Natural Language Processing NLP_CLT_1st_APR_27st_2025

Eng. Maytham Ghanoum

Artificial Intelligence & Deep Learning Specialist MTN Syria – SCS – SVU CLT +963947222064 - +963982018359 https://www.linkedin.com/in/maytham-ghanoum-69

https://www.facebook.com/maytham.ghanoum







the state

Python is a high-level, general-purpose programming language that has become one of the most popular and versatile languages in the world. Known for its simplicity, readability, and vast ecosystem of libraries, Python is widely used across various domains, including web development, data science, artificial intelligence, machine learning, and, most notably, bioinformatics.



Features of Python: Readability and Simplicity:

Python's clean and straightforward syntax is designed to be easy to read and write, making it an ideal language for beginners and experienced developers alike.

Interpreted Language:

Python is an interpreted language, meaning that code is executed line-by-line, which facilitates rapid prototyping, testing, and debugging.

Cross-Platform Compatibility:

Python is compatible with various operating systems, including Windows, macOS, and Linux, allowing developers to write code that runs seamlessly across different platforms. **Extensive Library Support:**

One of Python's greatest strengths is its extensive collection of libraries and frameworks. These libraries simplify complex tasks such as data analysis (e.g., NumPy, pandas), scientific computing (e.g., SciPy), and Deep learning (e.g., TensorFlow, PyTorch).

O DOGOL



Python Programming Languages: Preparing of Python Environment:

The Best Environment for Python called Anaconda 1- Go to https://www.anaconda.com/download



Do Dolla

2- Press on Free Download3- The environment will start downloading, when finish you will have a file look like this:

vocoueoseroetup-x04-1.95.1

O Anaconda3-2024.06-1-Windows-x86_64



Anaconda3 2024.06-1 (64-bit) Setup

NACOND

Welcome to Anaconda3 2024.06-1 (64-bit) Setup

 \times

O BOOL

Setup will guide you through the installation of Anaconda3 2024.06-1 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.





Cancel

Next >





License Agreement

X

00000

ANACONDA.

Please review the license terms before installing Anaconda3 2024.06-1 (64-bit).

Press Page Down to see the rest of the agreement.

ANACONDA TERMS OF SERVICE

Please read these Terms of Service carefully before purchasing, using, accessing, or downloading any Anaconda Offerings (the "Offerings"). These Anaconda Terms of Service ("TOS") are between Anaconda, Inc. ("Anaconda") and you ("You"), the individual or entity acquiring and/or providing access to the Offerings. These TOS govern Your access, download, installation, or use of the Anaconda Offerings, which are provided to You in combination with the terms set forth in the applicable Offering Description, and are hereby incorporated into these TOS. Except where indicated otherwise, references to "You" shall include Your Users. You hereby acknowledge that these TOS are binding, and You affirm and signify your consent to these TOS by registering to, using, installing, downloading, or accessing the Anaconda Offerings

If you accept the terms of the agreement, dick I Agree to continue. You must accept the agreement to install Anaconda3 2024.06-1 (64-bit).

Anaconda, Inc.

< Back I

I Agree

Cancel





Anaconda3 2024.06-1 (6)	i4-bit) Setup		×	
O ANACONDA.	Choose Install Location Choose the folder in which to install An (64-bit).	naconda3 2024.06-:	1	
Setup will install Anaconda: folder, click Browse and se	3 2024.06-1 (64-bit) in the following folder lect another folder. Click Next to continue.	r. To install in a diff 2.	erent	
Destination Folder C:\Users\\anaconda3\	New folder\	Browse		
Space required: 5.0 GB Space available: 350.3 GB Anaconda, Inc.				
	< Back N	lext > Ca	ancel	



1- From start menu open and type Jupyter.



2- The browser now will open automatically

💭 jupyter			
File View Settings Help			
E Files O Running			
Select items to perform actions on them.			✓ New
■ / ■ Name		▲	La Concolo
🔳 🖿 3D Objects			^{2 r} ≡ New File
anaconda3			^{6 n} 🗈 New Folder
Cisco Packet Tracer 8.2.1			last month
22 ² 12 ² 2	Number of the second se		1111 1111 1111 1111







3- Jupyter interface:

JUPYTET Untitled31 Last Checkpoint: 8 seconds ago	ę	
File Edit View Run Kernel Settings Help	Trusted	
🖻 + 🛠 🗖 🗂 ▶ ■ ♂ ≫ Code 🗸	JupyterLab 🗂 🗯 Python 3 (ipykernel) 🔿	
[]:	□ ↑ ↓ 古 早 章	ARANGO DO CONTRACTOR

4- If your PC or Laptop is old and with low resources or you don't have Laptop, don't worry you can work with Google Colab: https://colab.research.google.com/

It's a free platform from google that allow you to run any Python codes and you also can create and manage a full project in it.







