Programming for Data Science

2nd Session: Variable Declaration, Flow Control, I/O Operations

Flávio Pinheiro
fpinheiro@novaims.unl.pt
Liah Rosenfeld
lrosenfeld@novaims.unl.pt
Maria Almeida
malmeida@novaims.unl.pt
Niclas Sturm
nsturm@novaims.unl.pt

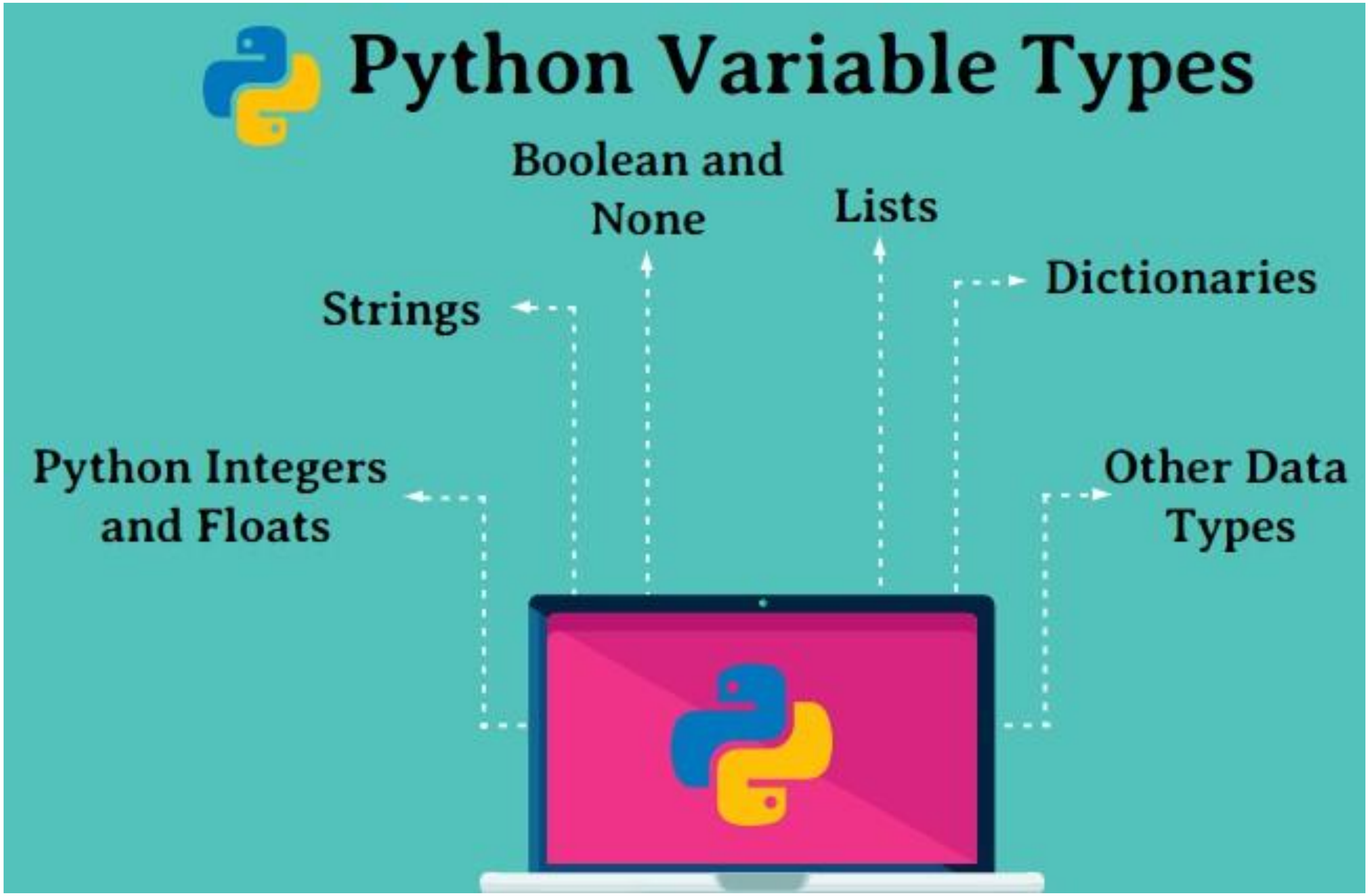
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

What’s next?

Class topics

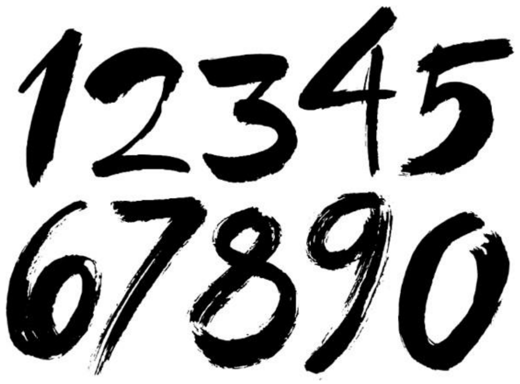
- **Variable Overview**
- Flow Control
- I/O Operations

