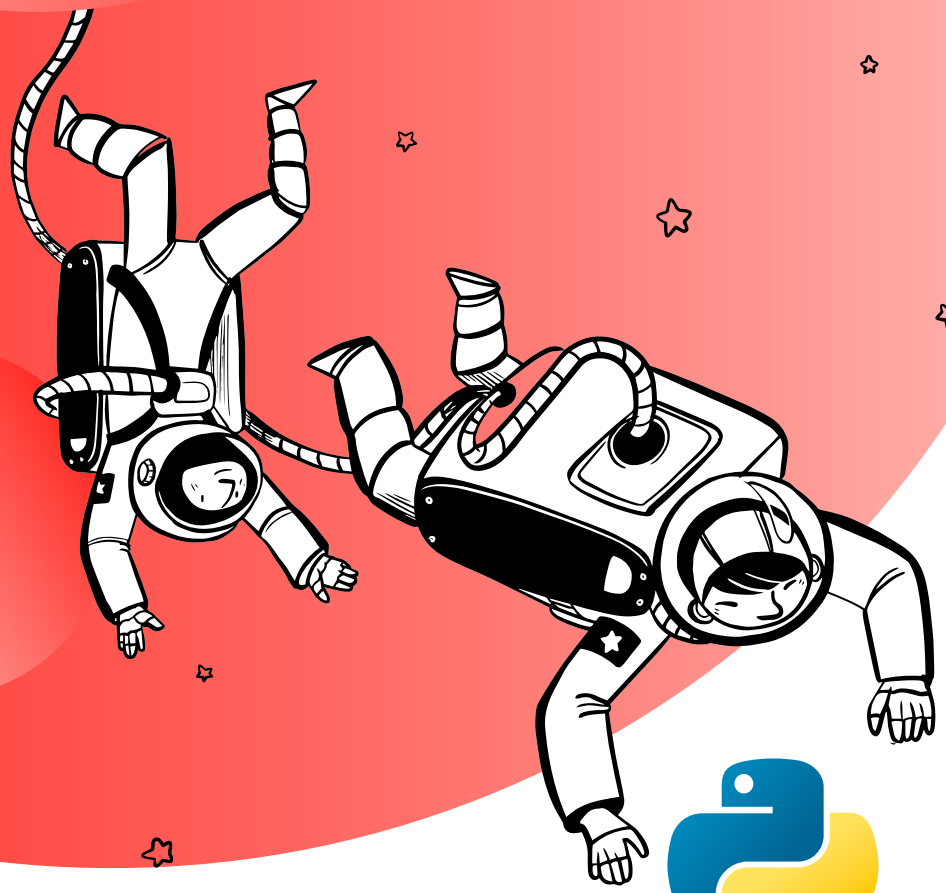
atoti.



What is Python?

In data science

atoti.

Python is an easy-to-learn programming language with a long history.

Python History

1991 — Van Rossum publishes **Python version 0.9.0** to alt.sources

1994 — **Python 1.0** includes functional programming (lambdas, map, filter, reduce)

2000 — **Python 2** introduces list comprehension and garbage collection

2008 — **Python 3** fixes fundamental design flaws and is not backwards compatible

2020 — **Python 2** is end of life, last version 2.7.18 released

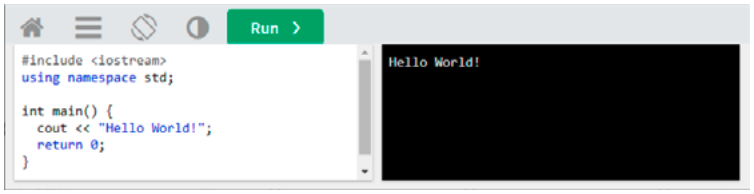
Timeline from https://python.land/python-tutorial/python-history.
Go to their webpage for full history of Python

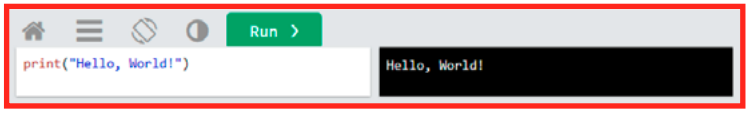
in @atoti

atoti.

How easy is it?

Let's compare with other languages using "Hello World".





Intuitive syntax

Try them out @ w3schools.com

- General-purpose language for development and deployment
- Used by developers, data engineers and data scientists
- Less constraints on memory usage, therefore it can easily handle large data
- Highly interactive visualizations and dashboarding tools such as atoti, Plotly and Seaborn.

