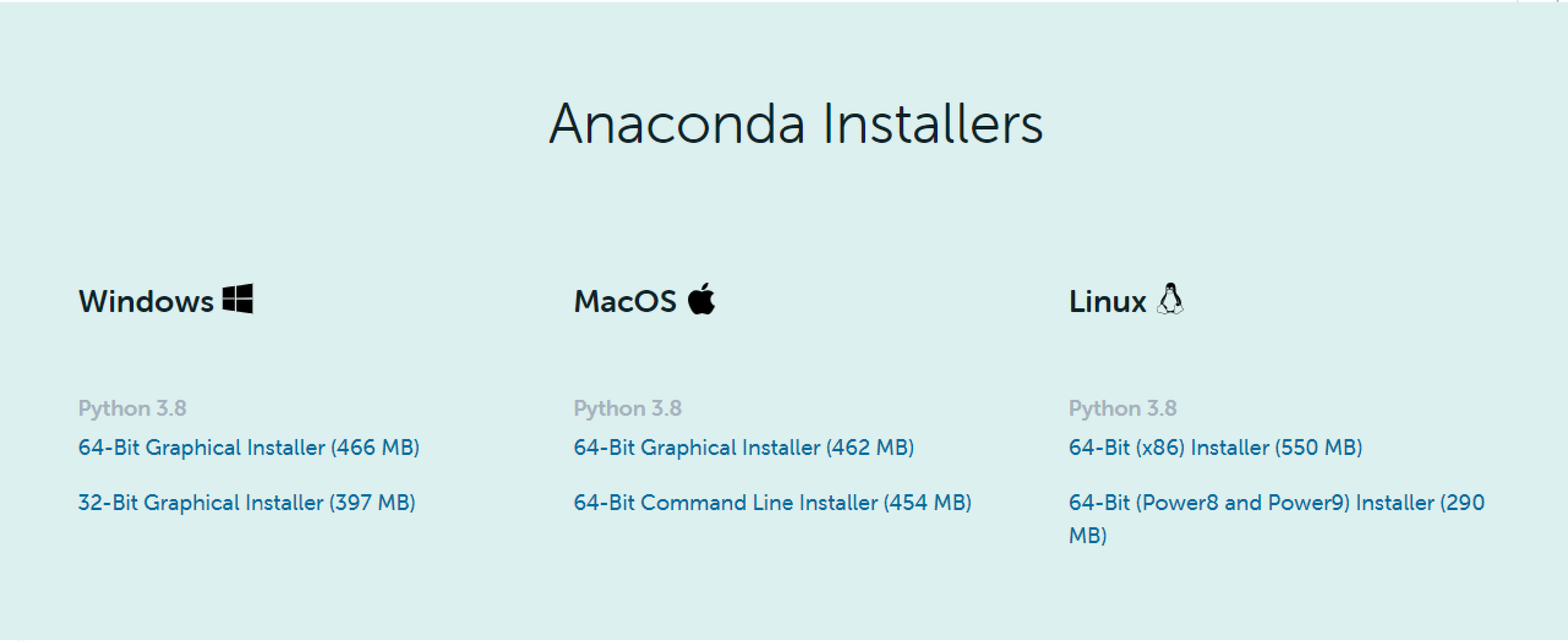
jupyter SciPy SPYDER NumPy

DASK bokeh scikit learn

TensorFlow Numba Panel pandas

HoloViews PYTORCH matplotlib CONDA®

- Download and Install…

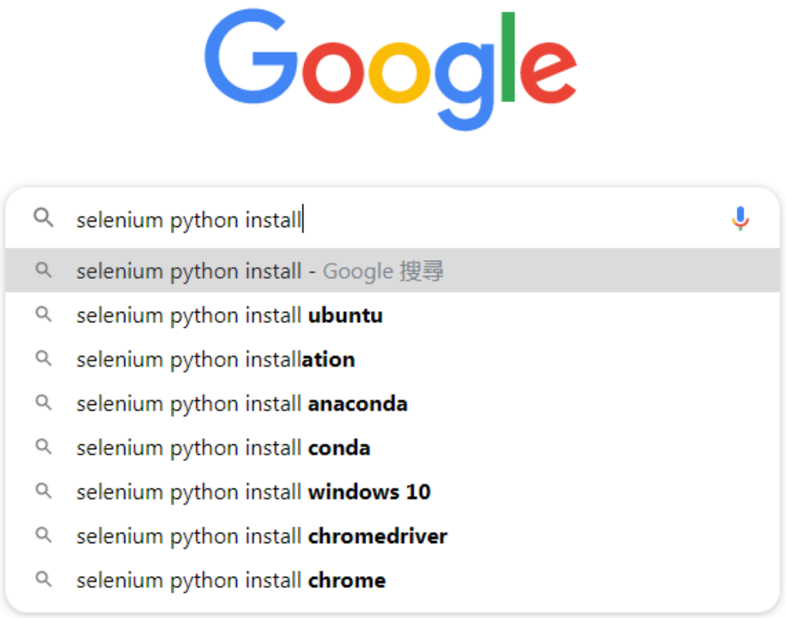


Anaconda Installers

Windows	MacOS	Linux
Python 3.8	Python 3.8	Python 3.8
64-Bit Graphical Installer (466 MB)	64-Bit Graphical Installer (462 MB)	64-Bit (x86) Installer (550 MB)
32-Bit Graphical Installer (397 MB)	64-Bit Command Line Installer (454 MB)	64-Bit (Power8 and Power9) Installer (290 MB)

TRIVIAL

