jungle program

Introduction to Python Programming

Thursday, June 17 at 6pm

Negar Vahid

Founder/CTO at ElaraTech

Agenda

1- Data Types

2- Lists

3- Loops

4- Functions

First let’s go over the very basics of Python.
When do we use Python and what code editors are the best for data science?

Python

- High-level Programming Language
- Focuses on code readability

When Python?

- You want to do some quick calculations
- For your new business, you want to develop a database-driven website.
- Your manager asks you to clean and analyze the results of the latest satisfaction survey.

Why Python for data science?

- Easy to pick-up
- Readable syntax
- Easy data reading in CSV and Json
- Fast and easy to use data science libraries such as numpy and pandas.

Which code-editor should I install?

VS-code

Pycharm

Anaconda

Jupyter

If you don’t have a code editor installed, no worries!
You can use an online code editor to code along

Basic Calculation

Variables

Data Types

Cool stuff that we have in Python

It's possible to convert a float to an integer and vice versa. To convert an integer to a float, we can use the `float()` command. To convert a float to an integer, we can use the `int()` command:

`int` → `float`

script.py

```
float(10)
```

```
Output
10.0
```

`float` → `int`

script.py

```
int(4.3)
```

```
Output
4
```

Notice the `int()` command rounded `4.3` down to `4`. `int()` will always round a `float` down, even if the number after the decimal point is greater than five.

Lists

1- What even is a list?

A *list* is an ordered set of objects in Python.

Grades example

We want to store the grades of a couple of students:

```
Lasha’s grade is 90
Giorgi’s grade is 95
Nene’s grade is 100
Tamar’s grade is 100
```

```
grades = [90,95,100,100]
```

```
grades = [90,95,100,100]
```

1 - A list begins and ends with square brackets ([and]).

2 - Each item (i.e., 95 or 100) is separated by a comma (,)

3 - It’s considered good practice to insert a space () after each comma, but your code will run just fine if you forget the space.

What happens if we run this line of code?

```
broken_grades = [78 60 100 67]
```

Can lists contain more than just numbers?

Yes!

Our lists can contain *strings* :

```
names = ['Lasha', 'Giorgi', 'Nene', 'Tamar']
```

And even a mix of stuff:

```
mixed_stuff = ['Lasha', 90]
print(type(mixed_stuff))
```

```
<class 'list'>
```

Wait can I have ... a list of lists?!

YES. We previously created a list like this :

```
lasha = ['Lasha', 90]
```

We can put several of these lists into one list, such that each entry in the list represents a *student's name* and *their grade*:

```
names_and_grades = [['Lasha', 90], ['Giorgi', 95], ['Nene', 100], ['Tamar', 100]]
```

Task

Create a `list of lists` called `ages` where each sublist contains a student's name and their age. Use the following data:

- '`Felix`' is `19`
- '`Aaron`' is `16`

Google trends

Here are the top **Query search trends of England in the past month**

1	The Grand National - Topic	+1,950%
2	Prince Philip, Duke of Edinburgh - Topic	+550%
3	Pub - Topic	+160%
4	Line of Duty - British television series	+130%
5	Gym - Topic	+90%