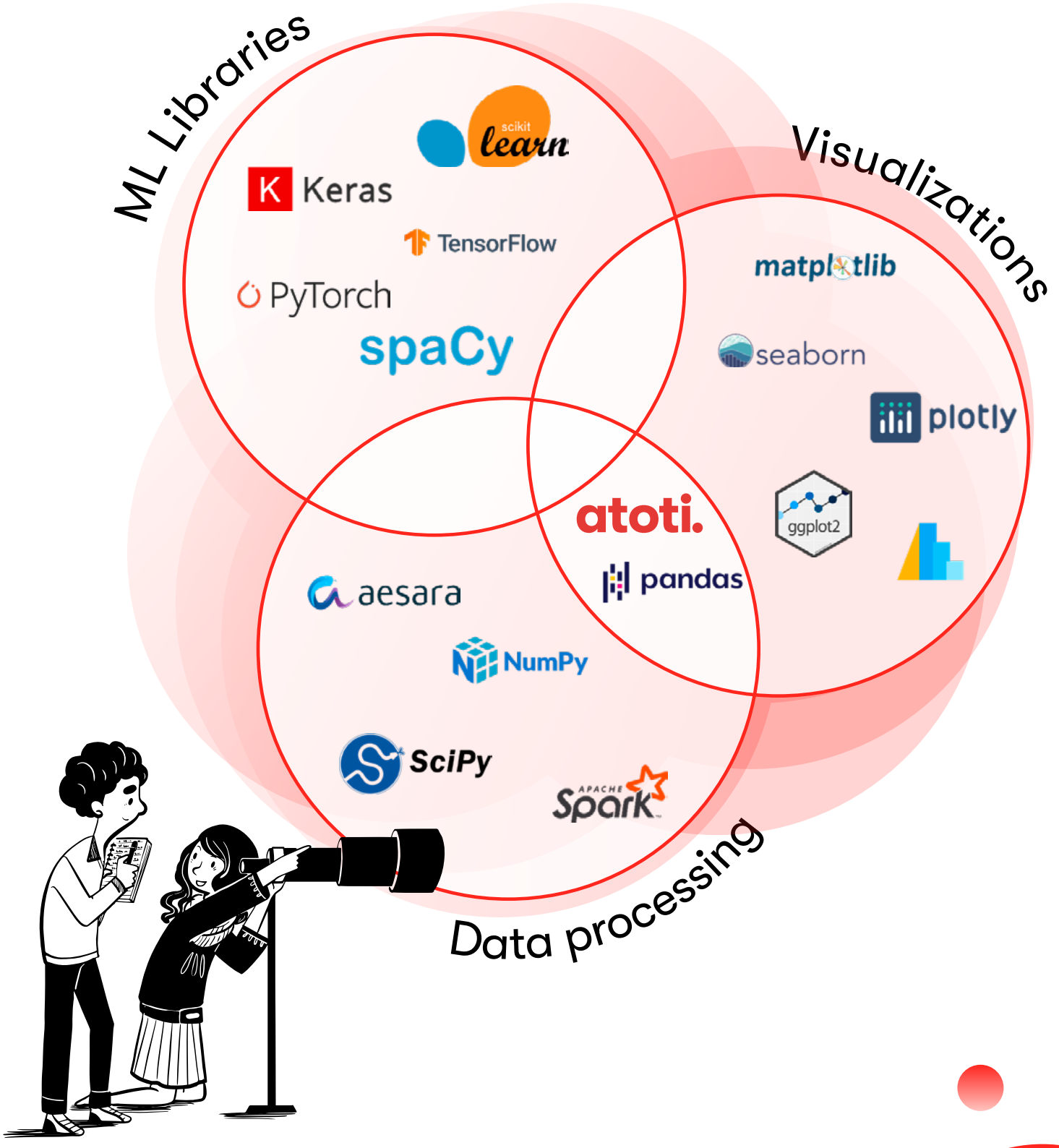
- Open-source statistical programming language
- Used by statisticians, analysts and data scientists for statistical analysis and graphical techniques
- Computes everything in memory, therefore it's limited by RAM size.
- Highly customizable static graphing tools such as ggplot2

@atoti

atoti.

Why is Python popular in Data Science?

Because it has amazing libraries!

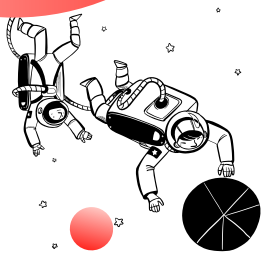


* Above are examples of Python libraries available for use but not limited to these.

@atoti

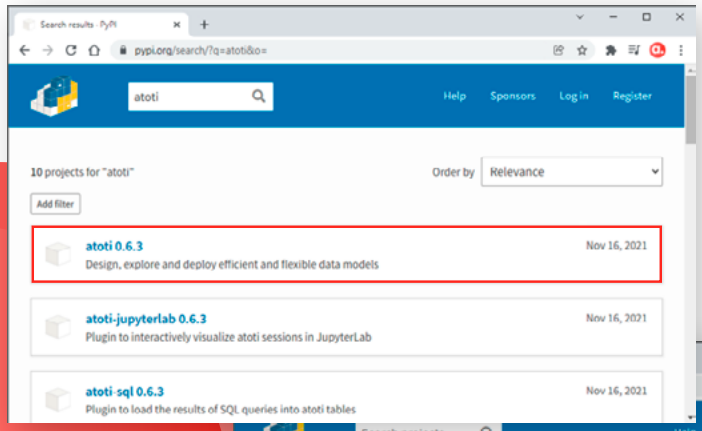
atoti.