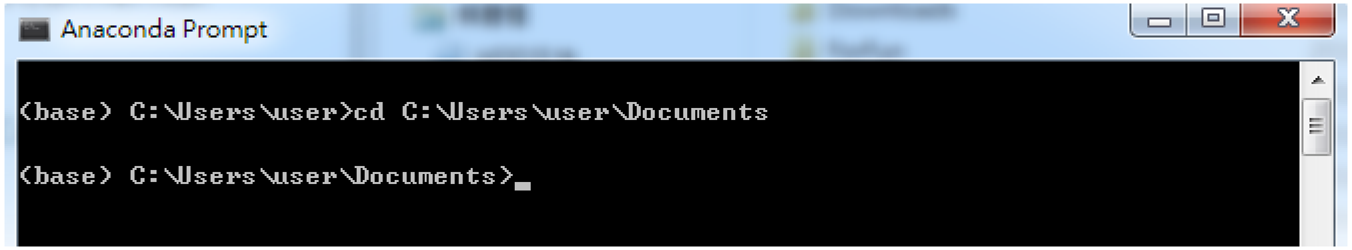
- Packages…


selenium python install

selenium python install - Google 搜尋
selenium python install **ubuntu**
selenium python install**ation**
selenium python install **anaconda**
selenium python install **conda**
selenium python install **windows 10**
selenium python install **chromedriver**
selenium python install **chrome**

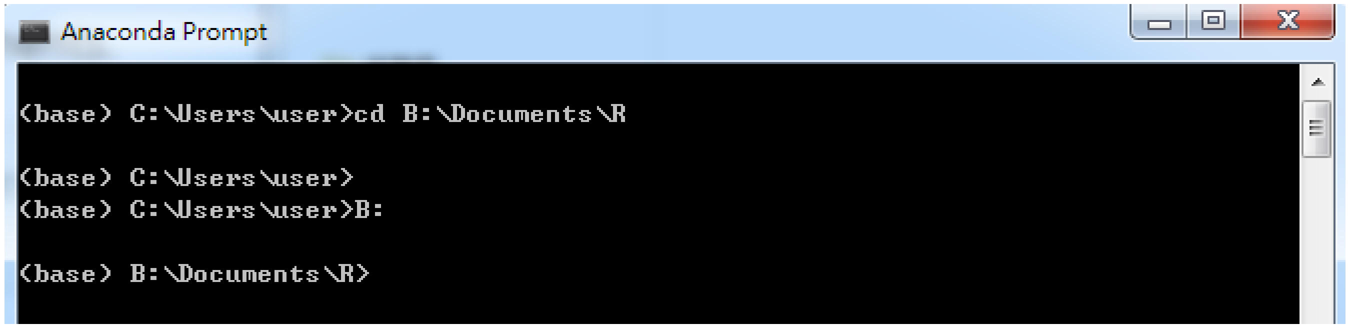
· Changing directory

Same parent



```
(base) C:\Users\user>cd C:\Users\user\Documents

(base) C:\Users\user\Documents>_
```