Python Programming Languages: Preparing of Python Environment: The Best Environment for Python called Anaconda During this course, we will focus our work on Google Colab

colab.res	earch.google.com					Google Lens	@ ☆)
_								
Welcome To	Colab						<u>ي</u> فع	\$
ile Edit Viet	Open notebo	ook						
tting started	Examples	>	Search notebooks		م	G	Ū	
ta science achine learnin <u>c</u>	Recent	>	Title	Last opened 🔸	First opened \uparrow			
ore Resources Featured exa	Google Drive	>	A Untitled65.ipynb	November 27	November 27		Z	
Section	GitHub	>	CO Welcome To Colab	November 27	November 18	C	Z	
			4 <u>Untitled64.ipynb</u>	November 27	November 23	a (2	
	+ New notebook		Use System Instructions	un chat			Cancel	





Preparing of Python Environment:

The Best Environment for Python called Anaconda

4- If your PC or Laptop is old and with low resources or you don't have Laptop, don't worry you can work with Google Colab:

Velcome To	Colab						<u> </u>	
e of content	Open noteb	ook						
tting started ta science	Examples	>	Search notebooks		٩	G	Ū	
achine learning	Recent	>	Title	Last opened ↓	First opened 🔨			
ore Resources Featured exa	Google Drive	>	A Untitled65.ipynb	November 27	November 27		نة 12	
Section	GitHub	>	CO Welcome To Colab	November 27	November 18		ß	
	Upload	>	A Untitled64.ipynb	November 27	November 23	2	ß	
	+ New notebool	k					Cancel	
			Use System instructions in chat					
			Autorite and		**********			
Learning Specialist				Hiller	aaaaaa			

Lets dive into Python, from Scratch!

Variables

Think of a **variable** as a container that stores information. Just like you might put a name label on a box to remember what's inside, a variable helps your program remember a value.

•Example:

If you want to store your age, you can use a variable like age = 25. In Python, creating a variable is as simple as giving it a name and assigning it a value using the = symbol.





Lets dive into Python, from Scratch!

Rules for Variable Names:

1 - **Must** start with a letter (a-z, A-Z) or an underscore (_) Example: name , _count

2- **Cannot** start with a number. Name : Correct, 1name: Wrong.

3- No spaces or special characters:@#\$%: don't use , firstname: correct, first_name: correct, first name: Wrong



4- Variables are **case-sensitive**: Name is different from name. A is totally different from a.



Lets dive into Python, from Scratch!

Numbers in Python

1- integers: 1, -5, +800, ...

2- float: 1.5 , -0.8, -3.1486458..., +18.26...

Basic Operations with Numbers:

Operation	Symbol	Example	Result
Addition	+	5 + 3	8
Subtraction	-	10 - 4	6
Multiplication	*	6 * 7	42
Division	/	15 / 3	5.0
Modulus (remainder)	%	10 % 3	1
Exponent (power)	**	2 ** 3	8





Lets dive into Python, from Scratch!

Strings in Python:

String is a sequence of characters—letters, numbers, symbols—enclosed in quotation marks.

- Single quotes: 'hello' or Double quotes: "hello" are the same in Python.
- To handle a string consist of multiple sentences and lines we use triple quotes.

"" "Welcome to Python programming language. It widely used in Artificial Intelligence and Bioinformatics" ""

String Operations:				
Operation	Example	Result		
Concatenation (joining)	'Hello' + ' World'	'Hello World'		
Repetition	'Hi' * 3	'HiHiHi'		
Length	<pre>len('Python')</pre>	6		

............



Lets dive into Python, from Scratch!

Strings in Python:

String is a sequence of characters—letters, numbers, symbols—enclosed in quotation marks.

Accessing Characters in a String:
You can access individual characters in a string using indexing (position of characters).

Example: word = ''Python'' first_letter = word[0] # 'P' (index starts at 0) last_letter = word[-1] # 'n' (negative index starts from the end)





Lets dive into Python, from Scratch!

Booleans in Python: Boolean represents one of two values: True , False

Boolean Operations:

Operation	Symbol	Example	Result
Equal to	==	5 == 5	True
Not equal to	! =	5 != 3	True
Greater than	>	7 > 3	True
Less than	<	2 < 5	True
Greater than or equal to	>=	6 >= 6	True
Less than or equal to	<=	4 <= 8	True







Lets dive into Python, from Scratch!

Now that we've covered the basics of variables, numbers, strings, and Booleans, it's time to introduce one of the most important and versatile data structures in Python: Lists.