How do we get libraries?

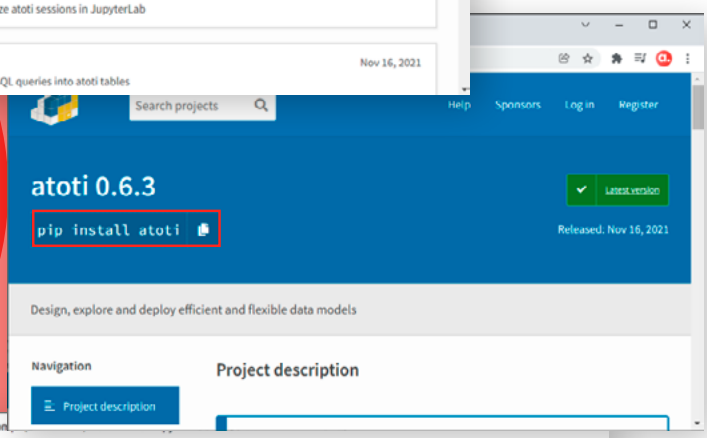


1. Using pip

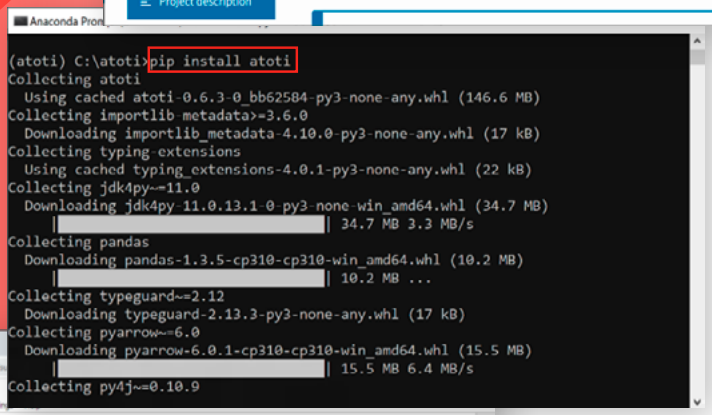
A repository of software for the Python programming language. Install package from https://pypi.org/.



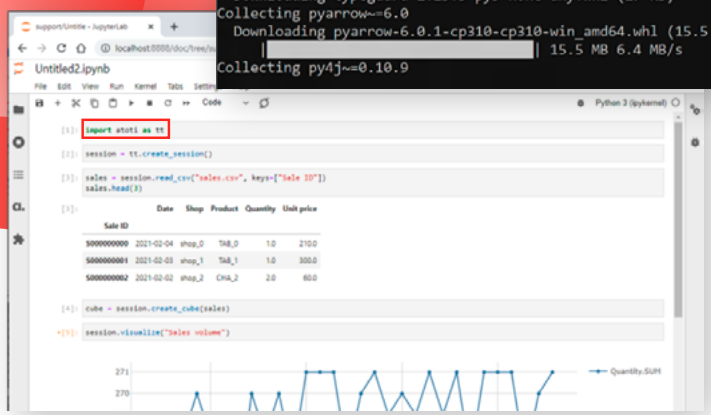
1 Search and select Python library



2 Copy pip install command



3 Run command in console



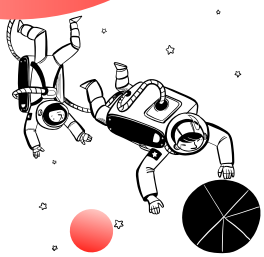
4 Import library and start using

* Installation instruction can usually be found on the library's website or GitHub repository.