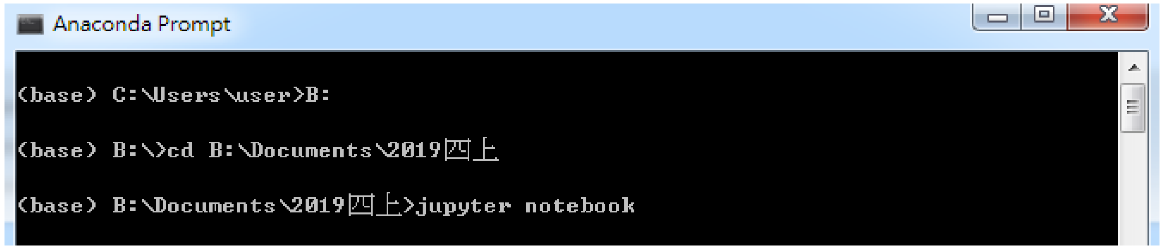
Different parent



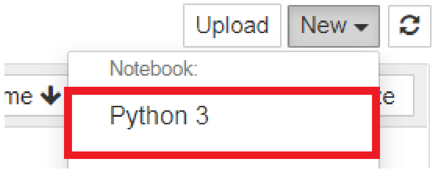
```
(base) C:\Users\user>cd B:\Documents\R

(base) C:\Users\user>
(base) C:\Users\user>B:

(base) B:\Documents\R>
```

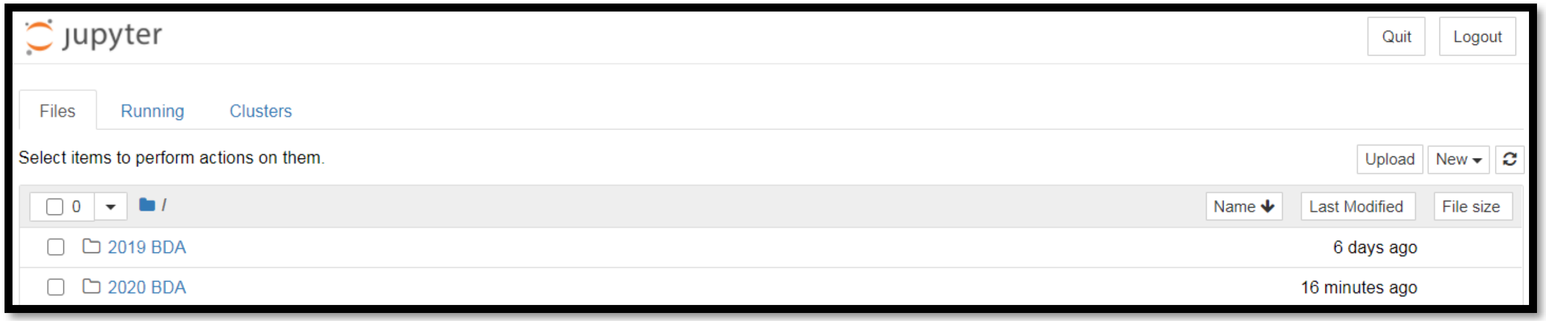
Jupyter Notebook

- Starting the Notebook



```
(base) C:\Users\user>B:

(base) B:\>cd B:\Documents\2019四上

(base) B:\Documents\2019四上>jupyter notebook
```





Select items to perform actions on them.

