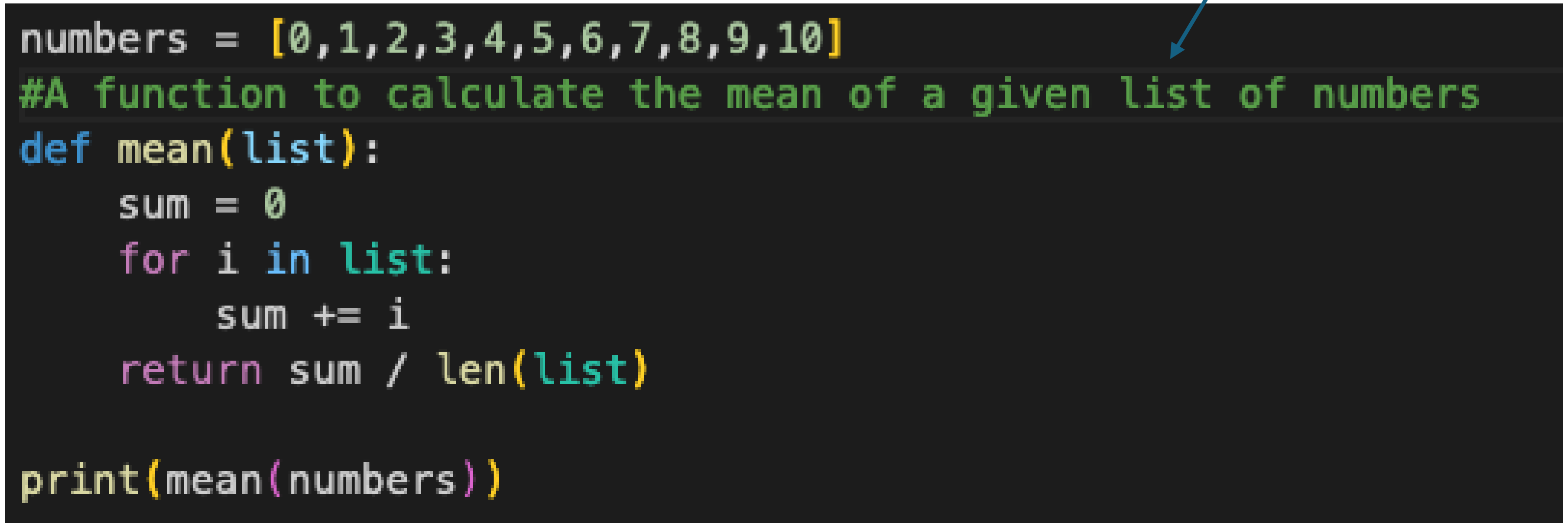
Basic Python Functions

First Tutorial for 3CP3 class

Python Indentation and basic syntax

- Python uses whitespace and indentation to construct the code structure

A comment that is not executed



```python
numbers = [0,1,2,3,4,5,6,7,8,9,10]
#A function to calculate the mean of a given list of numbers
def mean(list):
    sum = 0
    for i in list:
        sum += i
    return sum / len(list)

print(mean(numbers))
```

Whitespace to define the block of coding

- More readable and uniform

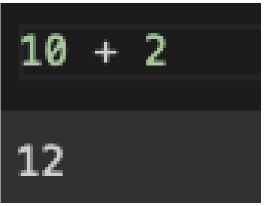
- Python is case sensitive, so it encourages precision and clarity while coding.

Basic operations

- You can compute basic operations directly.

Type on your jupyter notebook
(Google Colab)

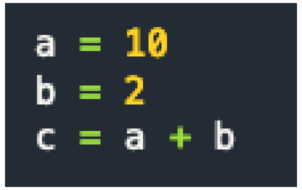
Get the answer



```
10 + 2
```

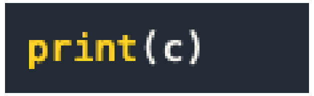
```
12
```

- But working with multiple variable is useful to assign each variable a name.



```
a = 10
b = 2
c = a + b
```

- And then you can print the variable that you stored the operation.