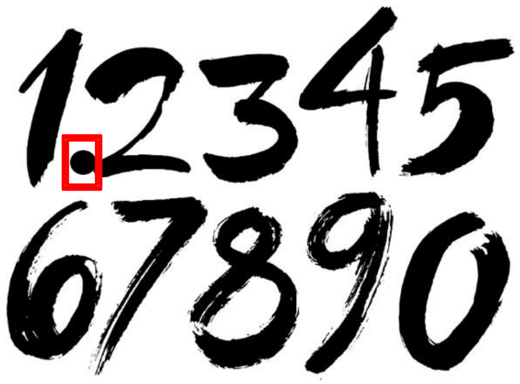
Variable Overview

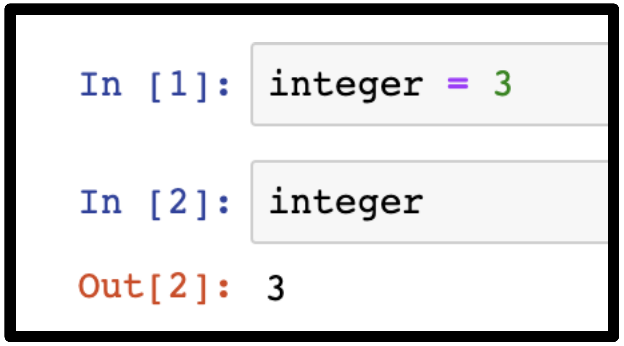


NOVA IMS Information Management School

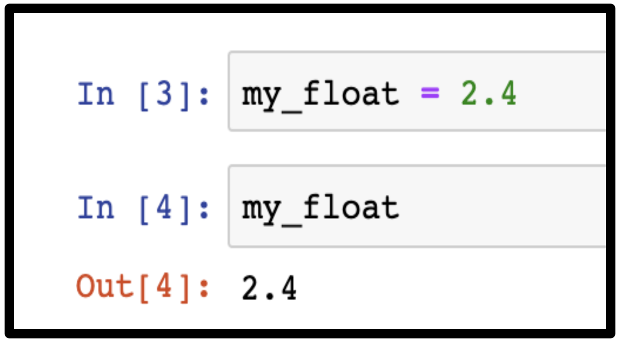
Python Integers and Floats







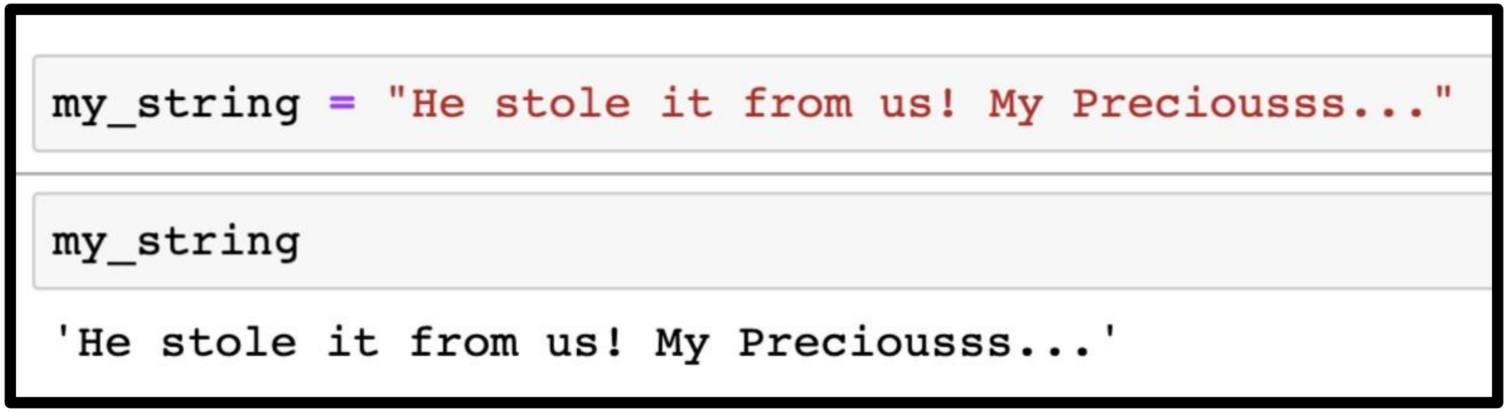
```
In [1]: integer = 3

In [2]: integer

Out[2]: 3
```



```
In [3]: my_float = 2.4

In [4]: my_float

Out[4]: 2.4
```

NOVA IMS Information Management School

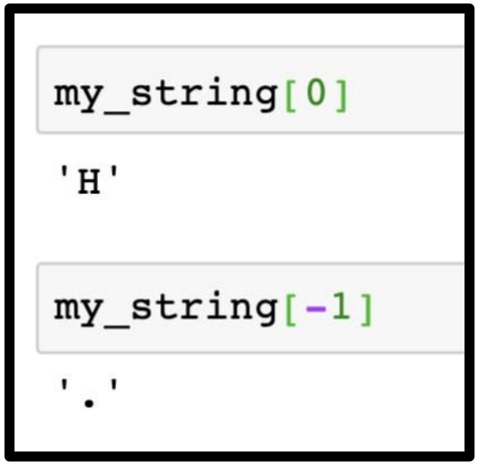
Strings



```python
my_string = "He stole it from us! My Precioussss..."
```

```python
my_string
```

```
'He stole it from us! My Precioussss...'
```

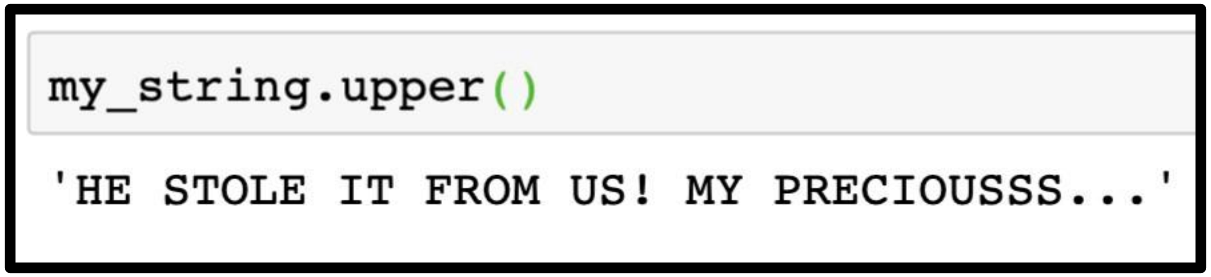


```python
my_string[0]
```

```
'H'
```

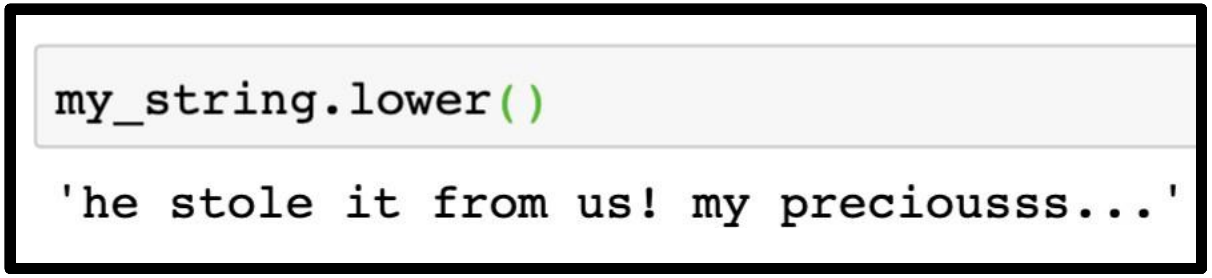
```python
my_string[-1]
```

```
'.'
```



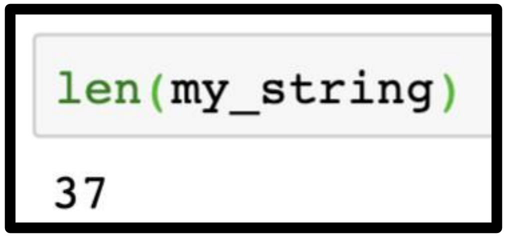
```python
my_string.upper()
```

```
'HE STOLE IT FROM US! MY PRECIOUSSSS...'
```



```python
my_string.lower()
```

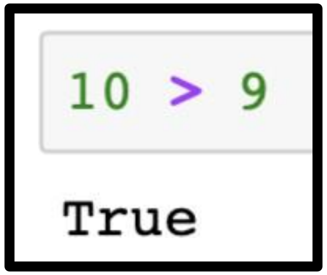
```
'he stole it from us! my precioussss...'
```



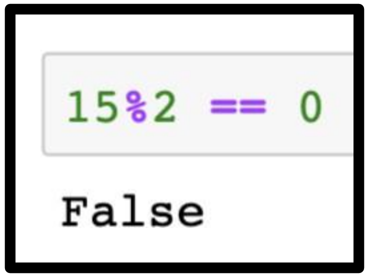
```python
len(my_string)
```

```
37
```