@atoti

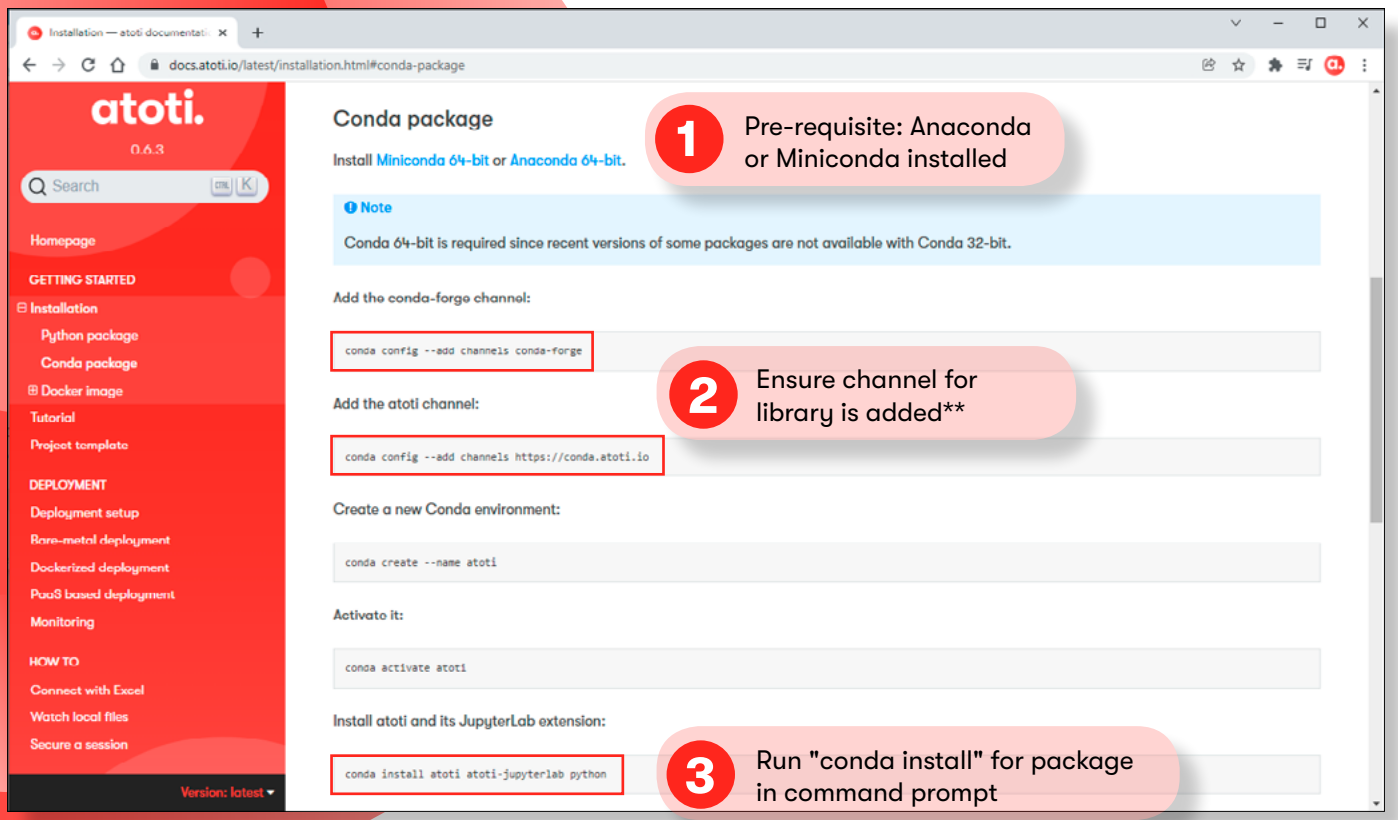
atoti.

How do we get libraries?



2. Using conda

An open source package management system and environment management system.
Install package from https://repo.anaconda.com/.



* Conda installs packages which may contain software written in any language.
** "conda-forge" channel is free for all to use.

Download conda cheatsheet from https://docs.conda.io/projects/conda/en/latest/user-guide/cheatsheet.html.

atoti.

Virtual environment

Create an isolated environment for Python projects with its own version of Python interpreter, libraries and scripts installed into it.

