Python workshop EIEIOO 2024

Jean-Marie Coquillat &
Annabelle Makowski

Arthur B. McDonald
Canadian Astroparticle Physics Research Institute

Queen's UNIVERSITY

What we are going to do today

Learn python3

Look at examples

Practice exercises

Fight snakes (if time allows)

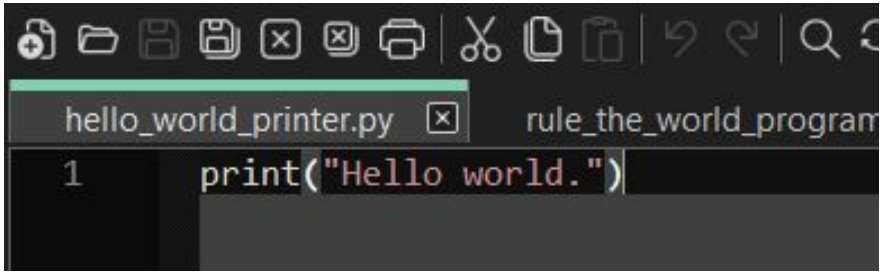
Python: what is it?

- Programming language
- Developed by Benevolent Dictator For Life Guido van Rossum in 1989 during his Christmas vacations.
- Easy to learn
- Simple syntax
- Slower than more complex language
- Perfect to write quick codes
- Has great documentation
- Preferred language for data analysis and machine learning.

Two ways to code in python

SCRIPT MODE

The code is already written, and is run at the end all at once.



INTERACTIVE MODE

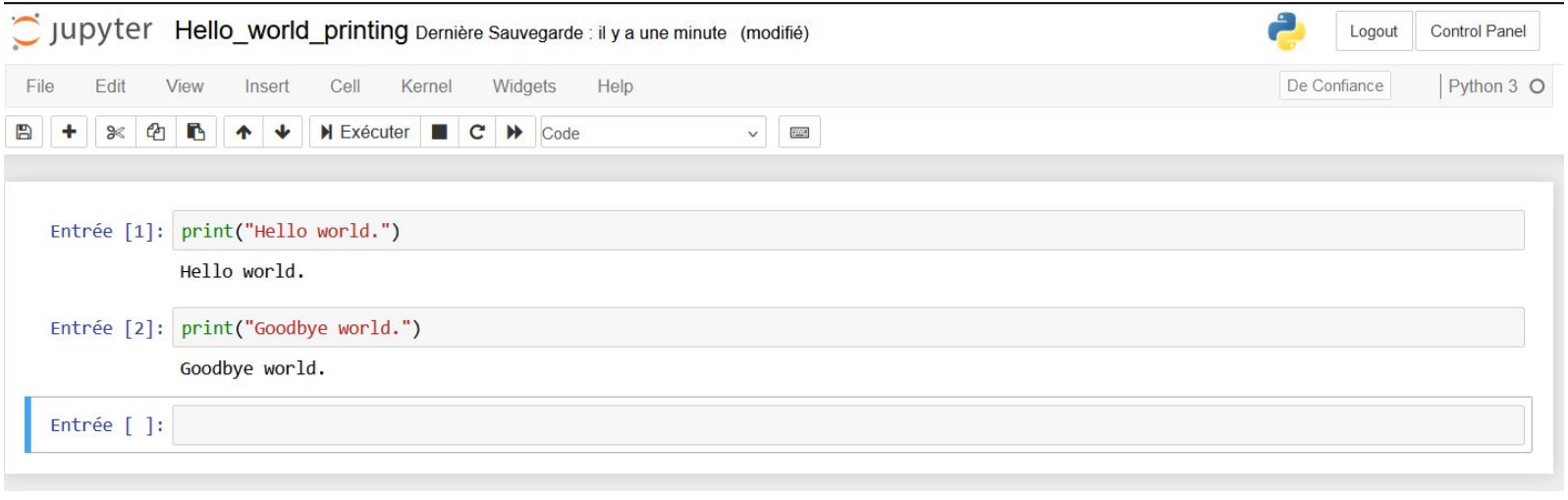
The code is run line by line, as they are written.