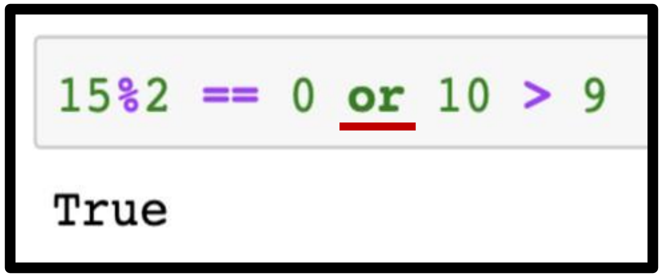
Boolean and None



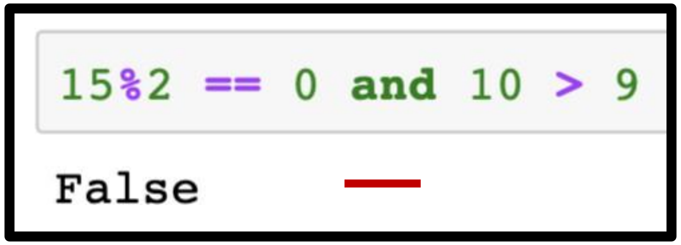
```
10 > 9
True
```



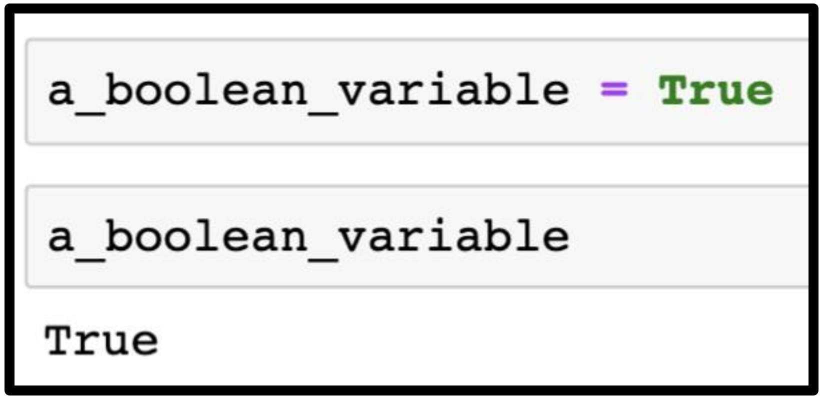
```
15%2 == 0
False
```



```
15%2 == 0 or 10 > 9
True
```

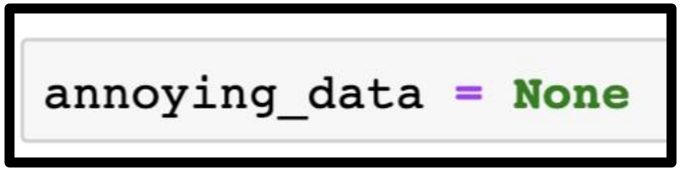


```
15%2 == 0 and 10 > 9
False
```

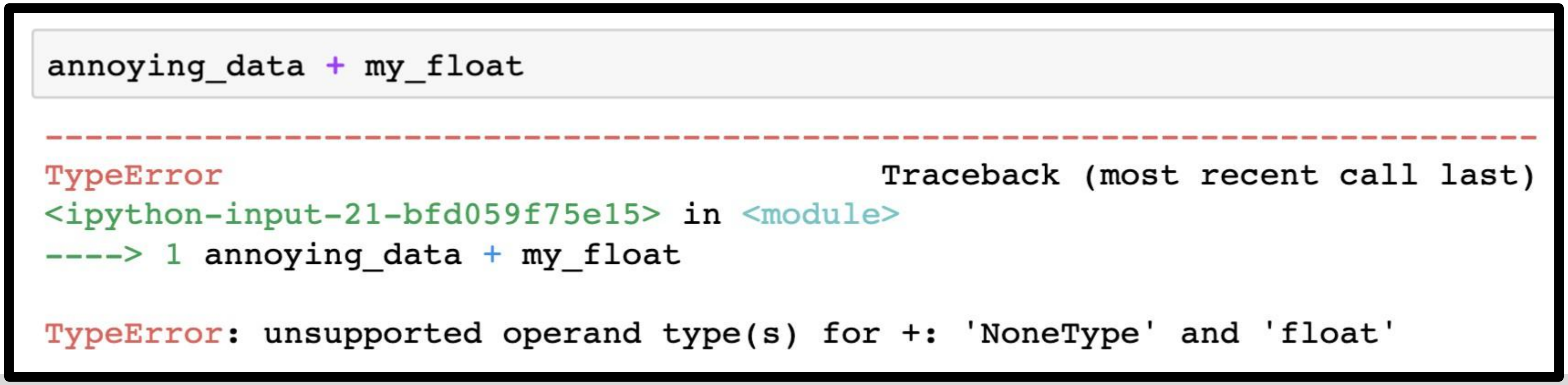


```
a_boolean_variable = True
```

```
a_boolean_variable
True
```

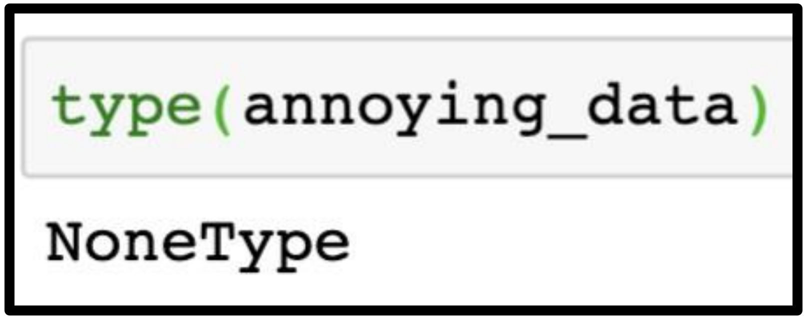


```
annoying_data = None
```



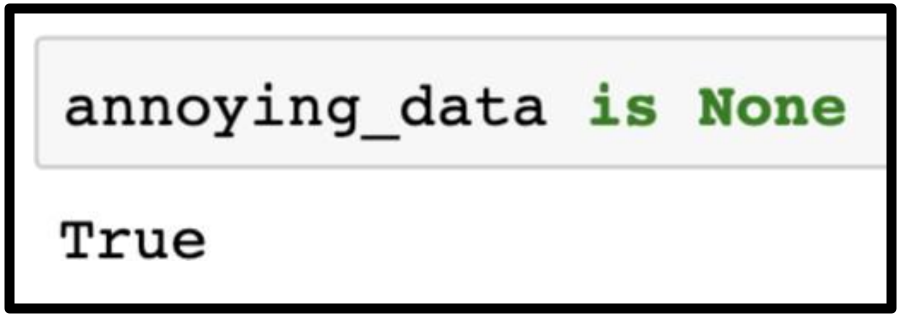
```
annoying_data + my_float
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-21-bfd059f75e15> in <module>
----> 1 annoying_data + my_float

TypeError: unsupported operand type(s) for +: 'NoneType' and 'float'
```