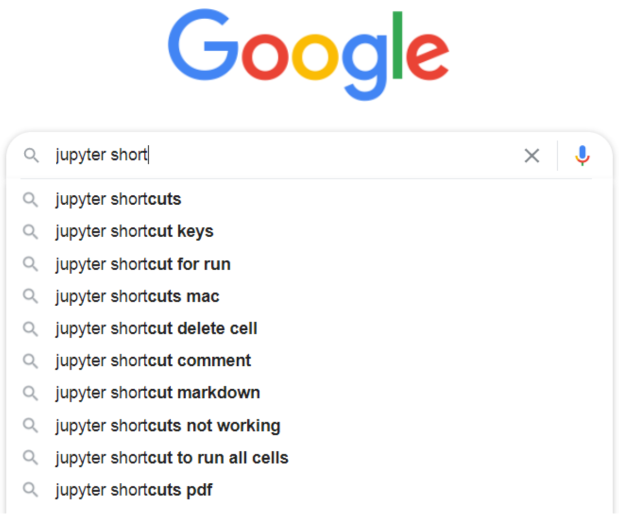
Name	Last Modified
2019 BDA	6 days ago
2020 BDA	16 minutes ago

TRIVIAL

(shift + Enter or Ctrl + Enter)

- Run your code block-wise
- Tab for auto-fill
- Ctrl + / for comment out multiple lines
- Shift + Tab for tooltips

And more...


jupyter short

jupyter shortcuts
jupyter shortcut keys
jupyter shortcut for run
jupyter shortcuts mac
jupyter shortcut delete cell
jupyter shortcut comment
jupyter shortcut markdown
jupyter shortcuts not working
jupyter shortcut to run all cells
jupyter shortcuts pdf

Python Quick Review

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one -- and preferably only one

-- obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than *right* now.

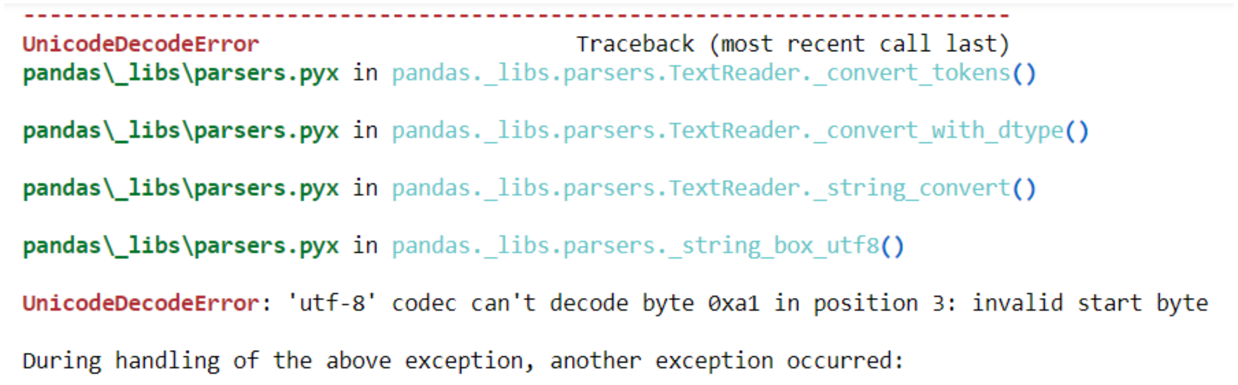
If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

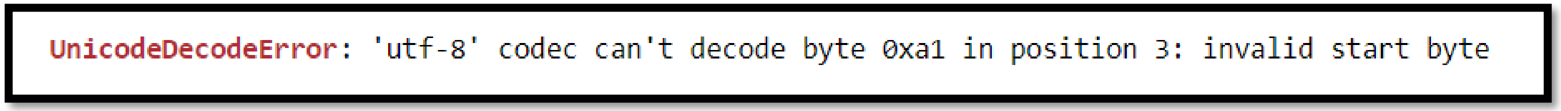
Namespaces are one honking great idea -- let's do more of those!

