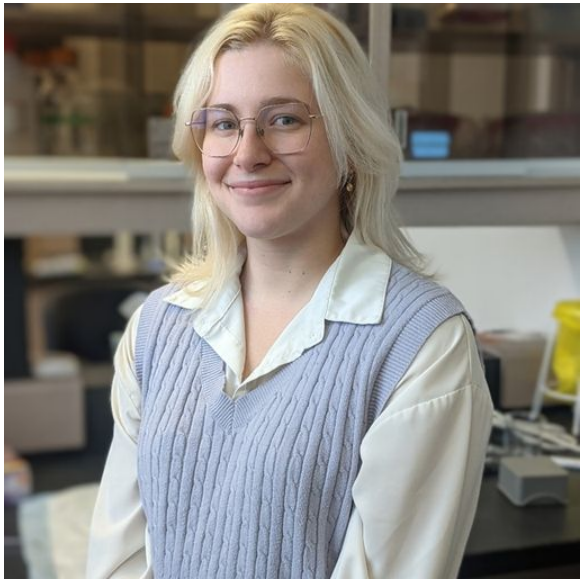
NeuroPy week 1: Introduction to coding

CENTRE DE RECHERCHE CERVO BRAIN RESEARCH CENTRE

Sandrine Poulin & Antoine Daigle

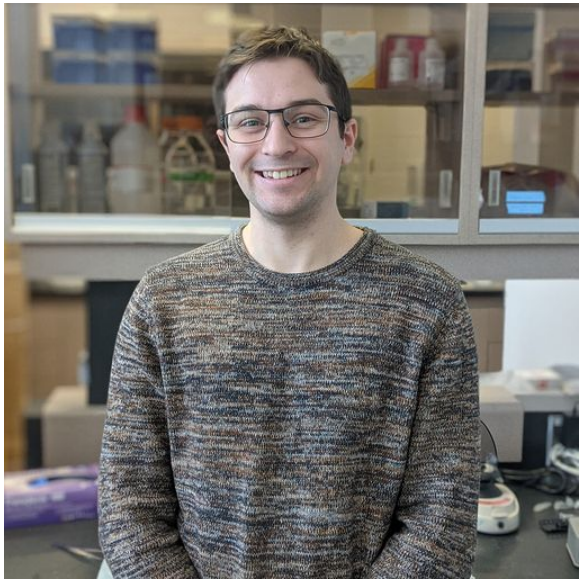
July 9th 2024

Short introduction



Sandrine Poulin - B.Eng

PDK Lab & VBP Lab



Antoine Daigle - B.Eng

VBP Lab & MD Lab

Plan of this workshop

Part 1 (~45 minutes): Theory

1. Setting up your Python environment
   Installation, IDE and Jupyter Notebook
2. Syntax structure
   Variables, Int, Float, Tuple, String, List, Dictionary
3. Control structures
   if, elif, else, for loop
4. Function
   Syntax and documentation

Part 2 (~45 minutes): Examples

Slides:

Google colab:

ChatGPT warning

For this course, **do not use ChatGPT**.

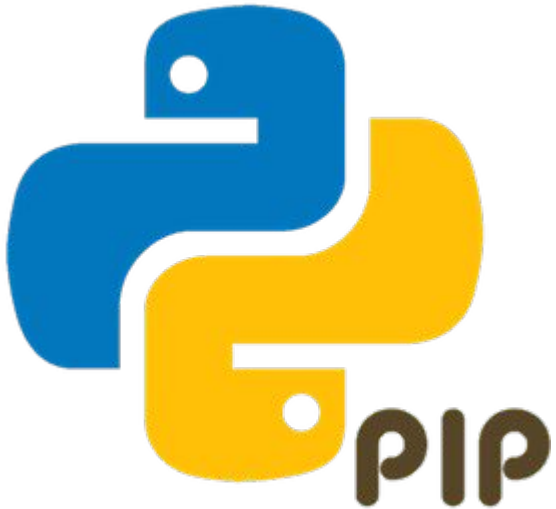
While ChatGPT is a very useful tool when coding, an underlying goal of the course is to **make you think like a programmer**. You can use online references, but no generative AI, since it's doing the thinking for you.