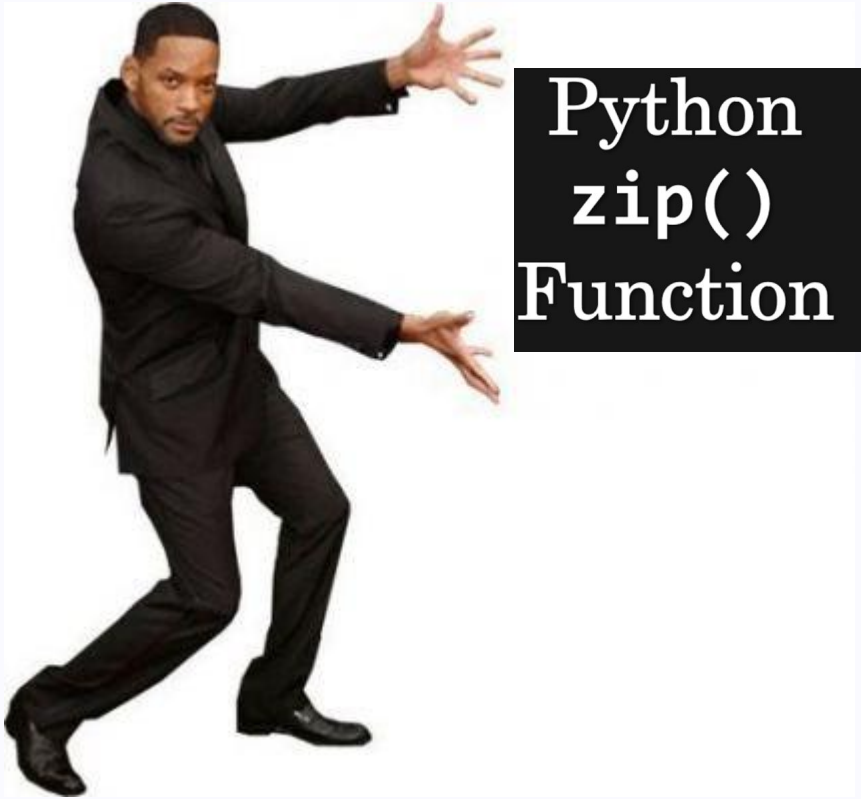
We have our data like this :

```
search_title = ['The Grand National', 'Prince Philip', 'Pub', 'Line Of Duty', 'Gym']

search_increase = [1950, 550, 160, 130, 90]
```

How do we combine these two lists?!(like a list of lists below, but automatically)

```
search_queries = [['The Grand National', 1950], ['Prince Philip', 550]]
```



Python `zip()` Function

2. Zip()

If we wanted to create a list of lists that paired each search title with its search increase, we could use the command `zip`. `zip` takes two lists as inputs and returns an object that contains a list of pairs. Each *pair* contains one element from each of the inputs.

Let's go explore zip() in VS Code

Task

Copy the content of search_trend from the chat!

1 - Use zip( ) to combine search_titles and search_increase .

2 - Print out the pairs you've made.