pip

Create environment

```
python3 -m venv myenv
```

Ensure using Python 3 is used instead of Python 2

Activate environment

- Unix / MacOS

```
source myenv/bin/activate*
```

*Use activate.csh/activate.fish for csh/fish shell

- Windows

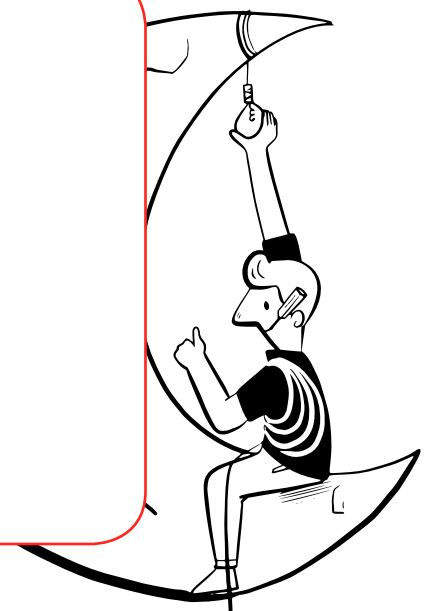
```
myenv\Scripts\activate.bat
```

Deactivate environment

```
deactivate
```

Remove environment

Close environment and delete directory



CONDA

Create environment

```
conda create --name myenv python=3.8 atoti scipy
```

Create env with specific Python version

Install multiple packages

Activate environment

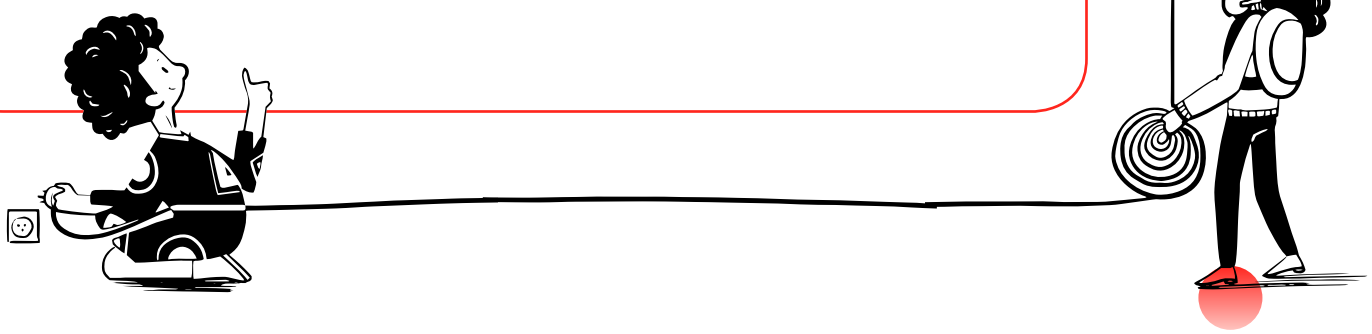
```
conda activate myenv
```

Deactivate environment

```
conda deactivate
```

Remove environment

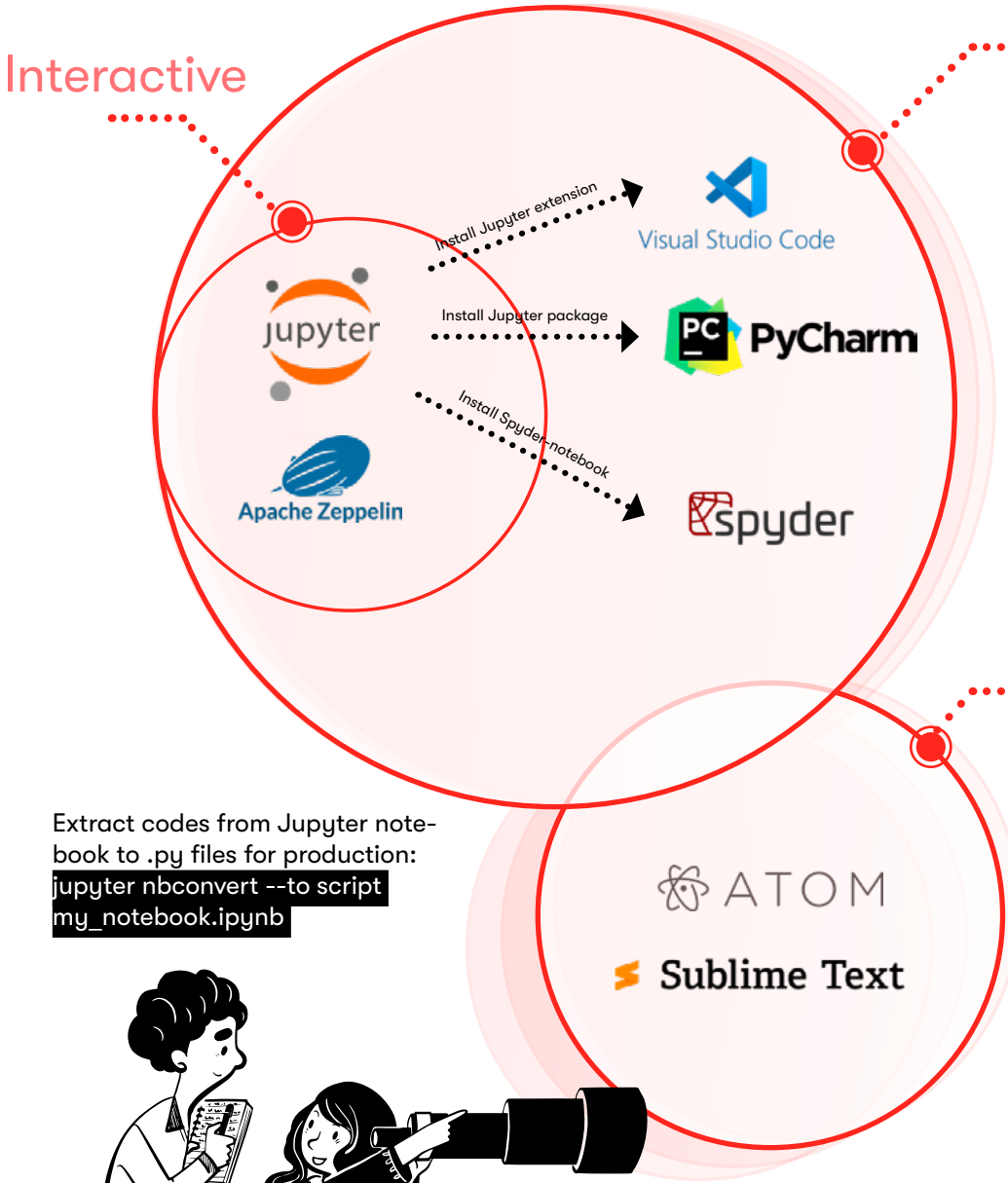
```
conda remove --name myenv --all
```



in @atoti

atoti.