Think of a **list** as a **collection of items**. Just like a shopping list holds multiple items, a Python list can hold multiple values—numbers, strings, or even other lists!





Lets dive into Python, from Scratch!

What is a List?

A **list** is a collection of items (or elements) that are ordered and changeable (mutable). Lists are defined using **square brackets** [], and the elements inside the list are separated by commas. Example: Fruits = ['apple', 'banana', 'cherry']

Here Fruits represent a variable which is a list containing three strings elements : 'apple', 'banana', 'cherry'





Lets dive into Python, from Scratch!

What is a List?

A list is a collection of items (or elements) that are ordered and changeable (mutable). Lists are defined using square brackets [], and the elements inside the list are separated by commas. Example: List of numbers: Numbers = [1, 2, 3, 4, 5] List of strings: Names = ['Ahmad', 'Sara', 'Lama', 'Mohammad'] Mixed list: Mixed = [1, 'hello', True, 3.148]



0.000

Lets dive into Python, from Scratch!

What is a List?

Accessing Elements in a List:

You can access individual elements of a list using indexing.

- **Indexing starts from 0** (just like strings).
- **Negative indexing** starts from the end.

Example:

fruits = ['apple', 'banana', 'cherry']
print(fruits[0]) # Output: 'apple'
print(fruits[1]) # Output: 'banana'
print(fruits[-1]) # Output: 'cherry' (last element)





Lets dive into Python, from Scratch!

What is a List? Changing Elements in a List

Lists are **mutable**, meaning you can change their content after they are created.

Example: fruits = ['apple', 'banana', 'cherry'] fruits[1] = 'blueberry' # Change 'banana' to 'blueberry' print(fruits) # Output: ['apple', 'blueberry', 'cherry']







Basic List Operations

Operation	Description	Example	Result
Length of list	len() returns the number of items	len([1, 2, 3])	3
Append (add to end)	list.append(item) adds an item	<pre>fruits.append('orange')</pre>	['apple', 'banana', 'cherry', 'orange']
Insert (add at index)	<pre>list.insert(index, item) adds an item at a specific index</pre>	<pre>fruits.insert(1, 'grape')</pre>	<pre>['apple', 'grape', 'banana', 'cherry']</pre>
Remove by value	<pre>list.remove(item) removes an item</pre>	<pre>fruits.remove('banana')</pre>	['apple', 'cherry']
Pop (remove by index)	<pre>list.pop(index) removes an item at a specific index (default is last item)</pre>	<pre>fruits.pop()</pre>	['apple', 'banana'] (removes 'cherry')
Concatenation	+ joins two lists	[1, 2] + [3, 4]	[1, 2, 3, 4]
Repetition	* repeats a list	[1, 2] * 3	[1, 2, 1, 2, 1, 2]







ALLE CONTRACTOR

Lets dive into Python, from Scratch!

What is a List?

Slicing Lists: You can extract a portion of a list using slicing. Syntax: list[start : end] Start from the start index and go up to end without including it Example: numbers = [0, 1, 2, 3, 4, 5]slice1 = numbers[1:4] # From index 1 to 3 print(slice1) # Output: [1, 2, 3]







Lets dive into Python, from Scratch!

What is a List? Slicing Lists: You can also use step in slicing:

numbers = [0, 1, 2, 3, 4, 5]
slice2 = numbers[0:6:2] # Every second element
print(slice2) # Output: [0, 2, 4]







Lets dive into Python, from Scratch!

What is a List?

Looping Through a List: You can use a for loop to iterate through each item in a list. Example: fruits = ['apple', 'banana', 'cherry'] for fruit in fruits: print(fruit)







List Methods

Here are some commonly used methods in Python lists:

Method	Description	Example
append(item)	Adds an item to the end of the list	<pre>fruits.append('orange')</pre>
<pre>insert(index, item)</pre>	Inserts an item at a specific index	<pre>fruits.insert(1, 'grape')</pre>
<pre>remove(item)</pre>	Removes the first occurrence of an item	<pre>fruits.remove('apple')</pre>
<pre>pop(index)</pre>	Removes and returns an item at an index	<pre>fruits.pop(0)</pre>
<pre>index(item)</pre>	Returns the index of the first occurrence	<pre>fruits.index('banana')</pre>
<pre>count(item)</pre>	Counts how many times an item appears	<pre>fruits.count('apple')</pre>
sort()	Sorts the list in ascending order	<pre>numbers.sort()</pre>
reverse()	Reverses the order of the list	numbers.reverse()

ALLER FEETENSIES







Lets dive into Python, from Scratch!

What is a List? Nested Lists: this is how we will handle Biological Data! A nested list is a list inside another list.