Jupyter Notebooks

In-between of both python modes

Code is run in blocks called “cells”

Queen’s students can use https://queensu.syzygy.ca/jupyter/



Basic mathematical operations

Operation	Symbol	Example
Assigning a variable	=	x=4
Addition	+	5+5 → 10
Subtraction	-	5-6 → -1
Multiplication	*	3*7 → 27
Division	/	7/2 → 3.5
Integer division	//	7//2 → 3
Exponentiation	**	2**3 → 8
Imaginary number	j	1j**2 → -1+0j
Modulo (remainder)	%	10%4 → 2
Powers of 10 (order of magnitude)	e	3e9 → 3×10^{9}

Credit: Hannah Fronenberg

Extra:

- Use parenthesis if needed
- Increment with **+=, -=, *=, /=.**

```
>>> x=1000
>>> x+=1
>>> print(x)
1001
```

x+=1
x=x+1

- Also, beware of floating-point imprecision

```
>>> 0.3-0.1-0.1-0.1
-2.7755575615628914e-17
```

Main objects in python

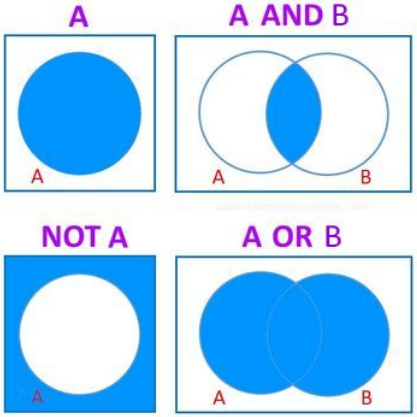
Object	Description	Examples
Boolean	Binary object, either True=1 or False=0.	True, False.
Integer (int)	Positive or negative integer number.	0, 1, -3, 42, -6472
Float	Positive or negative decimal number.	0.5, -12.34, 11.1111111
Complex	Number in the complex plane (use j for i)	0+1j, 12.5-34.5j, -10+1.2j
String	String of characters, inside " " or ' '	'0' , "Hello world." , ':)'
List	List of different objects, inside [ ]	[1,2,3], ["a", 2.5, [True,0.5]]
Tuple	Immutable list of same type of objects, inside ( )	(1,2,3), (True, False, False)
Numpy.array	Vectorized list from the numpy library	np.array([1,2,3])
Dictionary (dict)	Unordered list, with values associated to keys	{"a":1, "b":2, "c":3}
Function/Method	Function that can act, or output an object, often taking one or more objects as input.	print(), range(), type()

Logic and conditions

```
if a>b:
    print("a is greater than b")
else if a<b:
    print("b is greater than a")
else:
    print("a is equal to b")
```

- Use logic statements to check if an (in)equality is **True** or **False**.
- Use logic or bitwise operators to combine logic statements.
- Use an **if** block to only execute indented code if it follows specific logic conditions.

Condition	Symbol	Example
Equal to	==	2+2 == 4 → True
Different from	!=	2*3 != 6 → False
Greater/lesser than	> / <	3 > -5 → True
Greater/lesser or equal to	>= / <=	1/3 < 0.25 → False



Logic operator	Bitwise operator
and	&
or	|
not	~

Loops

- Use a **while** loop to repeat executing code as long as a condition is met.
- Use a **for** loop to iterate over all elements of a list and repeat execution code for each iteration.
- Just like for **if** blocks, use indentation (4 spaces or 1 tab) after a **:** to select which part is inside the loop.
- You can nest a loop inside another loop.
- The **break** command will immediately end a loop.
- The **continue** command will stop the current iteration and skip to the next one.

```
for i in range(0,4,1):
    print(i)
```

```
n=0
while n<4:
    print(n)
    n+=1
```

Functions

```
def quad(a,b,c,both=True):
    pos=(-b+(b**2-4*a*c))/(2*a)
    neg=(-b-(b**2-4*a*c))/(2*a)
    if both: return [pos,neg]
    else: return pos
```

- Functions can be called to execute a piece of code, with optional inputs and output.
- Define a function with the **def** operator. You can give mandatory inputs and optional inputs that have a default value.
- Import function from libraries with the **import** operator.