Integrated development environment (IDE) vs Code editor



A set of tools that makes coding easier: text editor, compiler, build, debugging, auto linting function, auto-completion, etc

Text editors with powerful built-in features and specialized functionalities to simplify and accelerate code editing process

Extract codes from Jupyter notebook to .py files for production:
`jupyter nbconvert --to script my_notebook.ipynb`

* Above are examples of IDE and code editors for Python but not limited to these.

in
@atoti

atoti.

Dependency management

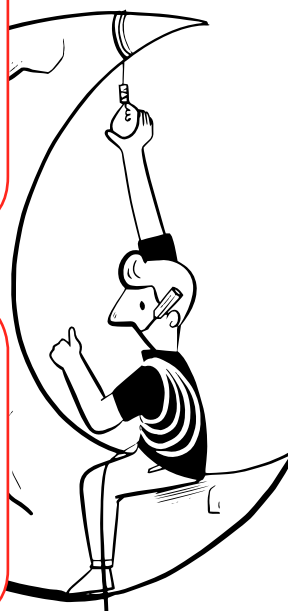
Dependencies are all the software components required by the project to work as intended.

pip

1. Set up virtual environment
2. Install required modules
3. pip freeze > requirements.txt

To install dependencies in new env: pip install -r requirements.txt

https://docs.python.org/3/tutorial/venv.html



CONDA

1. Set up virtual environment
2. Install required modules
3. conda list -e > requirements.txt

To install dependencies in new env:
conda install -n <env_name> requirements.txt

Poetry

1. poetry init *
2. poetry add <package>
3. pyproject.toml file updated with dependency
4. sub-dependency version updated in poetry.lock

To install dependencies in new env:

1. pyproject.toml present
2. poetry install

* To use poetry to control an existing project.
Use "poetry new <package name>" to start new project.

in
@atoti

atoti.

Beginner's Cheat Sheet

Base Type

Category	Data types	Description	Examples
Numeric Types	int	whole number, positive or negative, without decimals, of unlimited length	0, 1, -327
	float	number, positive or negative, containing one or more decimals	0.0, 2.8, -312.67, 1.102e+12 (x10^12)
Boolean Type	str	immutable sequences of Unicode characters String surrounded by single or double quotes. Triple quotes for multiline string. Backslash to escape characters. \' ~ single quote \\ ~ backslash \b ~ backspace \n ~ new line \r ~ carriage return \t ~ tab	"Hello world", 'Hello world', """ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.""" 'It isn\'t where you came from. \nIt's where you\'re going that counts. '
Boolean Type	bool		True, False
Binary Types	bytes	immutable sequences of single bytes	[1]: b"""Line 1. Line 2. """ [1]: b'Line 1.\nLine 2. \n'
	bytearray	mutable counterpart to bytes objects	[1]: byte_arr=bytearray("""Line 1. Line 2.""", "utf8") print(byte_arr) bytearray(b'Line 1.\nLine 2.') [2]: del byte_arr[0:8] print(byte_arr) bytearray(b'Line 2.')
	memoryview	Access the internal data of an object without copying, unlike bytes/str	[1]: mv = memoryview(byte_arr) print(bytes(mv[0:7])) b'Line 2.' [2]: mv[6] = 33 print(byte_arr) bytearray(b'Line 2!')

@atoti

atoti.

Beginner's Cheat Sheet

Base Type

Category	Data types	Description	Examples
Sequence Types	list	Container holding comma-separated values between square brackets. Mutable.	["apple", "banana", "cherry"], ['Friday', datetime.date(2022, 1, 14), 2022, 'Jan', 14]
	tuple	Container holding comma-separated values (parentheses are optional). Immutable.	("apple", "banana", "cherry"), date_tuple = "Friday", datetime.date(2022, 1, 14), 2022, "Jan", 14
	range	Function returning sequence of numbers, starting from 0 and increments by 1 by default. range(start, stop, step)	range(6), range(1,6)

Mapping Types	dict	Ordered collection of data stored in key-value pairs. Key is unique.	date_dict = {
			"day_of_week": "Friday",
			"date": datetime.date(2022, 1, 14),
			"day": 14,
			"month": "Jan",
			"year": 2022
			}

Set Types	set	Unordered collection of data that is mutable, iterable and has unique elements.	{"apple", "banana", "cherry"}, {"Friday", datetime.date(2022, 1, 14), 2022, "Jan", 14}
	frozen-set	Freeze iterable objects, making them unchangeable.	frozenset({"apple", "banana", "cherry"})

@atoti

atoti.