Boolean and None



```
type(annoying_data)
```

```
NoneType
```



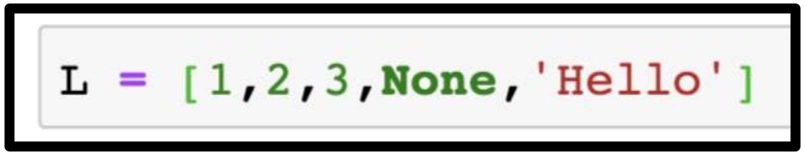
```
annoying_data is None
```

```
True
```

NOVA IMS Information Management School

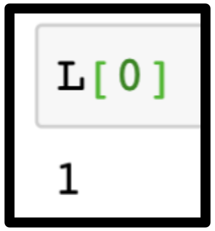
Lists

List Creation



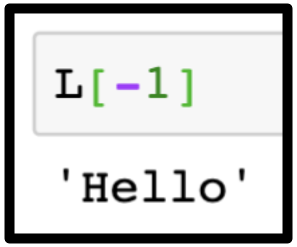
```
L = [1,2,3,None,'Hello']
```

First element



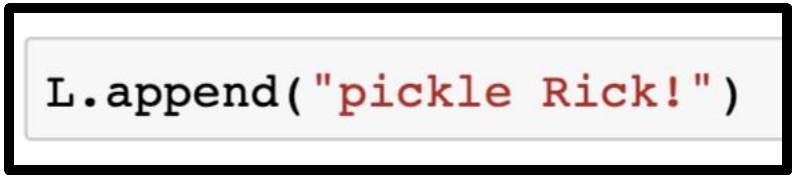
```
L[0]
1
```

Last element



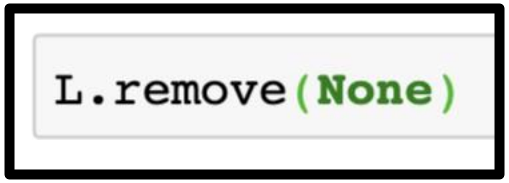
```
L[-1]
'Hello'
```

Adding a value



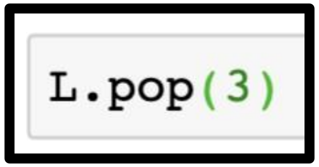
```
L.append("pickle Rick!")
```

Removing specific value

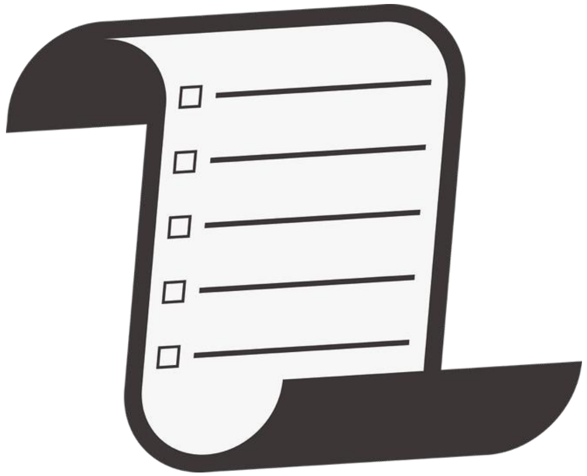


```
L.remove(None)
```

Removing by index

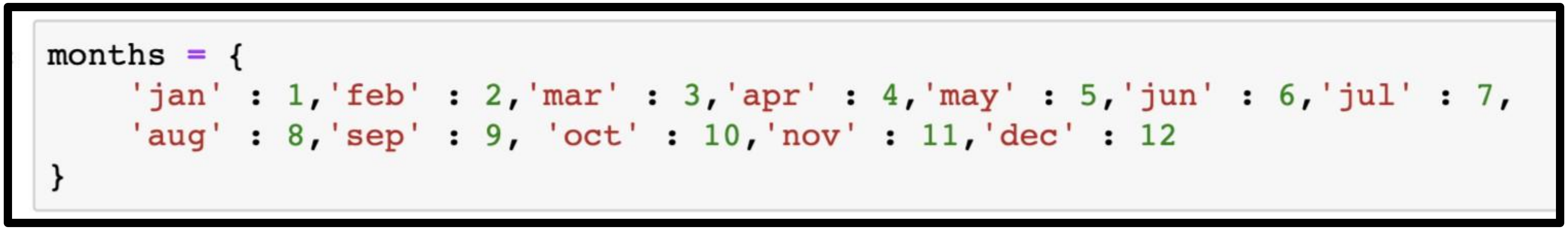


```
L.pop(3)
```



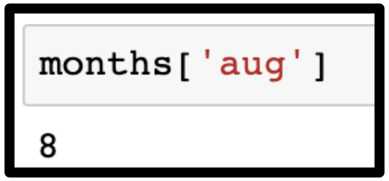
Etc...

Dictionaries



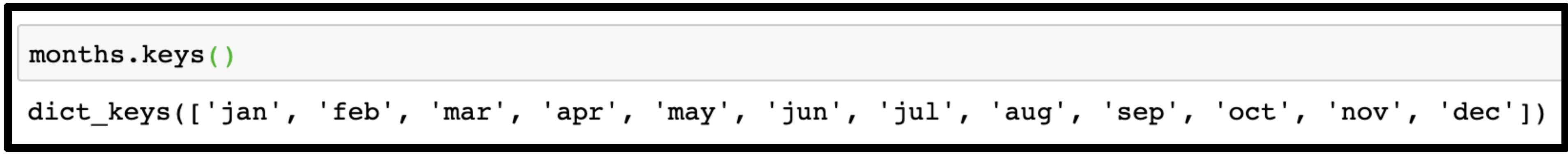
```
months = {
    'jan' : 1,'feb' : 2,'mar' : 3,'apr' : 4,'may' : 5,'jun' : 6,'jul' : 7,
    'aug' : 8,'sep' : 9, 'oct' : 10,'nov' : 11,'dec' : 12
}
```

Printing value of key



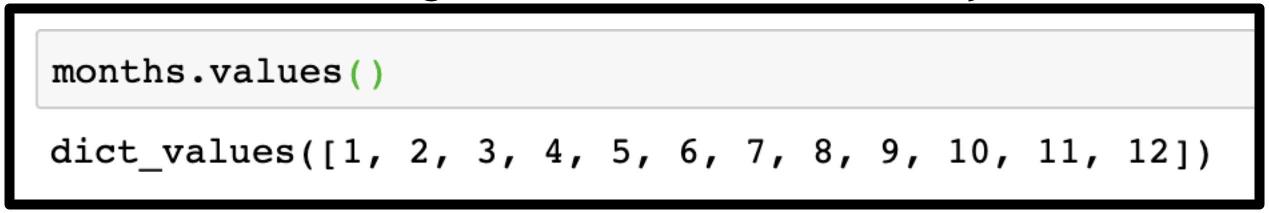
```
months['aug']

8
```

Printing the keys of the dictionary



```
months.keys()

dict_keys(['jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov', 'dec'])
```