Part 1

1. **Setting up your Python environment**
   - **Installation, IDE and Jupyter Notebook**
2. Syntax structure
   - Variables, Int, Float, Tuple, String, List, Dictionary
3. Control structures
   - if, elif, else, for loop
4. Function
   - Syntax and documentation

1 Python installation

- There is different way to install Python on your computer
  - On Windows, by the Microsoft Store or web site - Download Python
  - By using Anaconda - Free Download | Anaconda (Recommended)



1 IDE installation

- You need an interface to code, there's two recommended choice:
  - VSCode - Setting up Visual Studio Code
  - Spyder - Comes with Anaconda

- We recommend VSCode
  - Git integration
  - Possibility to open other file types
  - Don't forget to add the Python extension in VSCode

1 Jupyter Notebook

- Jupyter Notebook is like a code interpreter that is widely used to visualize data. Jupyter supports many coding languages, such as Python.
  - Installation
    - Jupyter Notebook comes with Anaconda, if you do not have anaconda, it is fairly simple to install jupyter notebook.
    - From terminal: pip install jupyter notebook
  - Launch:
    - From terminal: jupyter notebook

- Google Colab:
  - Web hosted Jupyter Notebook service
  - Require no setup to use

Part 1

1. Setting up your Python environment
   - Installation, IDE and Jupyter Notebook
2. **Syntax structure**
   - **Variables, Int, Float, Tuple, String, List, Dictionary**
3. Control structures
   - if, elif, else, for loop
4. Function
   - Syntax and documentation

2 Variables

- Assign to a **name** a **value** or an **expression**.
- The **print function** allows us to see the result of the equation.
  - In this example, we evaluate the perimeter of a circle (p = 2πr) of a radius of 5.

```
pi = 3.1416
perimeter = 2 * pi * 5
print(perimeter)

>>>> 31.416
```

2 How to add comments to the code

- Add comments by using the “#” character
  - You can also comment and uncomment a block of code by selecting it and press a combination unique to your system (Mac, Linux, Windows).

```
pi = 3.1416 # Variable pi
perimeter = 2 * pi * 5  # Evaluation of the perimeter
# Let's print the result
print(perimeter)

>>>> 31.416
```

2 Naming variables

What is the syntax of the identifier you can use?

- a-z
- A-Z
- 0-9 (but not in first position)
- underline character “_”

It is conventional to write in lower case with underscores for better clarity.

```
pi = 3.1416
radius circle 5 = 5
perimeter = 2 * pi * radius_circle_5
```

2 print function

One of the most useful function in *Python*.

```
print(2, pi, 5)
print()
print('perimeter =', 2*pi*5)

>>>> 2 3.1416 5
>>>>
>>>>perimeter = 31.416
```

2 Numbers

There are 3 categories of numbers:

- Integers (**int**): …, -2, -1, 0, 1, 2, …
- Floating point number (**float**): 32.90871, …
- Complex numbers (complex)

```
a = 3.86
print(a, type(a))
b = int(a)
print(b, type(b))

>>>> 3.86 <class 'float'>
>>>>3 <class 'int'>
```

2 Arithmetic operators

Use those operators to manipulate numbers.

- +, -, *, / (addition, subtraction, multiplication and regular division)
- ** (exponential)
- //, % (floor division, modulus)

```
print("7/4 =", 7/4)
print("7//4 = ", 7//4)
print("7%4 =", 7%4)

>>>> 7/4 = 1.75
>>>> 7//4 =  1
>>>> 7%4 = 3
```

2 Strings

- String (str) is a type that allows us to manipulate text.
- You can use some operators on this type:
  - “+”: Add two or more strings together
  - “*”: Multiply the string sequence
  - “len()”: Determine the length of the string

```
x = "Bonjour l'monde"
print(x)
>>>> Bonjour l’monde
```