Beginner's Cheat Sheet

Type conversions

From type → To type	Examples
str → int	int("10") → 10
float → int	int(10.23) → 10
anything → str	str(10.10) → '10.1', str(1==1) → 'True'
ASCII code → char	chr(36) → '$'
char → ASCII code	ord('$') → 36
str → list	list("atoti") → ['a', 't', 'o', 't', 'i'] "The quick brown fox jumped over the lazy dog".split( ) → ['The', 'quick', 'brown', 'fox', 'jumped', 'over', 'the', 'lazy', 'dog']
dict → list	d = {"day": 14, "month": "Jan", "year": 2022} list(d.items( )) → [('day', 14), ('month', 'Jan'), ('year', 2022)] list(d.keys( )) → ['day', 'month', 'year'] list(d.values( )) → [14, 'Jan', 2022] [(k, v) for k, v in d.items( )] → [('day', 14), ('month', 'Jan'), ('year', 2022)] list(zip(d.keys( ), d.values( ))) → [('day', 14), ('month', 'Jan'), ('year', 2022)] list(map(list, d. items())) → [('day', 14), ('month', 'Jan'), ('year', 2022)]
list → dict	dict([('day', 14), ('month', 'Jan'), ('year', 2022)]) → {"day": 14, "month": "Jan", "year": 2022}
list → set	set(['a', 't', 'o', 't', 'i']) → {'a', 'i', 'o', 't'} Only unique items in set, only 1 't' in result.
list → string	" ".join(['The', 'quick', 'brown', 'fox', 'jumped', 'over', 'the', 'lazy', 'dog']) → 'The quick brown fox jumped over the lazy dog'

@atoti

atoti.

Beginner's Cheat Sheet

Strings

Print

Code	Results
`print("Hello")`	Hello
`print("Hello", "world")`	Hello world
`print("Hello", "world", sep="...")`	Hello...world
`print("Hello", "world", sep="...", end="!")`	Hello...world!
`print("Hello") # each print uses a newline` `print("world")`	Hello world
`print("Hello", end=" ") # without newline` `print("world")`	Hello world

Concatenation

Code	Results

`a = "Hello"`
`b = "world"`
`year = 2022`
`print(a + " " + b + " " + str(year))`
`# cast numeric value to str before concatenation`
	Hello world 2022

Formatting

Code	Results
`"Price: ${price:,.2f}".format(price = 1000)`	'Price: $1,000.00'
`"{0}, {1}, {2}".format("apple", "banana", "cherry")`	'apple, banana, cherry'
`"{2}, {1}, {0}".format(*"abc") # unpacking argument sequence`	'c, b, a'

`"{:*<30}".format("left aligned") # with * as a fill char`
or
`"left aligned".ljust(30, "*")`
	'left aligned**********'

`"{:>30}".format("right aligned")`
or
`"right aligned".rjust(30)`
	' right aligned'

@atoti

atoti.

Beginner's Cheat Sheet

Strings

Formatting

Code	Results

`"{:^30}".format("centered")`
or
`"centered".center(30)`
	' centered '

`name = "Jane"` `f"Hello, {name}!"`	'Hello, Jane!'
`"198".zfill(5) # zero pad string`	'00198'

`import datetime`
`d = datetime.datetime(2022, 1, 14, 10, 15, 58)`
`"{:%Y-%m-%d %H:%M:%S}".format(d)`
	'2022-01-14 10:15:58'

Change case:
`"apple".capitalize( ) # Converts the first character to upper case`
`"APPLE".casefold( ) # Converts string into lower case`
`"APPLE".lower( ) # Converts a string into lower case`
`"apple".upper( ) #Converts a string into upper case`
`"Apple".swapcase( ) # Swaps cases, lower case becomes upper case and vice versa`
`"apple".title( ) # Converts the first character of each word to upper case`
	'Apple'
	'apple'
	'apple'
	'APPLE'
	'aPPLE'
	'Apple'

String functions

Code	Results

`"apple".startswith("a")`
`"apple".endswith("e")`
`"  apple  ".strip( )`
`"apple".count("p")`
`"apple".index("l")`
`"apple".find("l")`
`"apple".find("x")`
	True → string started with "a"
	True → string ended with "e"
	'apple' → remove whitespace before & after string
	2 → 2 "p"s are found in string
	3 → index of "l" is 3
	3 → "l" found at index 3 (index starts with 0)
	-1 → "x" not found in string

@atoti

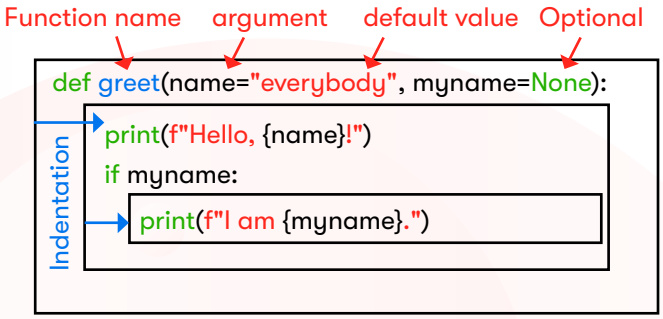
atoti.