Useful function	Description	Example
print(object)	Prints the value of the input.	print("1+2=",1+2) → 1+2= 3
type(object)	Returns type of the input.	type([23,4.5]) → <clasee 'list'>
int(numeral)	Returns the input converted to the integer type.	int(3.9) → 4
abs(number)	Returns the absolute value of the input.	abs(-10) → 10
range(min=0, max, step=1)	Returns a "list" starting at min, ending before max, with jumps of size step.	range(3) → [0,1,2] range(4,8,2) → [4,6]
len(object)	Returns the length of the input.	len("alphabet") → 8
LIST.append(element)	Appends the input at the end of the list it is applied to.	x=[1,2,3] x.append(7) print(x) → [1,2,3,7]

The numpy library

```
import numpy as np
```

Function	Description	Example
np.array(list,dtype=float)	Returns a vectorized numpy array from a list. Can be multidimensional like a matrix.	x = np.array([[1,2,3],[4,5,6]]) x[0,1] → 2.0 x[:,-1] → np.array([3,6])
np.arange(min=0,max,step=1)	Same as range, but outputs an array and can contain floats instead of only integers.	np.arange(0.25,1.75,0.5) → np.array([0.25,0.75,1.25])
np.linspace(min,max,N)	Same as np.arange, but you input the number of equally spaced elements instead of the step size .	np.linspace(0.25,1.75,3) → np.array([0.25,1.0,1.75])
np.zeros(N) / np.ones(N)	Returns an array with only {input} zeros / ones	np.zeros(5) → np.array([0,0,0,0,0])
np.min(arr) / np.max(arr)	Returns the minimum / maximum of the input	np.min([-2,41,10]) → -2.0
np.mean(arr) / np.median(arr)	Returns the mean / median of the input	np.mean([0,5,4]) → 3.0
np.sum(arr) / np.std(arr)	Returns the sum / standard deviation of the input	np.sum([-5,1,-3]) → -7.0
np.sort(arr) / np.argsort(arr)	Returns a sorted version of the input / the sorted argument of the input	np.sort([6,2,0]) → np.array([0,2,6]) np.argsort([6,2,0]) → np.array([2,1,0])
np.transpose(arr)	Returns a transposed version of the input	np.transpose([[1,2,3],[4,5,6]]) → np.array([[1,4],[2,5],[3,6]])
np.loadtxt(txtFile)	Imports data from a .txt file and returns it as a numpy array	f = np.loadtxt("datafile.txt")
np.sqrt(x) / np.exp(x) / np.pi	Returns $\sqrt{x}$ / e^x / π	np.exp(np.sqrt(np.pi)) → 5.885277…

The matplotlib library

```
import matplotlib.pyplot as plt
```

Function	Description	Example
plt.plot(x,y)	Plots x vs y	plt.plot(x, y, 'k.', label='data') plt.plot(x, x**2, 'r-', label='fit')
plt.errorbar(x,y,dy,dx)	Plots x vs y with errorbars	
plt.semilogx(x,y) / plt.semilogy(x,y)	Plots x vs y, with a logarithmic x / y axis	
plt.hist(data,nBins,[min,max])	Plots a 1D histogram of the data	h0 = plt.hist(x,50,[0,100],alpha=0.5)
plt.hist2d(xdata,ydata,[nBinsX, nBinsY],[[xmin,xmax],[ymin,ymax]])	Plots a 2D histogram of the data	plt.hist2d(x,y,[20,50],[0,1],[0,100])
plt.title(string)	Adds a title to the plot	plt.title("My beautiful graph")
plt.xlabel(string) / plt.ylabel(string)	Adds a label to the x / y axis	plt.xlabel("Time (s)")
plt.legend()	Adds a legend with the labelled plots	
plt.axis([xmin,xmax,ymin,ymax])	Selects the limits of each axis	plt.axis([-1,1,0,100])
plt.show()	Prints all the plots and clears them	

Other useful libraries

- **from scipy.optimize import curve_fit**
  curve_fit(function,xdata,ydata,p0)
  Enables you to fit a certain function to data, starting with parameters of the function p0
- **from scipy import stats**
  Contains multiple probability functions, from which you can get random variables or the probability density.
- **import pandas as pd**
  Useful data analysis library
- **import time**
  Useful for timing code
- Look online for the documentation on the functions you want!

Now, work and show what you learn!

Extra slides

Versions

[Slide stolen from last year's EIEIOO talk by Hannah Fronenberg]

"Online I see some stuff in python2 and some stuff in python3, does it matter which one I use?"