—The Zen of Python, by Tim Peters

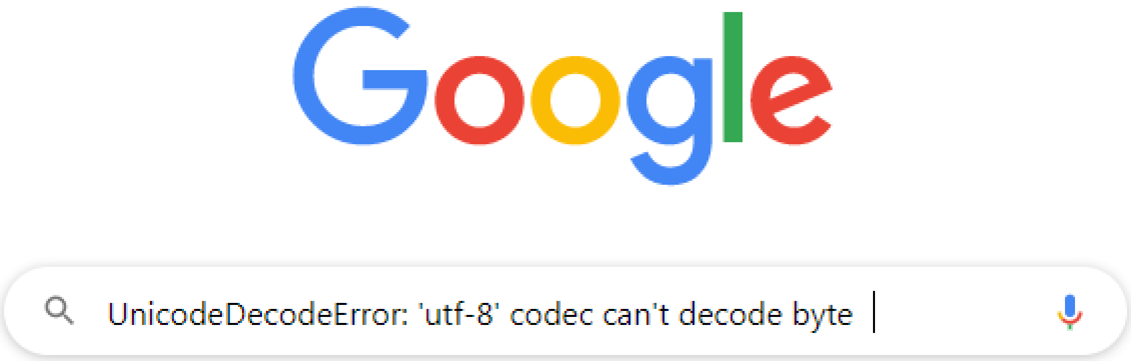
Wut... Error again ?!!



```
---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent call last)
pandas\_libs\parsers.pyx in pandas._libs.parsers.TextReader._convert_tokens()

pandas\_libs\parsers.pyx in pandas._libs.parsers.TextReader._convert_with_dtype()

pandas\_libs\parsers.pyx in pandas._libs.parsers.TextReader._string_convert()

pandas\_libs\parsers.pyx in pandas._libs.parsers._string_box_utf8()

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xa1 in position 3: invalid start byte

During handling of the above exception, another exception occurred:
```



```
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xa1 in position 3: invalid start byte
```



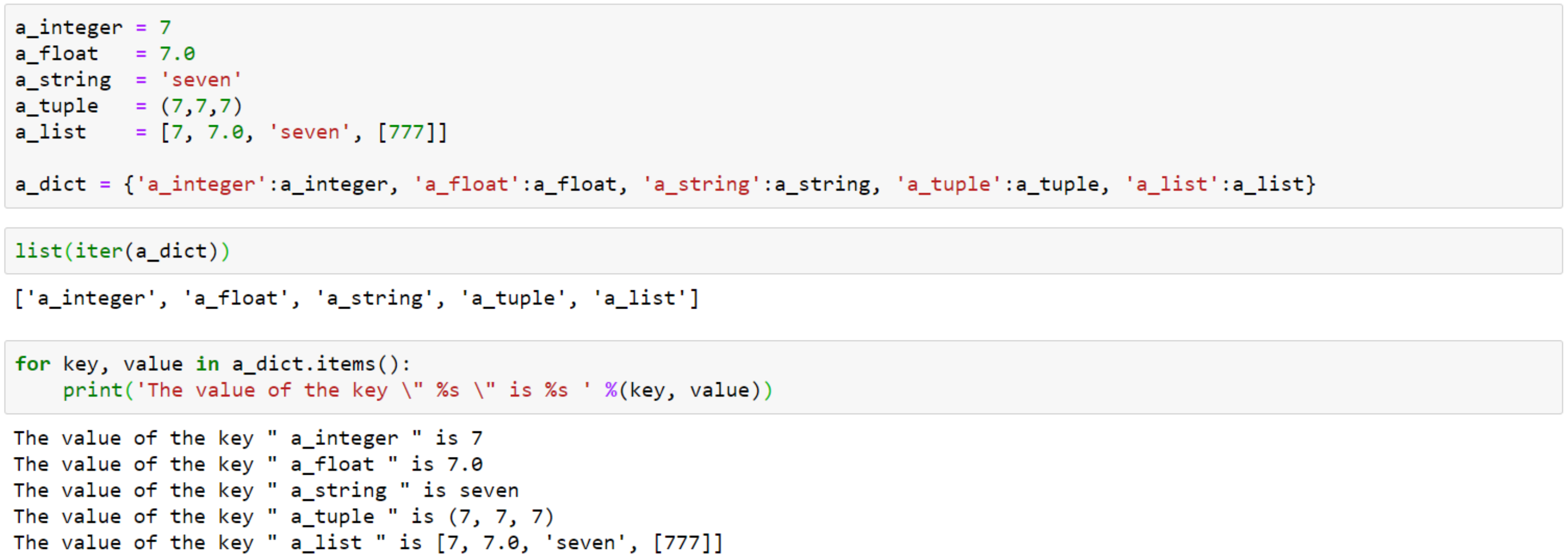
UnicodeDecodeError: 'utf-8' codec can't decode byte