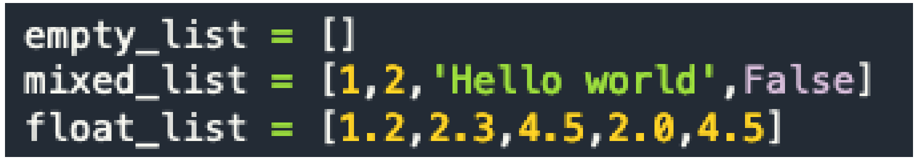


```
print(c)
```

- You can print multiple variables, strings, arrays…

List and operations with lists

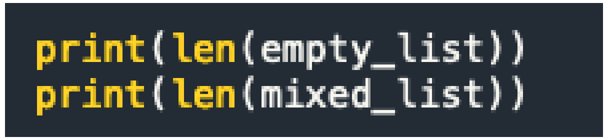
- You can create lists and populate them with numbers and strings.



```
empty_list = []
mixed_list = [1,2,'Hello world',False]
float_list = [1.2,2.3,4.5,2.0,4.5]
```

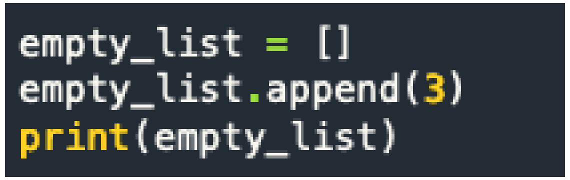
- Lists are:
  1. Ordered
  2. Mutable
  3. Denoted by square brackets

- You can check the length of a list



```
print(len(empty_list))
print(len(mixed_list))
```

- You can add more variables to your list by using the *append* command

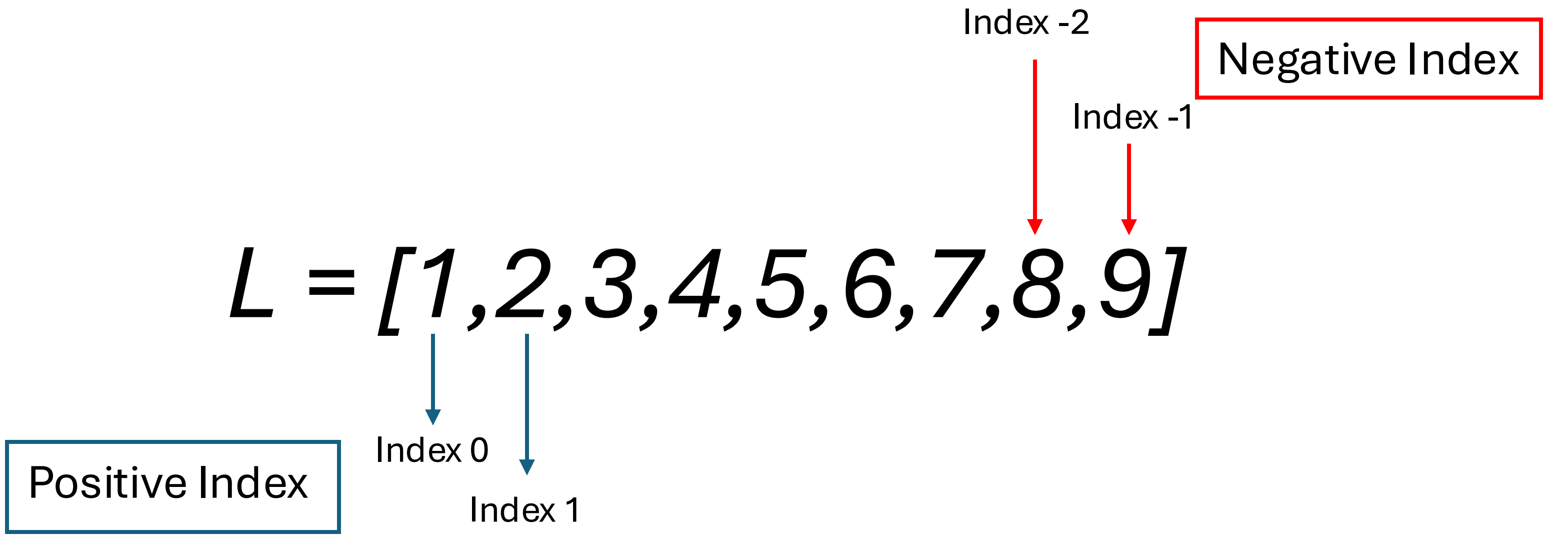


```
empty_list = []
empty_list.append(3)
print(empty_list)
```