Printing the values of the dictionary



```
months.values()

dict_values([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
```

Remember Mutability

Collection Data Structures

	Mutable	Immutable
Ordered	List	Tuple
Unordered	Dictionary	Sets

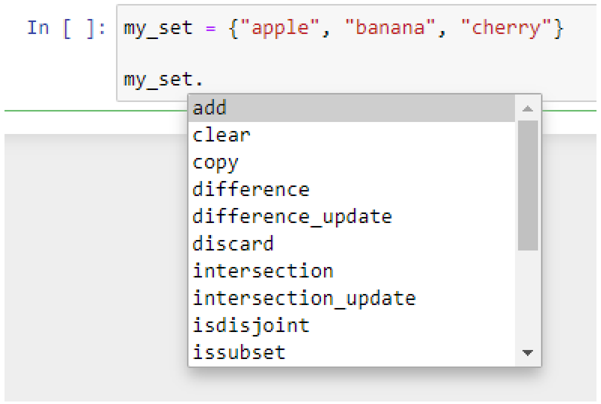
Can we change the value of an element in a data structure?

How are elements sorted? By an index or history of addition

Which will fail?

```
my_list = [1, 2, 3]
my_tuple = (1,2,3)

mylist[0] = 5
my_tuple[0] = 5
```

Use tab to see what functions we can use with our variable



```
In [ ]: my_set = {"apple", "banana", "cherry"}

        my_set.
```

We can "cast" object to different to types

```
converted_list = list(my_tuple)
converted_list
```

```
[1, 2, 3]
```

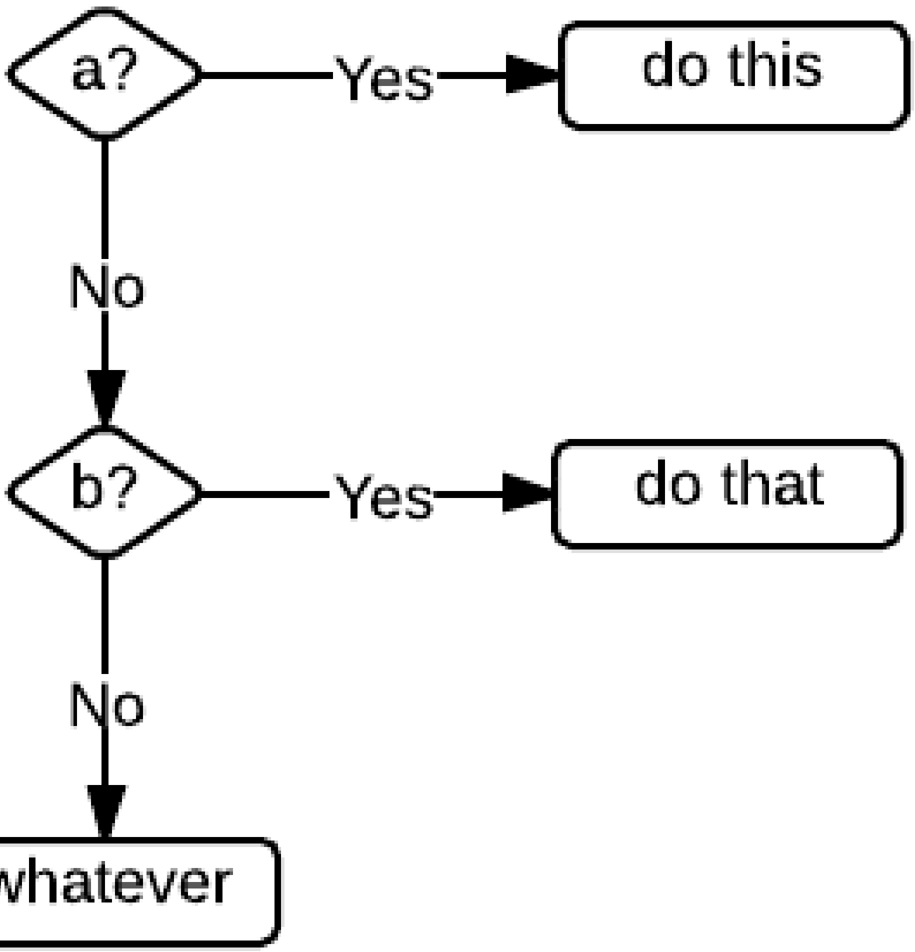
What’s next?

Class topics

- Quick reminder / variable overview
- **Flow Control**
- I/O Operations

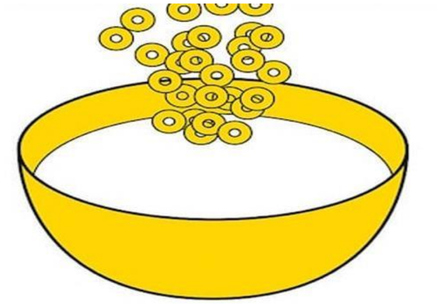
How can I control the flow of my program?

```
if a:
    do this
elif b:
    do that
else:
    whatever
```