You definitely are not the first person

to encounter this kind of problem.

Exception Type	Description
`IOError`	Raised when an I/O operation fails, such as when an attempt is make to open a nonexistent file in read mode.
`IndexError`	Raised when a sequence is indexed with a number of a nonexistent element.
`KeyError`	Raised when a dictionary key is not found.
`NameError`	Raised when a name (of a variable or function, for example) is not found.
`SyntaxError`	Raised when a syntax error is encountered.
`TypeError`	Raised when a built-in operation or function is applied to an object of inappropriate type.
`ValueError`	Raised when a built-in operation or function receives an argument that has the right type but an inappropriate value.
`ZeroDivisionError`	Raised when the second argument of a division or modulo operation is zero.

Different types of variables



```
a_integer = 7
a_float   = 7.0
a_string  = 'seven'
a_tuple   = (7,7,7)
a_list    = [7, 7.0, 'seven', [777]]

a_dict = {'a_integer':a_integer, 'a_float':a_float, 'a_string':a_string, 'a_tuple':a_tuple, 'a_list':a_list}
```

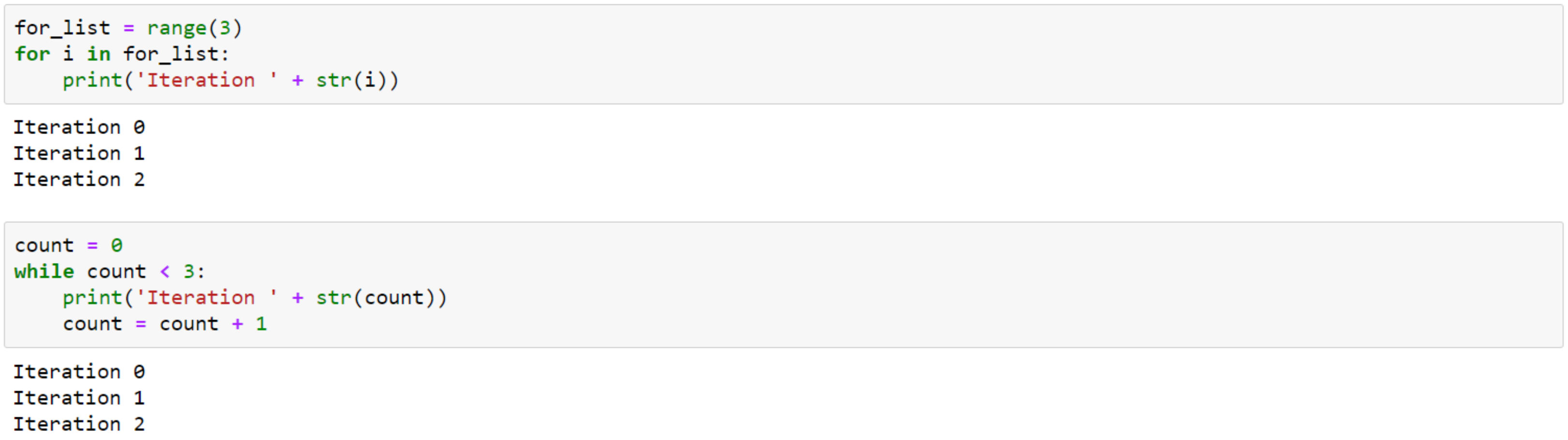
```
list(iter(a_dict))
```

```
['a_integer', 'a_float', 'a_string', 'a_tuple', 'a_list']
```

```
for key, value in a_dict.items():
    print('The value of the key \" %s \" is %s ' %(key, value))
```

```
The value of the key " a_integer " is 7
The value of the key " a_float " is 7.0
The value of the key " a_string " is seven
The value of the key " a_tuple " is (7, 7, 7)
The value of the key " a_list " is [7, 7.0, 'seven', [777]]
```