YES

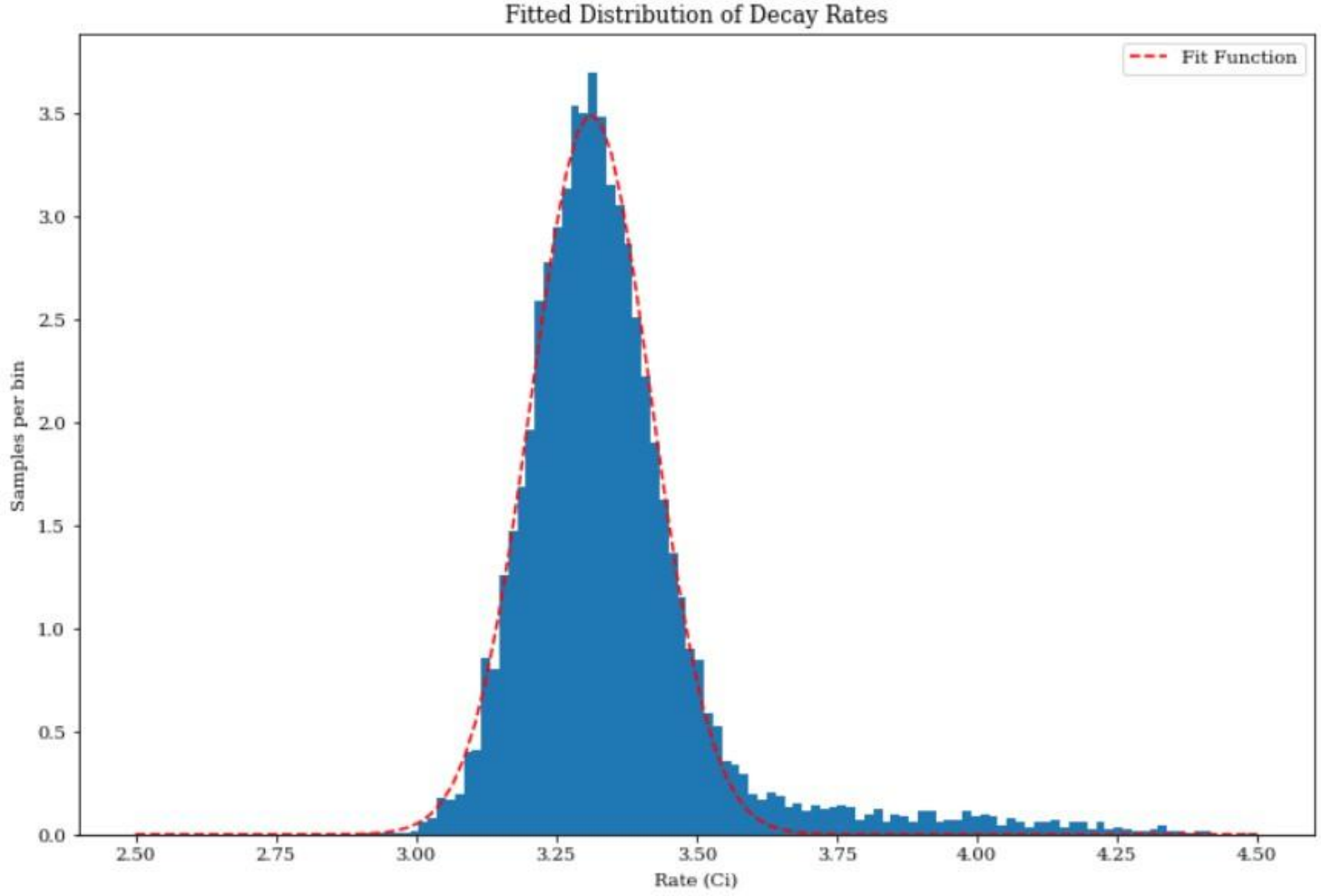
These two versions are mildly incompatible:

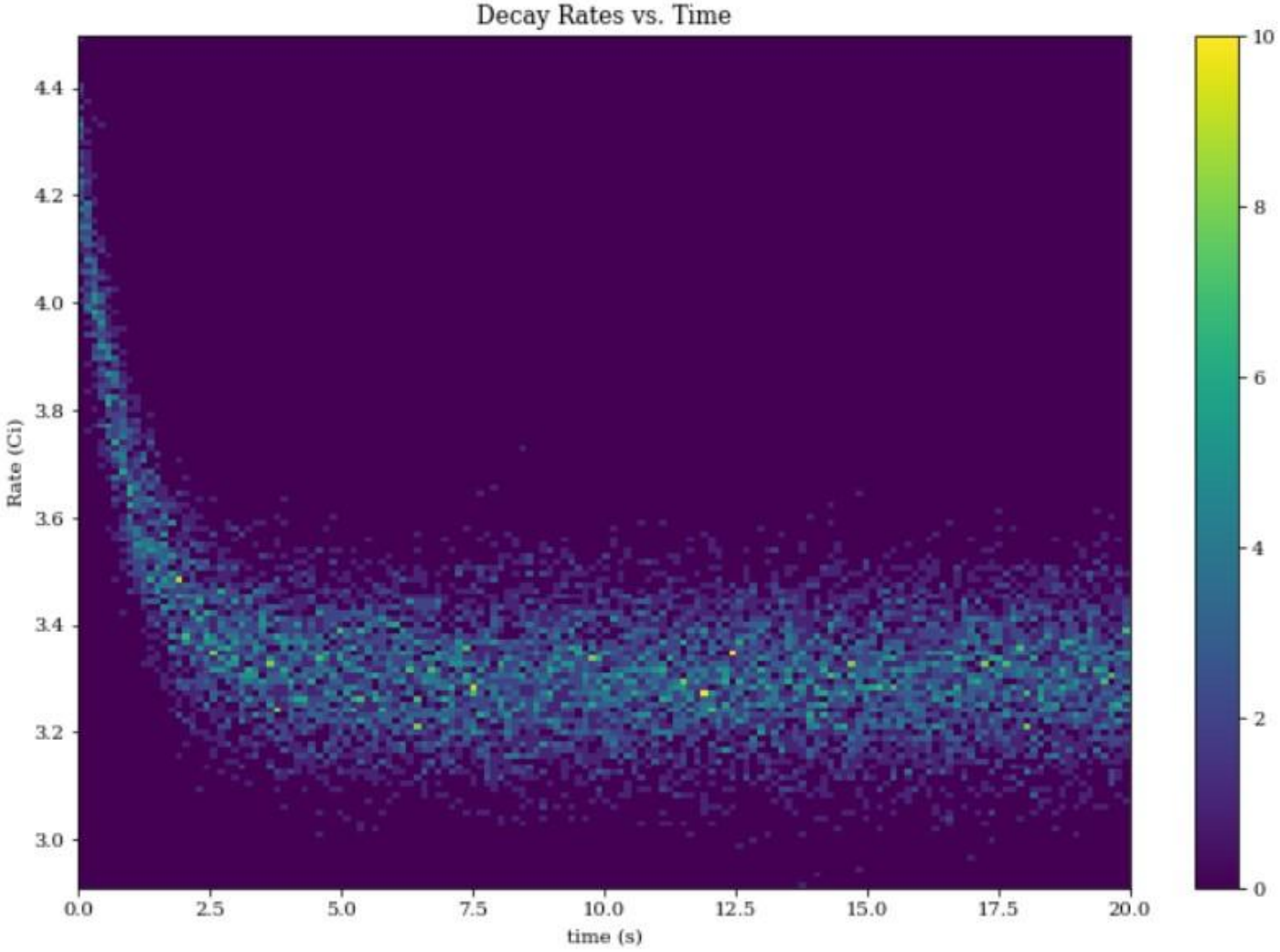
python2	python3
print a , print "hello world"	print(a), print("hello world")
7/2 = 3 (integer division)	7/2 = 3.5 (floating point division) 7//2 = 3 (integer division)