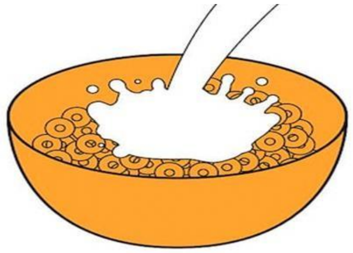


If, elif, else: Eating breakfast

If You are a sane person



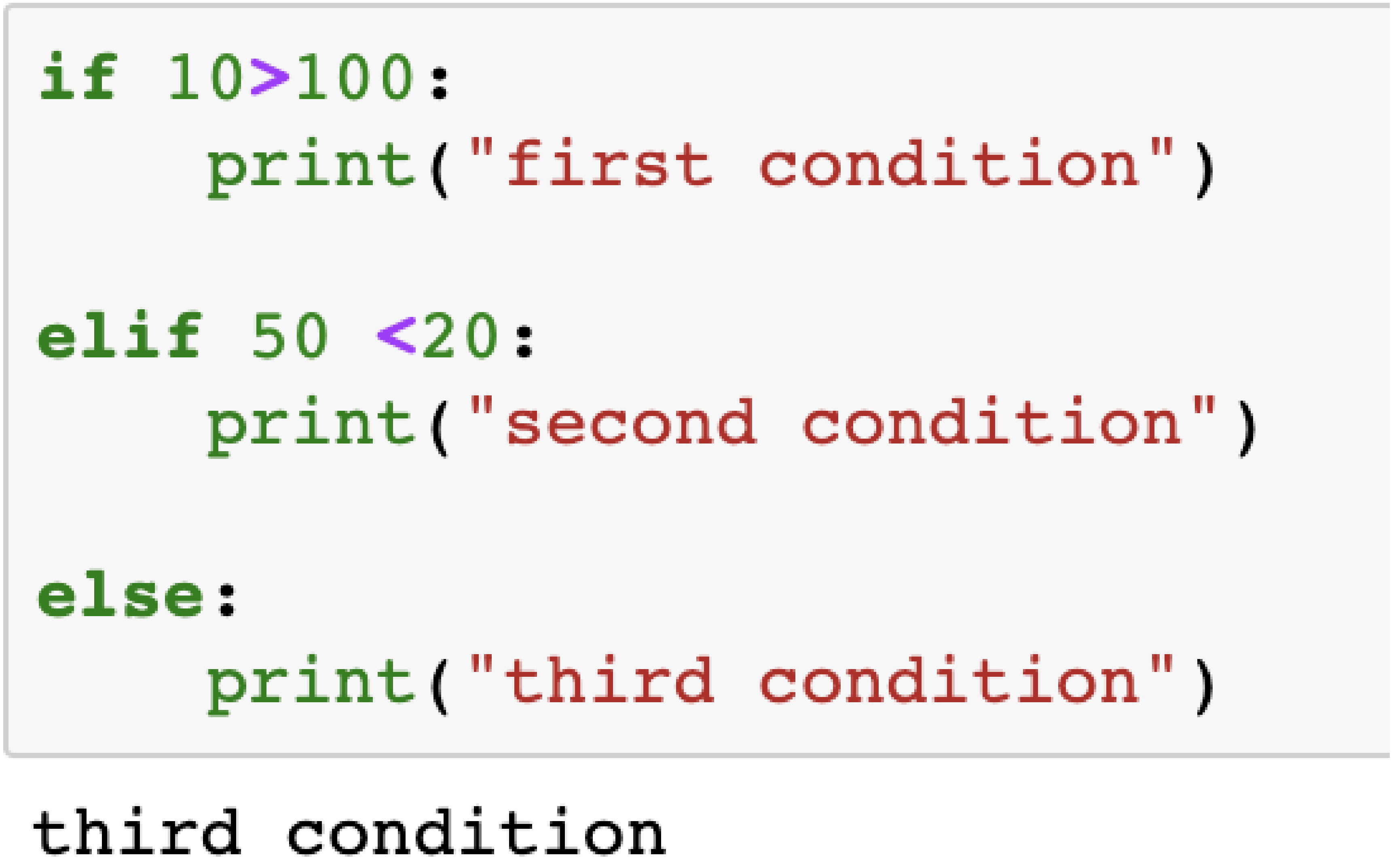
Elif You are a maniac



Else I guess no cereal for you...

NOVA IMS Information Management School

If, elif, else...

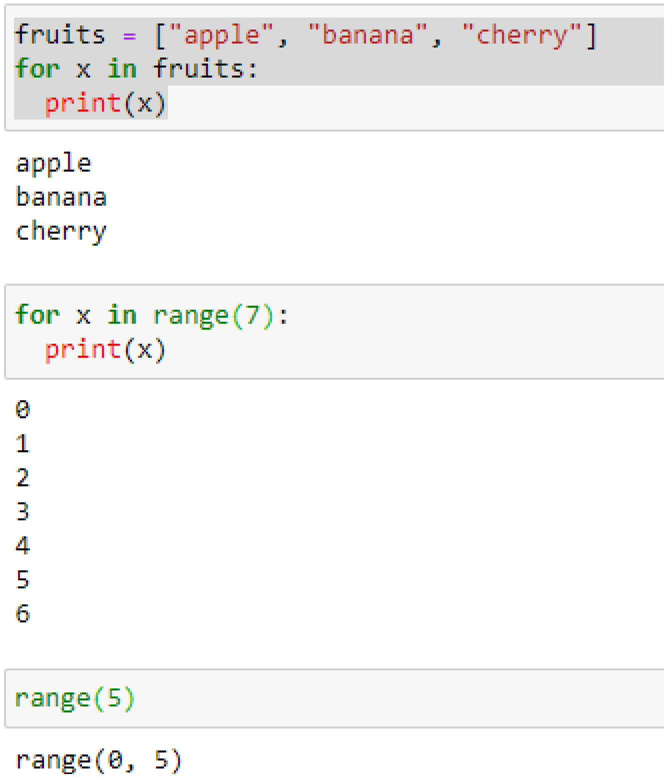


```python
if 10>100:
    print("first condition")

elif 50 <20:
    print("second condition")

else:
    print("third condition")
```

```
third condition
```

Loops - for loops

for loops are traditionally used when you have a block of code which you want to repeat a fixed number of times



```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

```
apple
banana
cherry
```

```
for x in range(7):
  print(x)
```

```
0
1
2
3
4
5
6
```

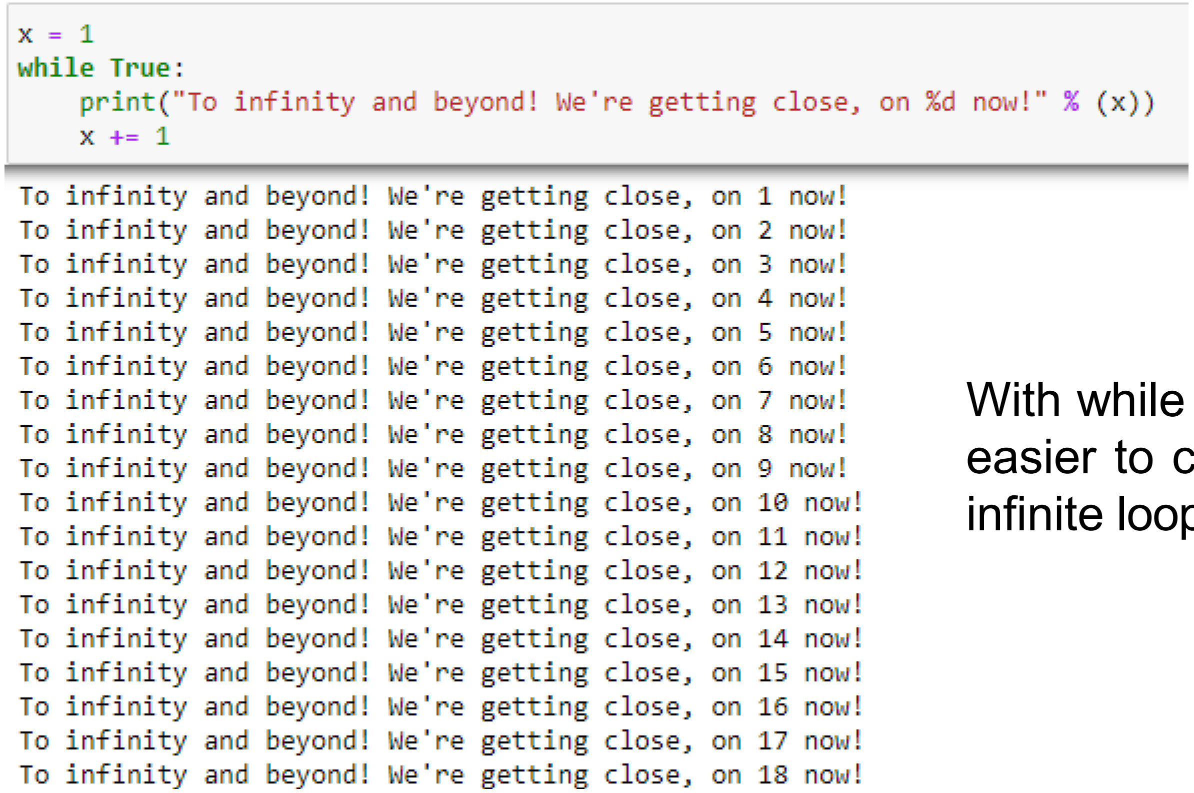
```
range(5)
```

```
range(0, 5)
```

In Python we make use of the range object

Loops – while loops

while loops can be used when a condition needs to be checked each iteration.



