



Python for DevOps

Module 1. Introduction to Python



Programming language types

Interpreted vs Compiled Programming

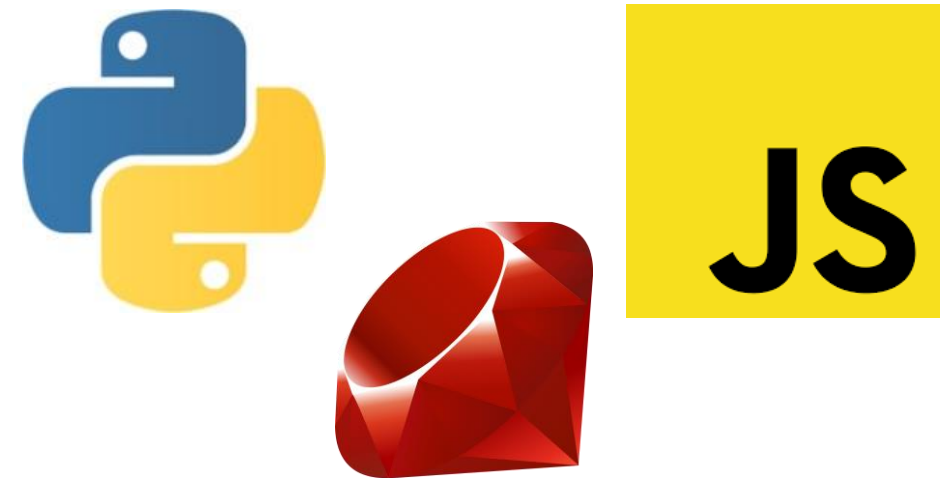
Compiled Languages

- We convert the code itself into machine understandable format
- More memory or CPU usege management
- Faster
- Need to be build first
- We depend on platform



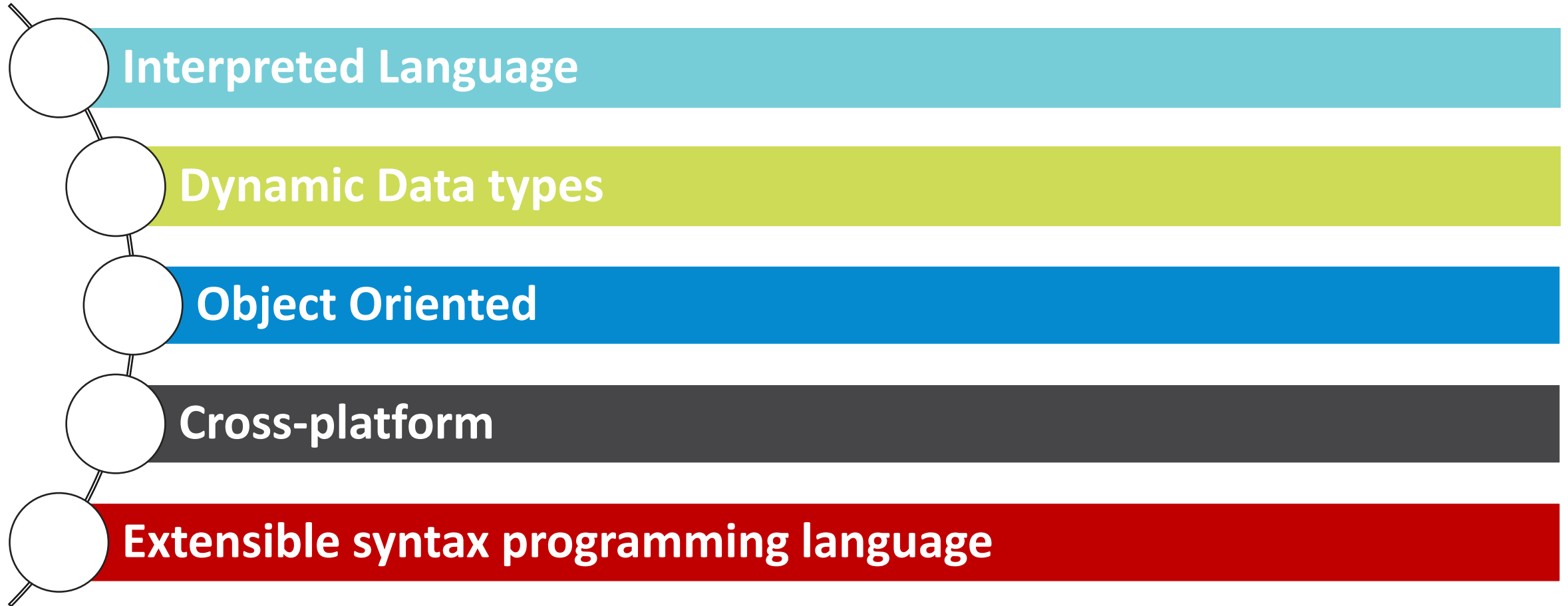
Interpreted Languages

- Interpreters reads code line by line and executes each command
- Slower (but we have JiT compilation so it's not 100% true)
- Smaller program size
- Easier to work with
- Good for proptotiping



WHAT IS PYTHON?