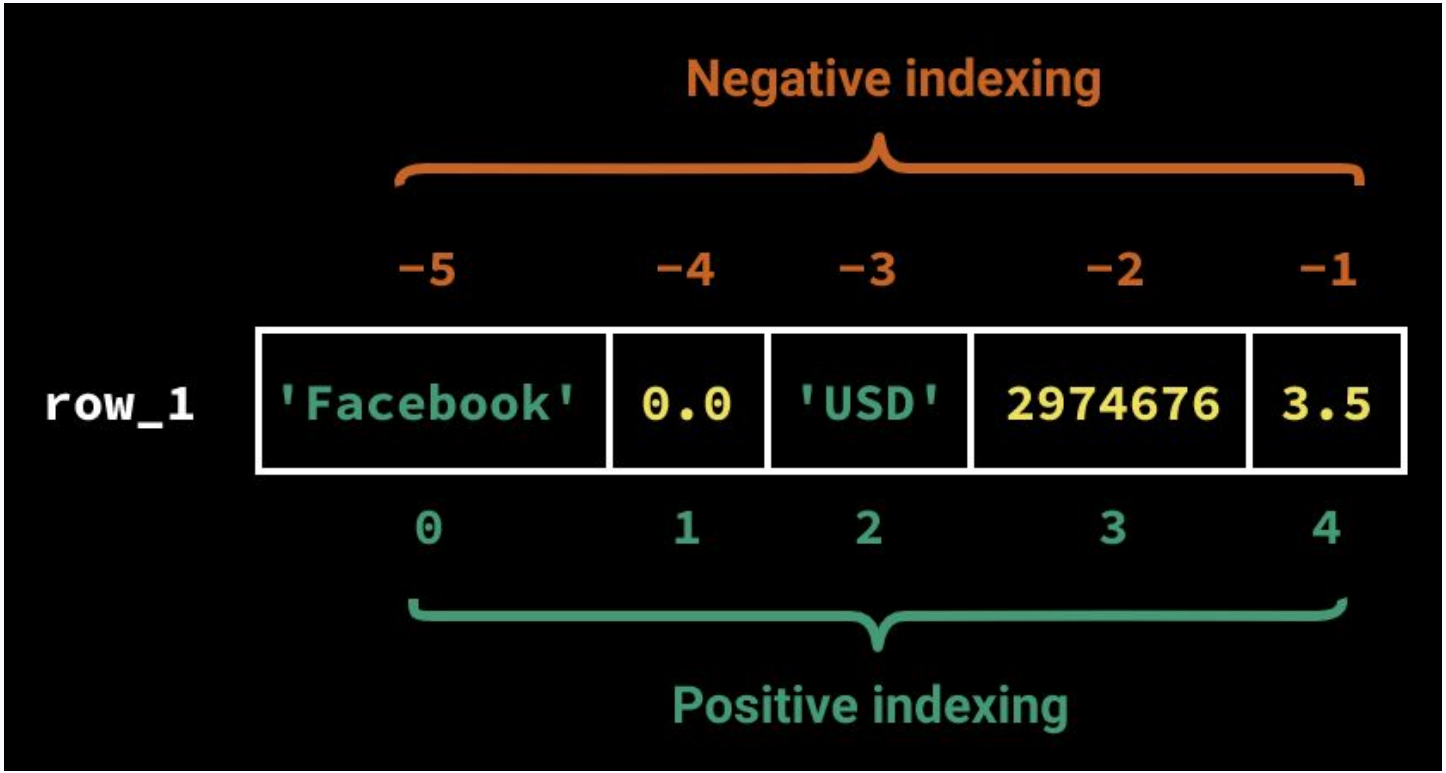
List Indices

Python list elements are ordered by *index*, a number referring to their placement in the list. List indices start at 0 and increment by one.

To access a list element by index, square bracket notation is used: `list[index]`.

```
row_1 = ['Facebook', 0.0, 'USD', 2974676, 3.5]

row_1[0] # 'Facebook'
row_1[4] # 3.5
```


Negative indexing
-5
-4
-3
-2
-1
row_1
'Facebook'
0.0
'USD'
2974676
3.5
0
1
2
3
4
Positive indexing

Determining List Length with `len()`

The Python `len()` function can be used to determine the number of items found in the list it accepts as an argument.

List Method `.count()`

The `.count()` Python list method searches a list for whatever search term it receives as an argument, then returns the number of matching entries found.

List Method `.sort()`

The `.sort()` Python list method will sort the contents of whatever list it is called on. Numerical lists will be sorted in ascending order, and lists of Strings will be sorted into alphabetical order. It modifies the original list, and has no return value.

```
exampleList = [4, 2, 1, 3]
exampleList.sort()
print(exampleList)
# Output: [1, 2, 3, 4]
```

Introduction to loops

If we wanted to print out each name in our names list:

```
names = ["Davit", "Negar", "Maia", "Demetre", "Sandro", "Nata"]

print(names[0])
print(names[1])
print(names[2])
print(names[3])
print(names[4])
print(names[5])
```

This seems inefficient. Luckily, Python (and most other programming languages) gives us an easier way of using, or iterating through, every item in a list. We can use loops!

A loop is a way of repeating a set of code many times.

```
for name in names:
    print(name)
```

Let’s print out every search_title with a for loop!

Functions

1. What even is a function?

A function is a collection of several lines of code. By calling a function, we can call all of these lines of code at once, without having to repeat ourselves.