```
x = 1
while True:
    print("To infinity and beyond! We're getting close, on %d now!" % (x))
    x += 1
```

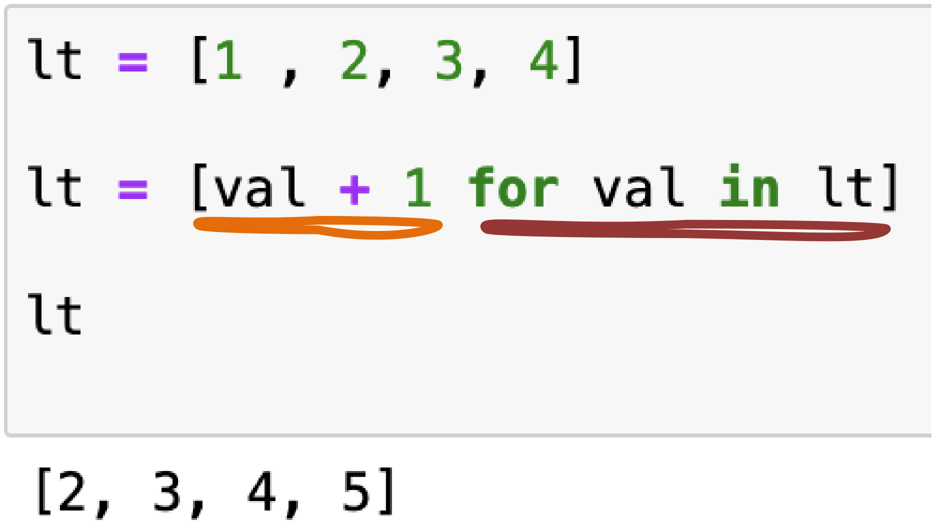
```
To infinity and beyond! We're getting close, on 1 now!
To infinity and beyond! We're getting close, on 2 now!
To infinity and beyond! We're getting close, on 3 now!
To infinity and beyond! We're getting close, on 4 now!
To infinity and beyond! We're getting close, on 5 now!
To infinity and beyond! We're getting close, on 6 now!
To infinity and beyond! We're getting close, on 7 now!
To infinity and beyond! We're getting close, on 8 now!
To infinity and beyond! We're getting close, on 9 now!
To infinity and beyond! We're getting close, on 10 now!
To infinity and beyond! We're getting close, on 11 now!
To infinity and beyond! We're getting close, on 12 now!
To infinity and beyond! We're getting close, on 13 now!
To infinity and beyond! We're getting close, on 14 now!
To infinity and beyond! We're getting close, on 15 now!
To infinity and beyond! We're getting close, on 16 now!
To infinity and beyond! We're getting close, on 17 now!
To infinity and beyond! We're getting close, on 18 now!
```

With while loops its easier to create an infinite loop

NOVA IMS Information Management School

Loops - list comprehensions

We can create a new list / iterate through a list by using *list comprehensions.*



```
lt = [1 , 2, 3, 4]

lt = [val + 1 for val in lt]

lt
```

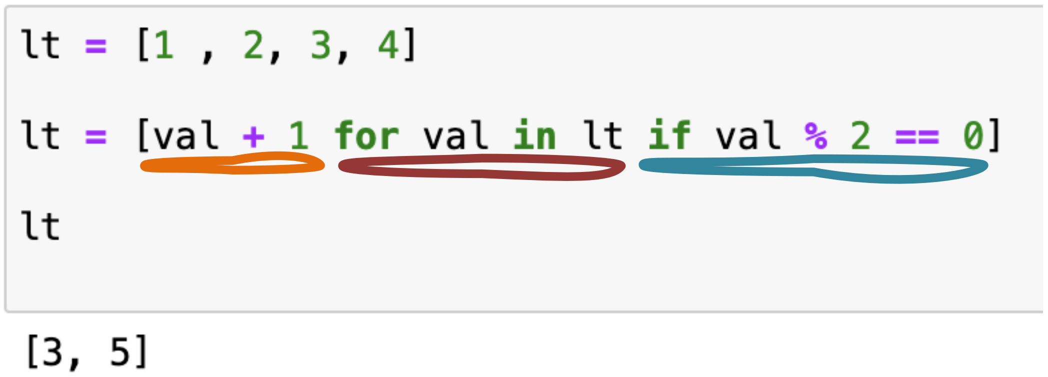
```
[2, 3, 4, 5]
```

Simple *list comprehensions* follow the following logic:

[**what is being saved / appended in each iteration how iterations occur**]

Loops - list comprehensions

More complex *list comprehensions* can have conditional statements.



```
lt = [1 , 2, 3, 4]

lt = [val + 1 for val in lt if val % 2 == 0]

lt
```

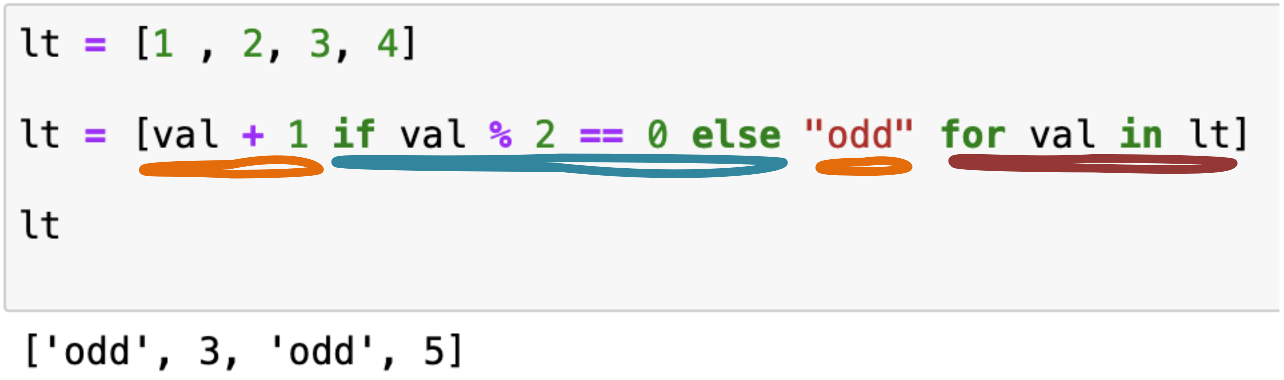
```
[3, 5]
```

list comprehensions with a <u>one way if </u>can be used for **filtering** values in the list. They follow the following logic:

[**what is being appended in each iteration** **how iterations occur** **under what condition to save what is in** 🟧]

Loops - list comprehensions

However, *list comprehensions* can also have two-way ifs (if-else) where we store different values depending on a given condition.