for / while / if else / def()



```
for_list = range(3)
for i in for_list:
    print('Iteration ' + str(i))
```

```
Iteration 0
Iteration 1
Iteration 2
```

```
count = 0
while count < 3:
    print('Iteration ' + str(count))
    count = count + 1
```

```
Iteration 0
Iteration 1
Iteration 2
```

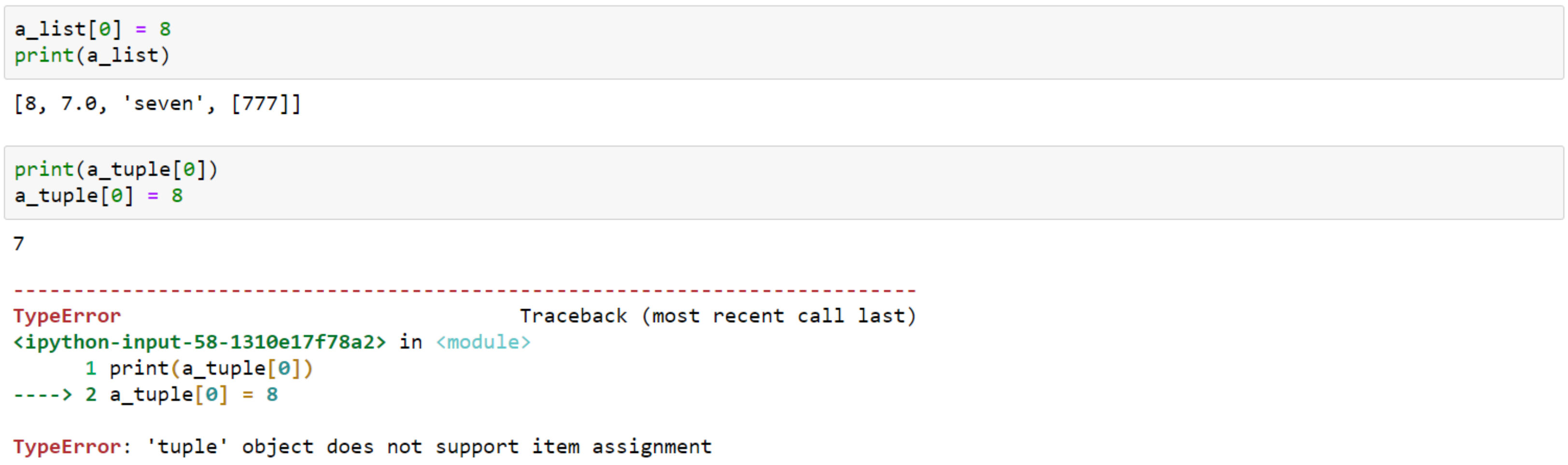
```python
if_list = [2,4,6,8]
for i in if_list:
    if i < 5:
        print('small')
    else:
        print('SUGOI DEKAI')
```

```
small
small
SUGOI DEKAI
SUGOI DEKAI
```

```python
def random_list(start, end, n):
    return list(random.randint(start, end) for _ in range(n))

random_list(100, 300, 10)
```

```
[123, 271, 147, 117, 152, 218, 242, 162, 154, 295]
```