- Given a list *L = [1,2,3,4,5,6,7,8,9]* . We can access a specific position of the list using slicing.
  - The whole list: *L[:]*
  - Everything after (and including) index position i : *L[i:]*
  - Everything before index position i: *L[:i]*
  - Everything before the position j steps from the end: *L[:-j]*
  - Everything after (and including) the position j steps from the end: *L[-j:]*

- *L[0] = [1]*
- *L[2:] = [3,4,5,6,7,8,9]*
- *L[:4] = [1,2,3,4]*
- *L[-2:] = [8,9]*
- *L[:-6] = [1,2,3]*

- Note



Index -2

Negative Index

Index -1

L = [1,2,3,4,5,6,7,8,9]

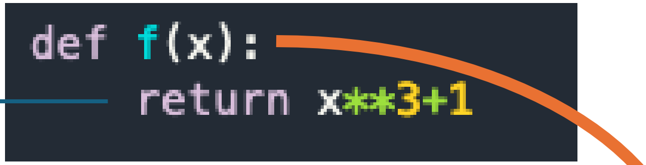
Index 0

Positive Index

Index 1

Functions

- A function in Python works the same as a function in math: you define an input and an output.



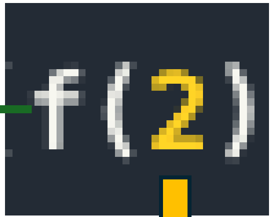
```
def f(x):
    return x**3+1
```

Output

  - Functions in Python are defined by the def keyword.
  - And you put the list of outputs inside a parenthesis followed by :

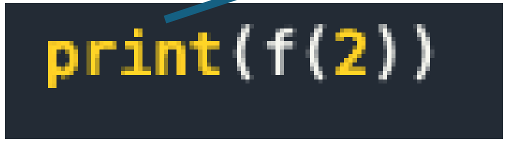
- This defines the function $f(x) = x^3 + 1$ and to evaluate the function in each input you do,

Call the function by its name



```
f(2)
```

Put the input inside the parenthesis. This is the value that you want to evaluate.

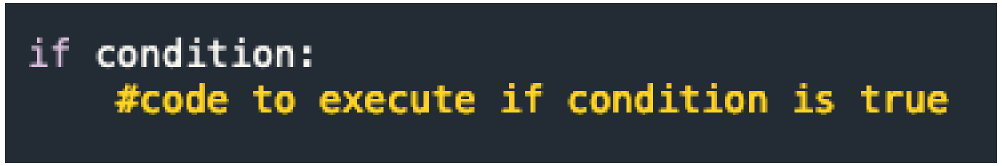


```
print(f(2))
```

Print the output

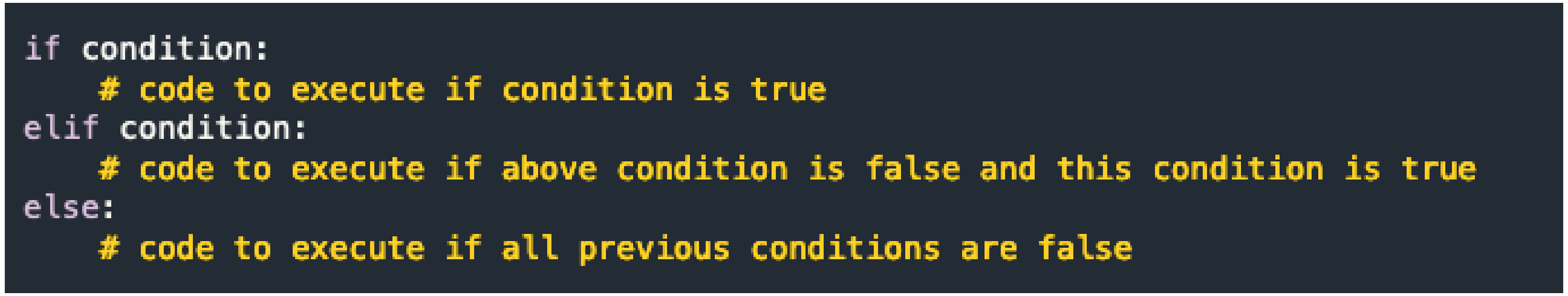
Conditional Statements

- There are instances where we want to only execute a particular block of code if a certain condition is true.



```
if condition:
    #code to execute if condition is true
```