So, what's about Python?



History

- Designed by: Guido van Rossum
- Developer: Python Software Foundation

Milestones:

0.9 version – 1991 – 1993

1.0 version – 1994 – 1994

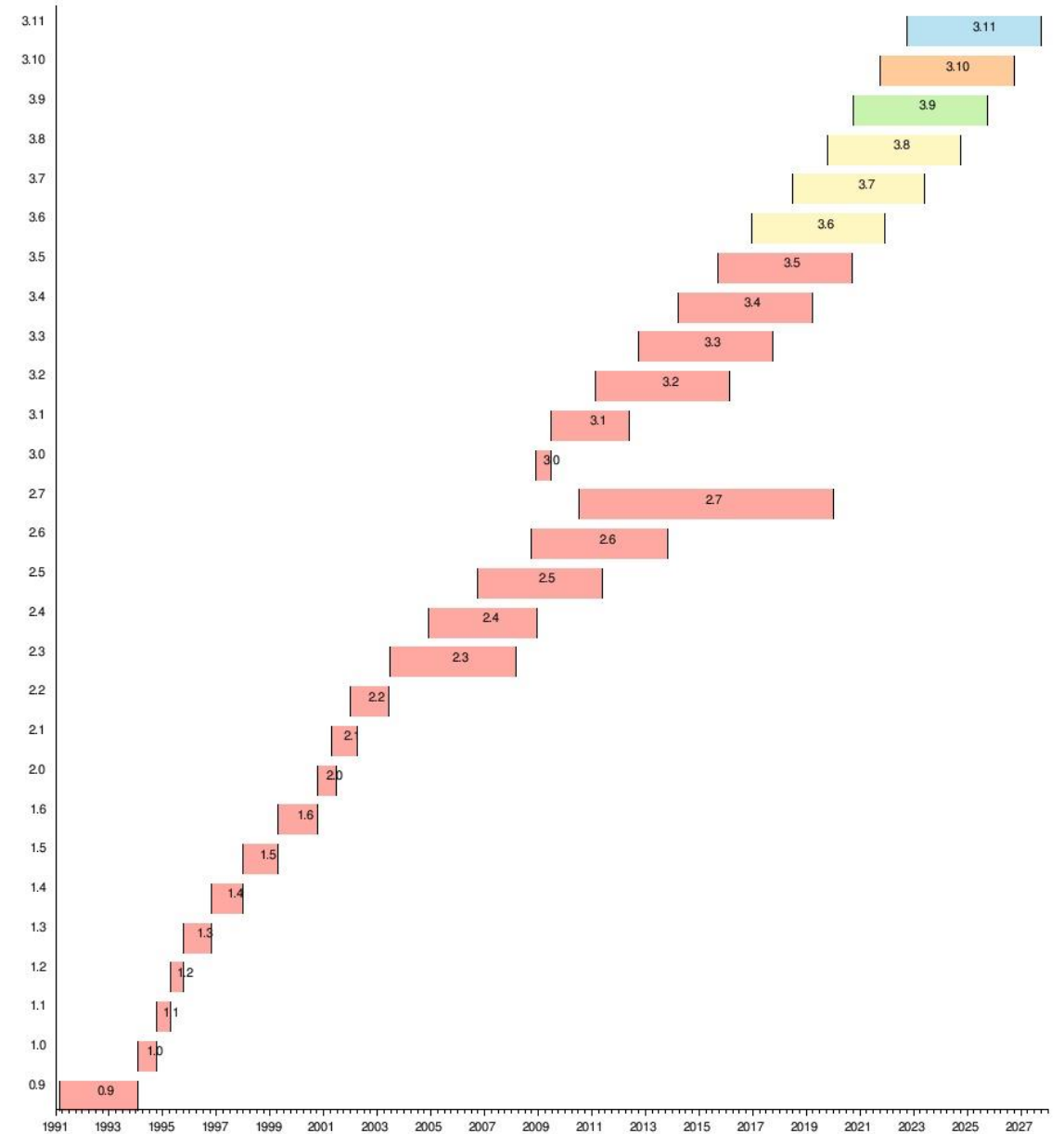
2.0 version – 2000 – 2001

2.7 version – 2010 – 2020

3.6 version – 2016 – 2021

3.9 version – 2020 – 2025

3.11 - version – 2024 – 2027



Let's install

LINUX

From repository:

- Ubuntu:

```
$ sudo apt update  
$ sudo apt install python3
```

- Centos

```
$ sudo yum update  
$ yum install -y python3
```

From Source Code:

<http://www.python.org/download/>

```
$ ./configure  
$ Make  
$ make install
```