mutable vs. immutable



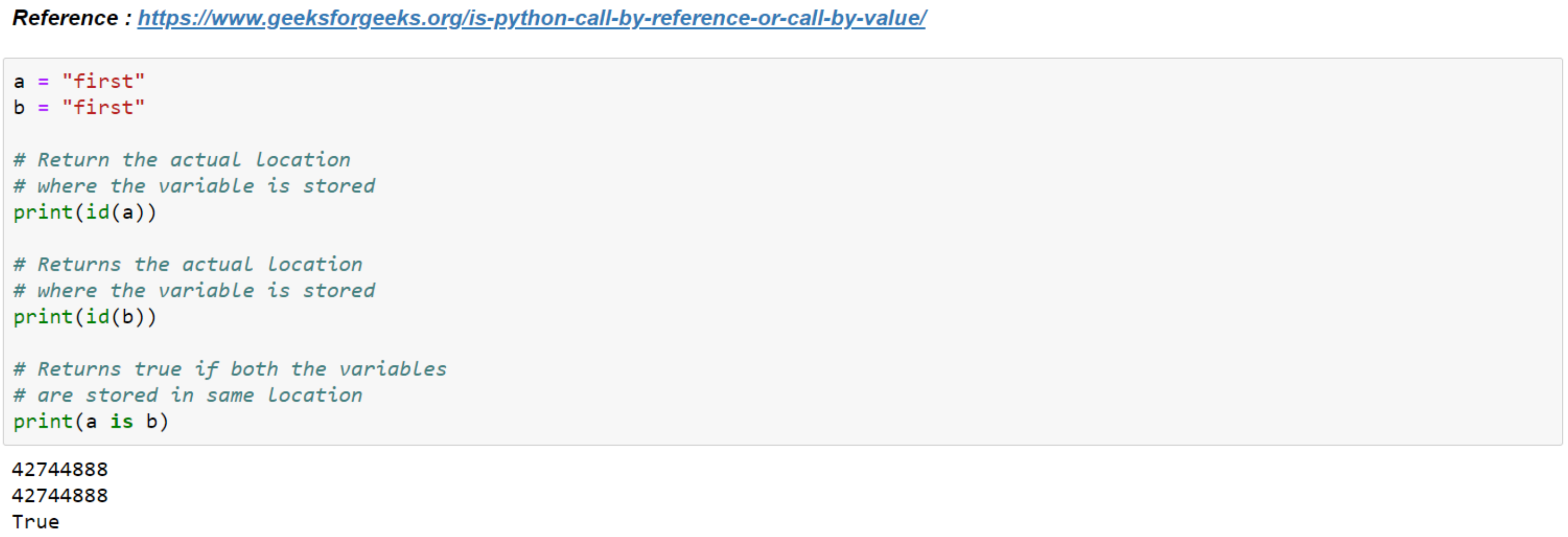
```
a_list[0] = 8
print(a_list)
```

```
[8, 7.0, 'seven', [777]]
```

```
print(a_tuple[0])
a_tuple[0] = 8
```

```
7

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-58-1310e17f78a2> in <module>
      1 print(a_tuple[0])
----> 2 a_tuple[0] = 8

TypeError: 'tuple' object does not support item assignment
```

Address - Binding Names to Objects



Reference : https://www.geeksforgeeks.org/is-python-call-by-reference-or-call-by-value/

```python
a = "first"
b = "first"

# Return the actual location
# where the variable is stored
print(id(a))

# Returns the actual location
# where the variable is stored
print(id(b))

# Returns true if both the variables
# are stored in same location
print(a is b)
```

```
42744888
42744888
True
```

```
a = [10, 20, 30]
b = [10, 20, 30]

# return the location where the variable is stored
print(id(a))

# return the location where the variable is stored
print(id(b))

# returns false if the location is not same
print(a is b)

# returns true if the locations are all the same
print(a[0] is b[0] and a[1] is b[1] and a[2] is b[2])
```

```
91308552
85554952
False
True
```

Call by assignment

```
def foo(a):

    # A new vriable is assigned
    # for the new string
    a = "new value"
    print("Inside Function:", a)

# Driver's code
string = "old value"
foo(string)

print("Outside Function:", string)
```

```
Inside Function: new value
Outside Function: old value
```

```
def foo(a):
    a[0] = "Hay"

# Driver's code
bar = ['Hi', 'how', 'are', 'you', 'doing']
foo(bar)
print(bar)
```

```
['Hay', 'how', 'are', 'you', 'doing']
```