Beginner's Cheat Sheet

Functional modules

Functions

Code




```
greet( )
greet("Jane")

greet("Jane", "Thomas")
```

Results

```
Hello, everybody!
Hello, Jane!

Hello, Jane!
I am Thomas.
```

Exception handling

Code

```
try:
   print("Hello " + 123)
except Exception as e:
   print(f"Error occurred: {str(e)}")
finally:
   print("Gets printed no matter what")
```

Results

```
Error occurred: can only concatenate str (not "int") to str
Gets printed no matter what
```

Code

```
num1 = 1
num2 = "1"
if num1 != num2:
   raise Exception(f"num1 ({num1}) is not equal to num2 ({num2})")
```

Results

```
Exception: num1 (1) is not equal to num2 (1)
```

in
@atoti

atoti.

Beginner's Cheat Sheet

Functional modules

Modules/Names Import

Code	Results

`import datetime` → access datetime module
 	
`datetime.datetime.now()`
`datetime.date.today()`
`datetime.date.today() + datetime.timedelta(days=10)`
	datetime.datetime(2022, 1, 17, 10, 52, 43, 795534)
	datetime.date(2022, 1, 17)
	datetime.date(2022, 1, 27)

`from datetime import date, datetime, timedelta` → direct function access
`from datetime import date as dt` → give function an alias
 	
`datetime.now()`
`date.today()`
`date.today() + timedelta(days=10)`
 	
`dt.today()`
	datetime.datetime(2022, 1, 17, 10, 52, 43, 795534)
	datetime.date(2022, 1, 17)
	datetime.date(2022, 1, 27)
	
	datetime.date(2022, 1, 17)

`def greet(name):`
`  print(f"Hello, {name}!")` → define function and saved in hello.py in same directory as main prog
 	
`from hello import greet`
`greet("Thomas")`
	Hello, Thomas!

Iterative looping

Code	Results

Indentation
`for var in sequence:`
→ Do something
 	
`for num in range(1,6):`
`  print(num, end=",")`
	1,2,3,4,5,

`fruits = ["apple", "banana", "cherry"]`
 	
`for f in fruits:`
`  if f == "banana":`
`  break` → exit from the loop
 	
`print(f)`
	apple

`i = 1` → initialize variable with value
`while i <= 10:`
`  i += i` → conditional iteration; stops when i becomes greater than 10
`print(i)`
	16

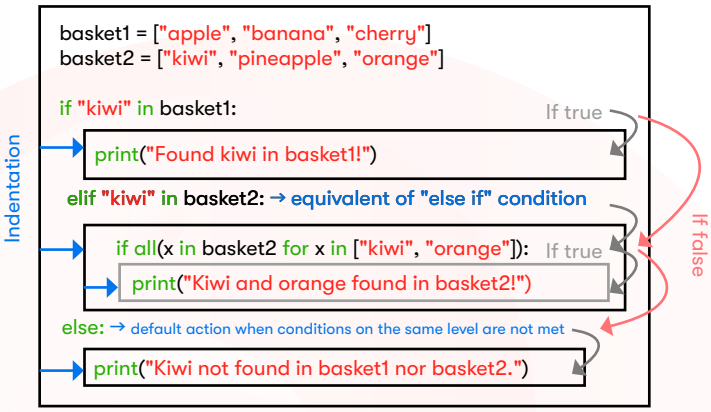
in
@atoti

atoti.

Beginner's Cheat Sheet

Strings

Conditional

Code	Results
	Kiwi and orange found in basket2!

Read/Write files

Code	Results
Python's built-in func **f = open("file.txt", "w")** ← w: create a file if it does not exist x: create a file, returns error if file already exists **f.write("Existing content will be cleared!")** **f.close()**	
f = open("file.txt", "a") ← a: append mode **f.write("\nThis will be appended to existing content! ")** **f.close()** To append as newline	
f = open("file.txt", "r") ← r: read-only **print(f.read( ))**	Existing content will be cleared! This will be appended to existing content!

Refer to the following sites for more Python examples:
https://www.w3schools.com/python/
https://www.geeksforgeeks.org/

Check out https://www.atoti.io/ to find out how to create a BI analytics platform in Python.
Stay tuned to https://www.youtube.com/atoti for useful tutorials.

in @atoti