```
lt = [1 , 2, 3, 4]

lt = [val + 1 if val % 2 == 0 else "odd" for val in lt]

lt
```

```
['odd', 3, 'odd', 5]
```

When we use such a structure, the *list comprehension* is no longer used for filtering!

[**what to append** **if** **a certain condition is met otherwise (else)** **append this** **how iterations occur**]

What's next?

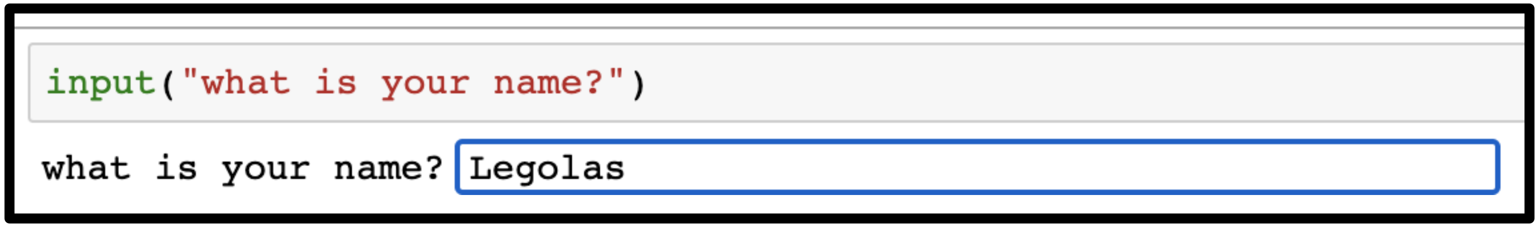
Class topics

- Variable Overview
- Flow Control
- **I/O Operations**

NOVA IMS Information Management School

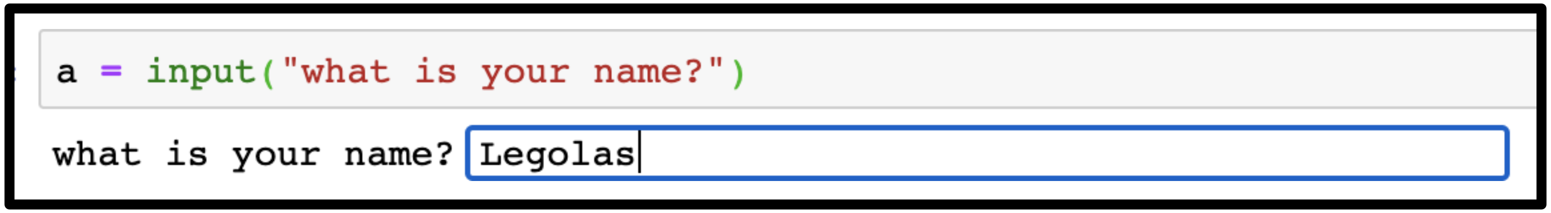
How to obtain input from a user?

- As we said, python is pretty straight forward. You want the user's input? Type **input**

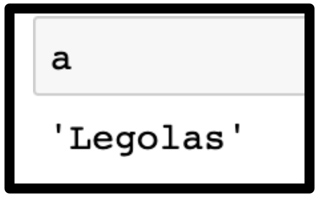


```
input("what is your name?")

what is your name? Legolas
```

- You can also save this input into a variable!



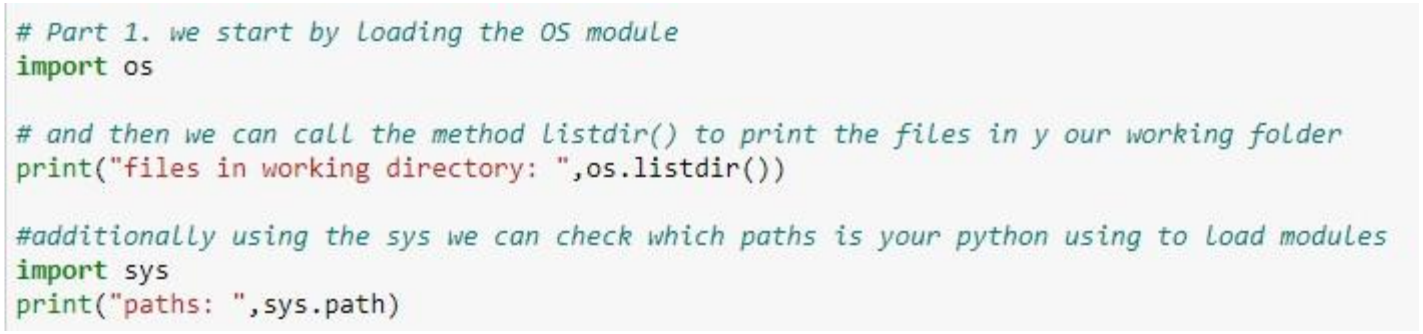
```
a = input("what is your name?")

what is your name? Legolas
```



```
a

'Legolas'
```

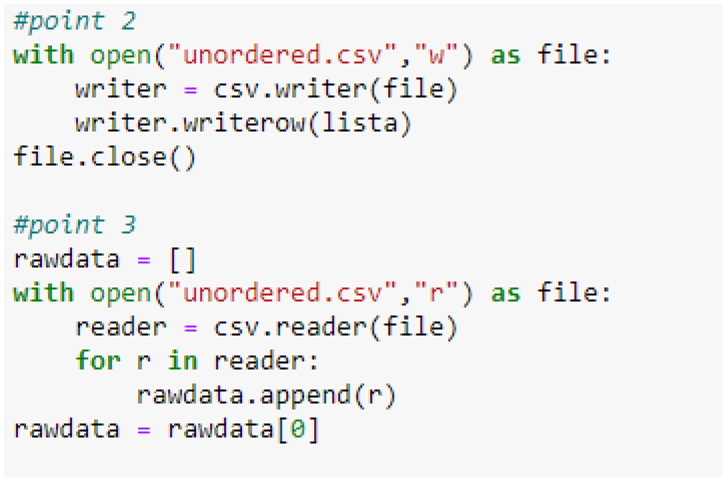
How to load a dataset so that we can start exploring?

Accessing the filesystem (or you can use magics):



```python
# Part 1. we start by loading the OS module
import os

# and then we can call the method listdir() to print the files in y our working folder
print("files in working directory: ",os.listdir())

#additionally using the sys we can check which paths is your python using to load modules
import sys
print("paths: ",sys.path)
```

Writting to and reading from files:



```python
#point 2
with open("unordered.csv","w") as file:
    writer = csv.writer(file)
    writer.writerow(lista)
file.close()

#point 3
rawdata = []
with open("unordered.csv","r") as file:
    reader = csv.reader(file)
    for r in reader:
        rawdata.append(r)
rawdata = rawdata[0]
```

See you next week ☺

me: accidentally adds one extra space
python:

End