- For multiple conditions, the syntax is,

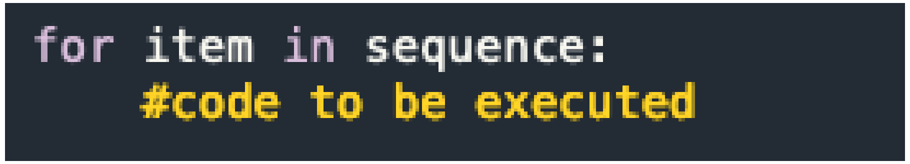


```
if condition:
    # code to execute if condition is true
elif condition:
    # code to execute if above condition is false and this condition is true
else:
    # code to execute if all previous conditions are false
```

- Comparison operations,
    - Equals x == y
    - Not Equal x != y
    - Less Than (strictly) x < y
    - Greater Than (strictly) x > y
    - Less Than or Equal to x <= y
    - Greater Than or Equal to x >= y

Loop

- When programming, there are times when you need to repeatedly perform a specific operation/action while updating certain parameters. In these situations, we use loops,



```
for item in sequence:
    #code to be executed
```

Exercise

- Test the convergence of the alternating series,

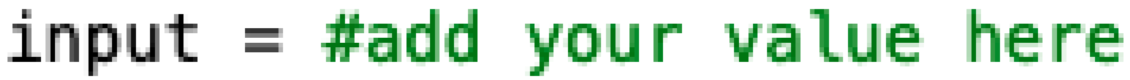
$$\sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n}$$

→ Converges to ln(2)

Quick Feedback on Quiz 1

- Pay attention to the information on the website on how to send the files. You need to follow the template there otherwise your quiz **will not** be accepted.
- You don’t need to write #return print() (This can lead to errors)
- For the next assignments you can delete the #Write your code here.
- Do not write the exercise asking for the input of an user:

Like:

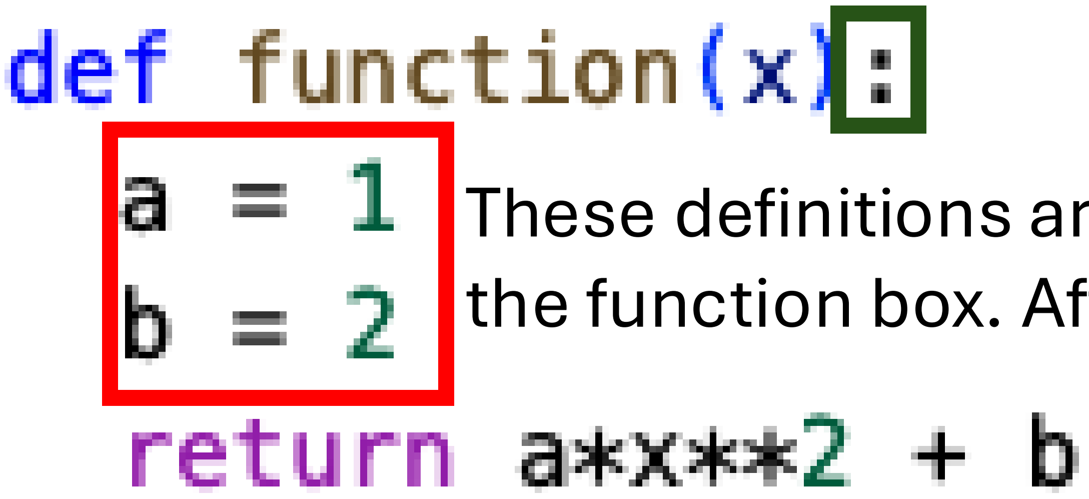


```
input = #add your value here
```

The code must work on its own.

Last Class

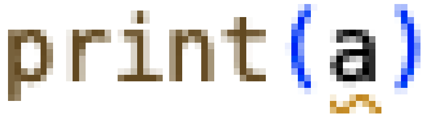
- I think it is still not super clear how does the definition of a function works using Python.



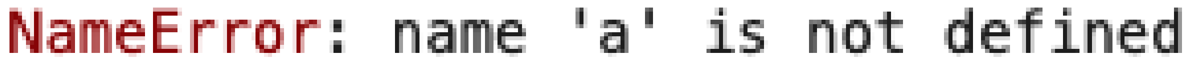
```
def function(x):
    a = 1
    b = 2
    return a*x**2 + b
```

These definitions are only valid inside the function box. After the colon.

What happens if I try to print the value of ‘a’ outside the function ?