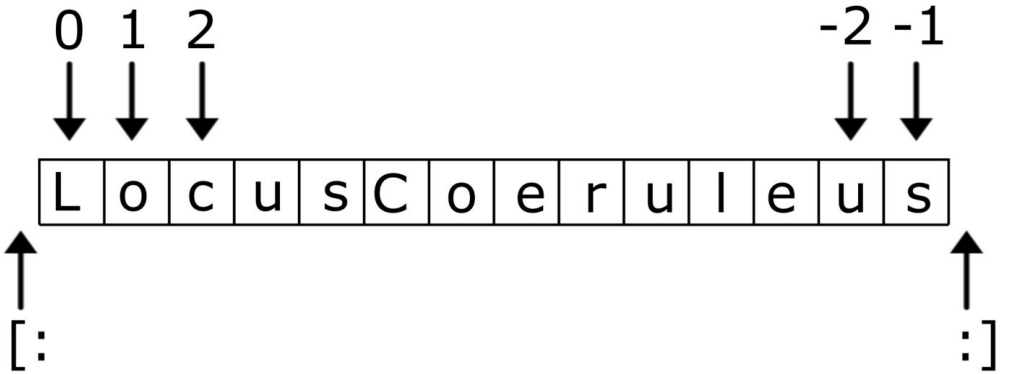
```
print(10 * "-")
print("-" * 3 + " Mouse ", "-" * 3)
print(len("Mouse"))
>>>> ----------
>>>> --- mouse  ---
>>>> 5
```

2 Index and slicing

By using the “[ ]” operator, we can index and slice a string (or a list).

- **[i]**: Get one element at this position “i”.
- **[i : j]**: Get all element between “i” and “j”.
- **[i : j : k]**: Between “i” and “j”, but jumping of “k” indexes each time.

```
a = "LocusCoeruleus"
print(a[1])
print(a[3:8])
print(a[-2:])
>>>> o
>>>> usCoe
>>>> us
```



2 The list

Can stock object, data, … The notation of a list is the “[ ]” and the elements are separated by a comma. **Careful**, “[ ]” is used for list and indexing/slicing.

- You can index and slice a list.

```
a = [] # This is an empty list
b = [2, 3, "cervo", 5] # This is another list
print(a, b)
print(b[2])

>>>> [] [2, 3, 'cervo', 5]
>>>> cervo
```

2 Some function of the list

Multiple functions can be applied to the list, such as

- list.append()
- len(list)

```
a = []
a.append("Mouse")
print(a)

>>>> [“Mouse”]
```

```
b = [2, 3, "cervo", 5]
print(len(b))

>>>> 4
```

2 Dictionary

Another way to encapsulate data is with a dictionary. There are **two** components to every **element of a dictionary**. The *key* and the *value*. Basically, you call the key to get the value.

```
my_dic = {"KEY": "value"}
print(my_dic)

my dic["neuron_type"] = ["VIP", "NDNF"]
print(my_dic)

>>>> {“KEY”: “value”}
>>>> {“KEY”: “value”, “neuron_type”: [“VIP”, “NDNF”]}
```

Some python modules (FaceMap, Suite2p, …) will give you their output in a dictionary.

Part 1

1. Setting up your Python environment
   - Installation, IDE and Jupyter Notebook
2. Syntax structure
   - Variables, Int, Float, Tuple, String, List, Dictionary
3. **Control structures**
   - **if, elif, else, for loop**
4. Function
   - Syntax and documentation

3 Conditional statement

There are three statements that allows us to implement conditions in the code.

- “if”: If the condition is True, execute this block of code. (required)
- “elif”: If the condition is True, execute this block of code. (optional)
- “else”:If no conditions is True, execute this block of code. (optional)

```
if expression 1:
    # Statement block 1

elif expression 2:
    # Statement block 2

elif expression n:
    # Statement block n

else:
    # Statement block n+1
```

3 For loop

- “Used to iterate over an iterable.” ~ Marc Parizeau
- When **you want to go over elements** in a list, tuple, string, …

```
for target in iterable:
    # Do something with the target
```

```
names = ["Antoine", "Sandrine"]

for name in names:
    print(f"I appreciate {name}'s efforts to teach me Python.")

>>>> I appreciate Antoine's efforts to teach me Python.
>>>> I appreciate Sandrine's efforts to teach me Python.
```