Example: nested_list = [[1, 2, 3], ['a', 'b', 'c']] print(nested_list[0]) # Output: [1, 2, 3] print(nested_list[0][1]) # Output: 2





Lets dive into Python, from Scratch!

Now that we've explored lists, let's move on to another essential data structure in Python: **Dictionaries**.

A **dictionary** is like a real-world dictionary where you look up a word (the key) to find its meaning (the value). In Python, dictionaries are used to store data in **key-value pairs**. They are powerful, flexible, and widely used in programming.







Lets dive into Python, from Scratch!

What is a Dictionary?

A **dictionary** is a collection of key-value pairs, where each key is unique, and it is used to access its corresponding value.

Keys can be strings, numbers, or even lists.

Values can be any data type, including numbers, strings, lists, or even other dictionaries.

Creating a Dictionary:

student = {

'name': 'Alice', 'age': 22, 'major': 'Biology'

}

name is the 1st key, Alice is its value age is the 2nd key, 22 is its value Major is the 3rd key, Biology is its value







Lets dive into Python, from Scratch!

What is a Dictionary?

Accessing Values in a Dictionary:

To access a Value, you can use the key inside square brackets [] or the get() method. Example: student = {'name': 'Alice', 'age': 22, 'major': 'Biology'} print(student['name']) # Output: Alice print(student.get('age')) # Output: 22.

Adding or Updating Key-Value Pairs:

You can add a new key-value pair or update an existing one by assigning a value to a key. student = {'name': 'Alice', 'age': 22} student['major'] = 'Biology' # Adding a new key-value pair student['age'] = 23 # Updating the value of an existing key print(student) # Output: {'name': 'Alice', 'age': 23, 'major': 'Biology'}



Lets dive into Python, from Scratch!

What is a Dictionary? Removing Key-Value Pairs: You can remove a key-value pair using the del statement or the pop() method. Example:

student = {'name': 'Alice', 'age': 22, 'major': 'Biology'}
del student['major'] # Removes the key 'major'
print(student) # Output: {'name': 'Alice', 'age': 22}
age = student.pop('age') # Removes and returns the value of 'age'
print(student) # Output: {'name': 'Alice'}
print(age) # Output: 22





Lets dive into Python, from Scratch!

What is a Dictionary?

Looping Through a Dictionary

You can loop through a dictionary to access its keys, values, or both. Example: student = {'name': 'Alice', 'age': 22, 'major': 'Biology'} # Loop through keys for key in student: print(key) # Output: name, age, major

Loop through values
for value in student.values():
 print(value) # Output: Alice, 22, Biology

Loop through key-value pairs
for key, value in student.items():
 print(key, ':', value)





Lets dive into Python, from Scratch!

What is a Dictionary? Nested Dictionaries:

You can store dictionaries inside other dictionaries, which is useful for representing more complex data. students = { 'student1': {'name': 'Alice', 'age': 22},

```
'student2': {'name': 'Bob', 'age': 24}
```

print(students['student1']['name']) # Output: Alice





Dictionary Methods

Here are some commonly used dictionary methods:

Method	Description	Example
<pre>dict.get(key)</pre>	Returns the value for the specified key	<pre>student.get('name')</pre>
<pre>dict.keys()</pre>	Returns all keys as a list	<pre>student.keys()</pre>
<pre>dict.values()</pre>	Returns all values as a list	<pre>student.values()</pre>
<pre>dict.items()</pre>	Returns all key-value pairs	<pre>student.items()</pre>
<pre>dict.update()</pre>	Updates the dictionary with key-value pairs from another dictionary	<pre>student.update({'grade': 'A'})</pre>
<pre>dict.clear()</pre>	Removes all items from the dictionary	<pre>student.clear()</pre>







Basic Dictionary Operations

Operation	Description	Example	Result
Access a value	dict[key]	<pre>student['name']</pre>	'Alice'
Add/Update a value	<pre>dict[key] = value</pre>	<pre>student['age'] = 23</pre>	{'name': 'Alice', 'age': 23}
Remove a key- value pair	<pre>del dict[key] or dict.pop(key)</pre>	<pre>del student['age']</pre>	{'name': 'Alice'}
Check if key exists	key in dict	'age' in student	True
Get all keys	<pre>dict.keys()</pre>	<pre>student.keys()</pre>	<pre>dict_keys(['name', 'age'])</pre>
Get all values	dict.values()	<pre>student.values()</pre>	<pre>dict_values(['Alice', 22])</pre>
Get all key-value pairs	<pre>dict.items()</pre>	<pre>student.items()</pre>	<pre>dict_items([('name', 'Alice'), ('age', 22)])</pre>