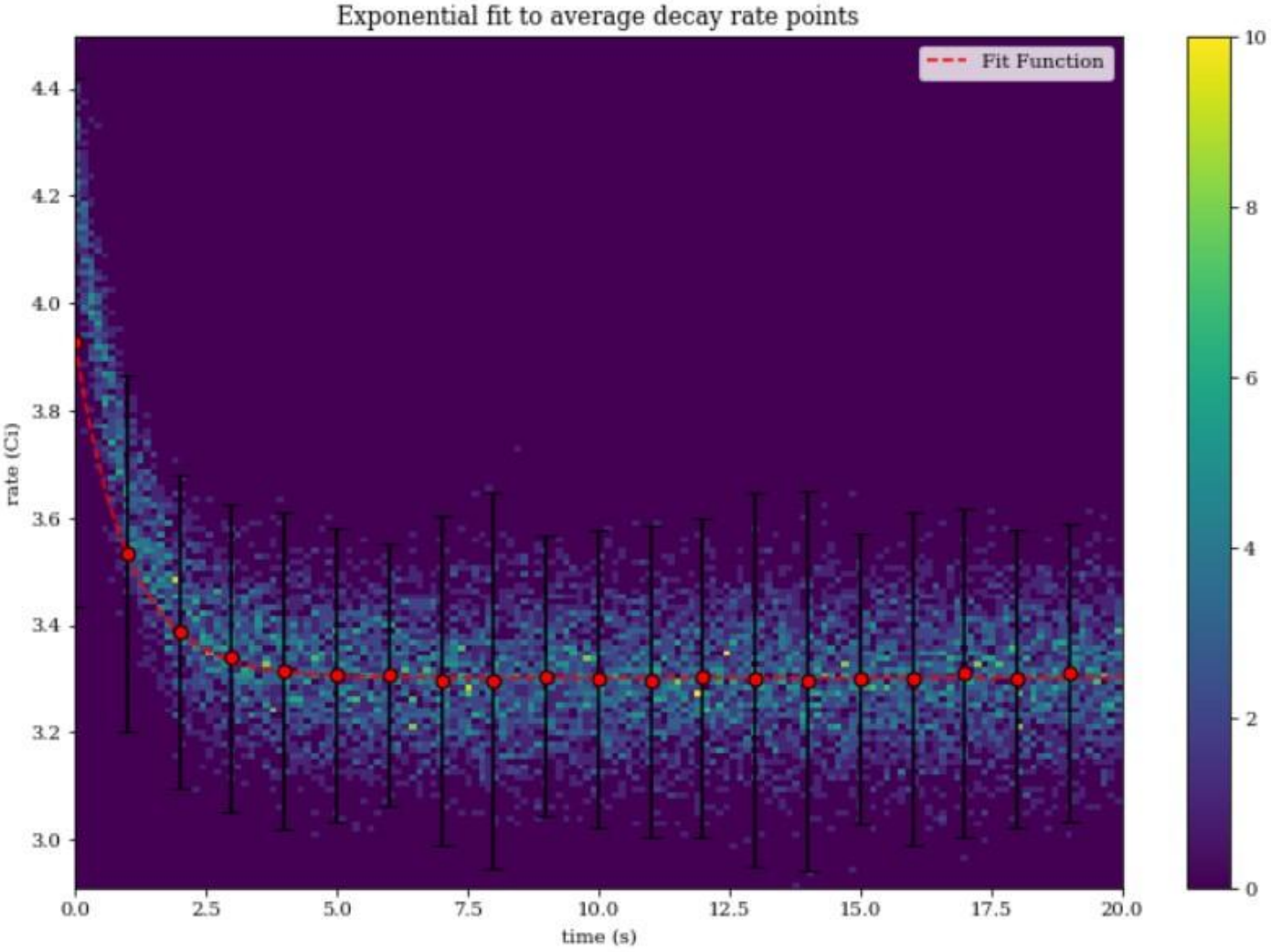
Rule #1: We always use python3

Rule#2: We never ever ever ever use python2 (unless you're using someone's old code that they rudely did not update for you, but it's okay because at least they optimized it and it runs really fast)

You can check which version you have by typing `$ python -V` into the command line


Fitted Distribution of Decay Rates
Fit Function
Samples per bin
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
2.50
2.75
3.00
3.25
3.50
3.75
4.00
4.25
4.50
Rate (Ci)


Decay Rates vs. Time
Rate (Ci)
4.4
4.2
4.0
3.8
3.6
3.4
3.2
3.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
time (s)
10
8
6
4
2
0


Exponential fit to average decay rate points
Fit Function
rate (Ci)
time (s)
4.4
4.2
4.0
3.8
3.6
3.4
3.2
3.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
10
8
6
4
2
0