Let’s write some code

We want to create a program that greets customers every time they enter our restaurant :

```
print("welcome to our store!")
print("make sure to have your mask on in
all times!")
```

Imagine writing two lines of code per person when 8 customers enter the store together…that's 16 repetitive lines of code! Annoying.

What we can do, is to create a **function**, so every time a customer enters, we call the function and it does the job for us!

We'll call it -> `greet_customer()`

```
def greet_customer():
    print("welcome to our store!")
    print("make sure to have your mask on in all times!")

greet_customer()
```

We have isolated this behavior from the rest of our code. Once we determine that `greet_customer()` works the way we want, we can reuse it anywhere and be confident that it greets, without having to look at the implementation.

Let's go run this block of code is Pycharm

Task

1 - Write a function called `loading screen` that prints "`This page is loading...`" to the console.

2 - Outside of the function body, call `loading_screen()`.

```
def function_name():
    print("something")

function_name()
```

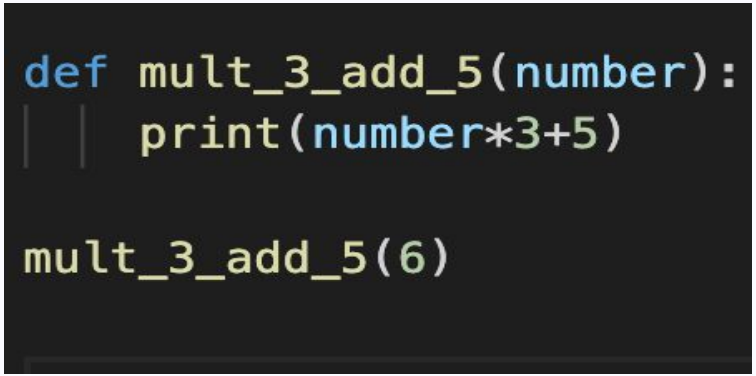
2. Parameters

Task

1 - Create a function called mult_x_add_y().

2 - The function takes values of x,y, and number.

3 - The function should print(x*number+y).



3. Return

Okay… till now we learned how to print stuff with a function… cool…
But what if we wanted to store the result of a function in variable?

We can do this with “`return`”
So basically our function can “return” a certain thing. It can be a number, a string, boolean, you name it!

Example

Let’s create a hello_dog function!
It returns “hello” + the name of a dog that the user has entered .

Create a function which takes a string, and reverses it!

For example : Hi there -> ereht iH

Task

We’re gonna create a function that takes the level of awesomeness (0-100), and return the difference of that level and 100, so that person knows how many levels they have left to become 100% awesome!

1 - Create a function called `awesomness_level`

2 - The function takes `level` as an input

3 - Return (100 - level)

4 - Store the output of our `aweosomness_level` function in a variable called `awesome_count`

5 - print out `awesome_count`!

Python Essentials For Data Science

Next training: from August 02 to August 27

Join a 4-week long training to practice all the essentials you need to start your data science journey.

We'll go through all the Python you'll need and get a taste of data science at the same time.

Enroll & Start Learning

Learn Python Programming

👋 Who is this for?

This program is for data enthusiasts who want to gain strong foundations in data science, but who need an introduction, or a refresher in, Python programming or core data science concepts. **No prior coding experience is required**

Week 1

Python Basics: Control Flow and Lists

Hello World! This chapter teaches you how to control the flow of your program and comfortably use python lists.

Week 2

Loops And Functions

We'll go through two of the most important programming concepts: loops and functions! Learn how to reuse your code.

Week 3

Strings And Files

Handling Our Data: Learn how to work with files in an automated way, how to manipulate strings, and how we to organize our data.

Week 4

Final project

Contact

Gautier Roquancourt

Co-founder at Jungle Program

- Phone: +33 6 38 22 42 16
- Email: gautier@jungleprogram.com
- Website: jungleprogram.com

jungle program

We are doneeeeeee
Any questions?


jungle
program