3 break, pass and continue

- **break:** This statement breaks out of the innermost enclosing of the loop.
- **pass:** Used as a placeholder: nothing happens and the rest of the loop is executed as usual.
- **continue:** Continue to the next iteration of the loop

```
for num in range(10):
    print(num)
    if num == 2:
        break

>>>> 0
>>>> 1
>>>> 2
```

```
for num in range(2, 5):
    if num % 2 == 0:
        print("Found an even number", num)
        continue
    print("Found an odd number", num)

>>>> Found an even number 2
>>>> Found an odd number 3
>>>> Found an even number 4
```

Part 1

1. Setting up your Python environment
   - Installation, IDE and Jupyter Notebook
2. Syntax structure
   - Variables, Int, Float, Tuple, String, List, Dictionary
3. Control structures
   - if, elif, else, for loop
4. **Function**
   - **Syntax and documentation**

4 The function

The function is one of the most important concepts in python.It lets the user reuse their code to perform a specific task, making the code more organized and easier to manage.


```
def name(arg1, arg2, ... , argn):
    # Indented block with the code
    return expression # Facultative
```

4 A simple function

```
def addition(a, b):
    return a + b

print(addition(3, 5))

>>>> 8
```

4 A more complex function

```
def find_maximum(list_to_analyse):
  maximum = list_to_analyse[0] # Assigning the maximum value
to the first element of the list
  for element in list_to_analyse:
    if element > maximum:
      maximum = element # Replacing the value of maximum if
element is   bigger than the actual value
  return maximum

example_list = [9,13,56,73,24,5,3,-91,53,0]
print(find_maximum(example_list))

>>>> 73
```

Final note: Built-in functions

Python itself has a number of functions and types built into it that are always available.

- range()
- enumerate()
- int(), float(), …
- abs()
- max(), min()
- print()
- …

Those functions are great, but so much more can be done with modules!

Built-in Functions

A	E	L	R
abs()	enumerate()	len()	range()
aiter()	eval()	list()	repr()
all()	exec()	locals()	reversed()
anext()			round()
any()	F	M	
ascii()	filter()	map()	S
	float()	max()	set()
B	format()	memoryview()	setattr()
bin()	frozenset()	min()	slice()
bool()			sorted()
breakpoint()	G	N	staticmethod()
bytearray()	getattr()	next()	str()
bytes()	globals()		sum()
		O	super()
C	H	object()	
callable()	hasattr()	oct()	T
chr()	hash()	open()	tuple()
classmethod()	help()	ord()	type()
compile()	hex()		
complex()		P	V
	I	pow()	vars()
D	id()	print()	
delattr()	input()	property()	Z
dict()	int()		zip()
dir()	isinstance()		
divmod()	issubclass()		_
	iter()		__import__()

Part 2: Exercices

Slides:

Google colab:

1 Strings, index

Using the phrase:

"The;zebrafish;model;is;valid;and;great"

1. Replace the ; with spaces
2. Isolate the word zebrafish

Notes:

In python, there is a function to replace a character in a string with another one

```
yourstring.replace(character_to_replace,character_to_replace_with)
```

2 Manipulating a dictionary

- Create an empty dictionary named “country_capital”.
  - Add 4 countries and their capitals as key-value pairs.
  - Check if the country “France” is in the dictionary.
  - Print all the keys of the dictionary.
  - Print all the values of the dictionary.

3 Exercise: Conditional statements

Given a number (x):

if x is between 1 and 50, we want to print:

“The number is between 1 and 50”

if x is between 51 and 100, we want to print:

“The number is between 51 and 100”

in other cases (x is not between 1 and 100), we want to print:

“The number is not between 1 and 100”

4 Recap exercise

You have this list of dna sequences:

```
# Input
dna_sequences = ["ATCGA", "TTAAGC", "CGATG"]

# Output
# GC content for each DNA sequence respectively
# [40.0, 33.33333333333333, 60.0]
```

Create a function that calculates the ratio of guanine and cytosine in the DNA. The function must take a string as argument and return a float.