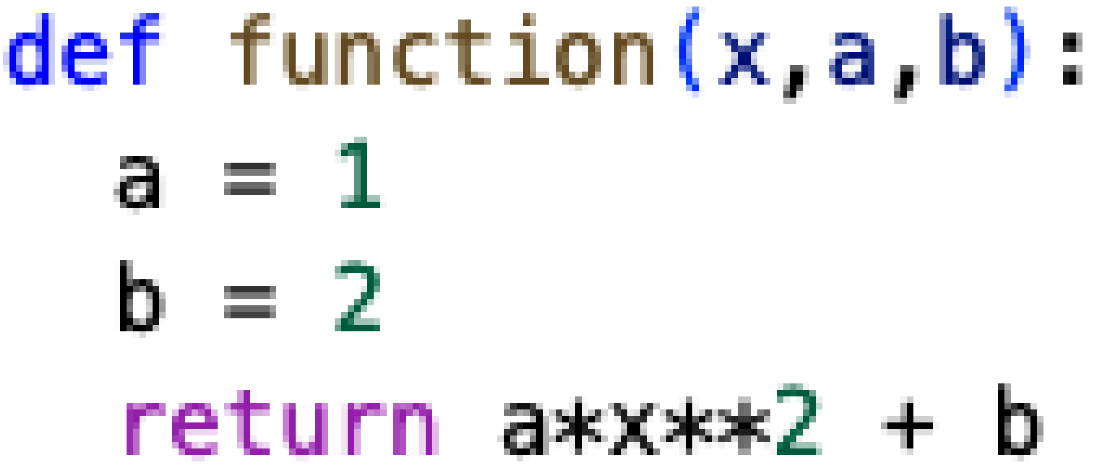


```
print(a)
```



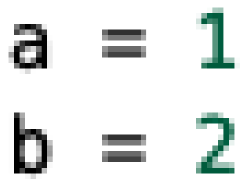
```
NameError: name 'a' is not defined
```

A function can have many variables



```
def function(x,a,b):
    a = 1
    b = 2
    return a*x**2 + b
```

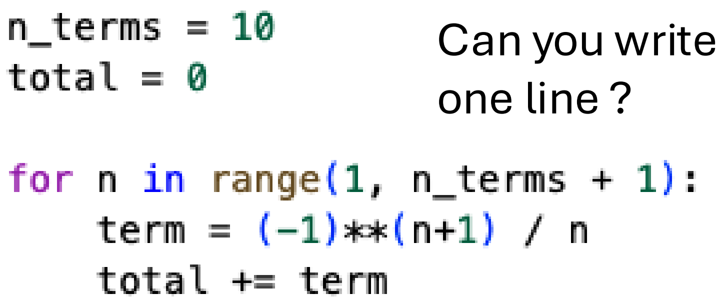
What if we want to change the values of ‘a’ and ‘b’ ?



```
a = 1
b = 2
```

Written like this the values of the variables are immutable inside the function.

A function can have many variables



```
n_terms = 10
total = 0

for n in range(1, n_terms + 1):
    term = (-1)**(n+1) / n
    total += term
```

Can you write this in one line ?

$$\sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n}$$

Converges to ln(2)

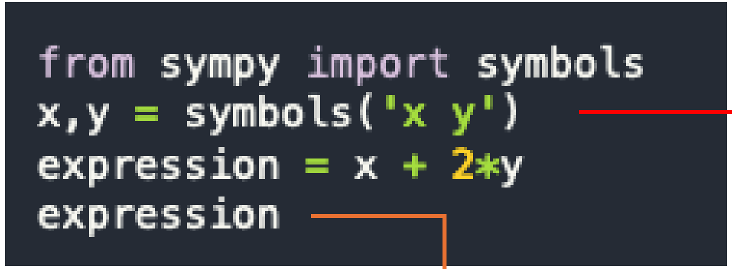
$$\sum_{i=1}^{3} \sum_{j=1}^{2} (i + j)$$

How to code ?

Derivatives

- You can evaluate derivatives and integrals symbolically using the SymPy library.

Call the library



```
from sympy import symbols
x,y = symbols('x y')
expression = x + 2*y
expression
```

Define x and y as symbols

$$x + 2y$$

- Calculate the derivative of the logistic function by hand and using SymPy.
- Write the derivative as $f'(x) = f(x)(1 - f(x))$

$$f(x) = \frac{1}{(1 + e^{-x})} = \frac{e^{x}}{1 + e^{x}} \longrightarrow$$

Logistic function