WINDOWS

- <https://www.python.org/downloads/release/python-397/>
- python-XYZ.msi

MACOS

```
$ brew install python3
```

Integrated Development Environment (IDE)



```
27
28
29 @property
30 def debug(self):
31     return self._debug
32
33 def add_middleware(self, middleware_cls, **kwargs):
34     self._middleware.add(middleware_cls, **kwargs)
35
36 def route(self, pattern, methods=None):
37     """Decorator that adds a new route"""
38     def wrapper(handler):
39         self.add_route(pattern, handler, methods)
40         return handler
41     return wrapper
42
43 def add_route(self, pattern, handler, methods=None):
44     """Add a new route"""
45     assert pattern not in self._routes
46     self._routes[pattern] = Route(path_pattern=pattern, handler=handler,
47                                   methods=methods)
48
49 def add_exception_handler(self, handler):
50     self._exception_handler = handler
51
52 def handle_exception(self, request, response, exception):
53     self._exception_handler(request, response, exception)
```

```
160
161 if time_limit < self.error_time_limit_max:
162     return time_limit
163
164 return self.error_time_limit_max
165
166 @property
167 def error_time_limit_max(self):
168     return self.error_time_started + timedelta(minutes=15)
169
170 def _error(self):
171     if not self.errors:
172         self.error_time_started = datetime.now()
173
174     self.errors += 1
175
176 def _error_reset(self):
177     self.errors = 0
178     self.error_time_started = None
179
180 @property
181 def is_allowed(self):
182     if not self.errors:
183         return True
184
185     if datetime.now() > self.error_time_limit_max:
186         self._error_reset()
187         return True
188
189     return datetime.now() > self.error_time_limit
190
191 class ElasticClusterCollector(CollectorBase):
192     def collect(self):
193         if not self.is_allowed:
194             raise CollectorError(f'Request is not allowed until {self.error_time_limit}')
195
196         try:
197             es = Elasticsearch(url=self.url)
198             cluster = ClusterMetrics(es=es)
199             for metric in cluster.metrics:
200                 yield metric
201         except:
202             self._error()
203             raise
```


Interpreter usage

```
python3
```

```
python3 myscript.py arg1 arg2
```

Add Interpreter to path

Linux:

```
export PATH="$PATH:/usr/local/bin/python"
```

Windows:

```
set path=%path%;C:\python39;C:\python39\Scripts
```

Enviroment variables

PYTHONHOME

Location of the standard Python libraries

PYTHONPATH

The default search path for module files

PYTHONCASEOK

Python ignores case in import statements. This only works on Windows and OS X.

PYTHONSTARTUP

If this is the name of a readable file, the Python commands in that file are executed before the first prompt is displayed in interactive mode.

Interactive mode

```
➔ ~ python3.9
Python 3.9.6 (default, Jun 29 2021, 05:25:02)
[Clang 12.0.5 (clang-1205.0.22.9)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

What can we do here?

- Write actual code
- Run tests
- It's a cool calculator

How can I run my script??

Code:

```
#!/usr/bin/python  
  
print("Hello World!")
```

Make it executable:

```
$ chmod +x test.py
```

Run it!!

```
$ ./test.py
```



```
→ ~ vim test.py  
→ ~ cat test.py  
#!/usr/bin/python  
  
print("Hello World!")  
→ ~ chmod +x test.py  
→ ~ ./test.py  
Hello World!  
→ ~
```

Syntax

Identifiers and Variables

VARIABLE

The name “variable” based on idea that something is able to vary.

Basically, it’s a memory cell that contains some data that we can read via our code.

IDENTIFIERS

“An identifier is a name given to an entity”

In other words:

Identifier is a user-defined name to represent the basic building blocks of Python

Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.

```
>>> counter = 100          # An integer assignment
>>> miles    = 1000.0      # A floating point
>>> name     = "John"      # A string
>>>
>>> print(counter)
100
>>> print(miles)
1000.0
>>> print(name)
John
>>> print(f'{name} managed to go {miles}')
```

Rules for Identifiers

The Python identifier is made with a combination of **lowercase** or **uppercase letters**, **digits** or an **underscore**.

Lowercase letters (a to z)

Uppercase letters (A to Z)

Digits (0 to 9)

Underscore (_)

It cannot start with digit!
You cannot use symbols like !, #, @, %, \$

Best practice

- Class names should start with a capital letter and all the other identifiers should start with a lowercase letter
- Begin private identifiers with an underscore (_)
- Use double underscores (__) around the names of magic methods
- Always prefer using names longer than one character. `index=1` is better than `i=1`
- It's better to use combined names like `car_colour` instead of `carcolour` or `CarColour`

Key words

Interactive help: in python interpreter type *help()* and after it *keywords*

```
help> keywords
```

```
Here is a list of the Python keywords.  Enter any keyword to get more help.
```

False	break	for	not
None	class	from	or
True	continue	global	pass
__peg_parser__	def	if	raise
and	del	import	return
as	elif	in	try
assert	else	is	while
async	except	lambda	with
await	finally	nonlocal	yield

Let's talk about style

Lines and spaces:

```
if True:
    print("True")
else:
    print("False")
...
True
```

Multiline

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```

Quotes

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

Comments

Don't forget about comments! It'll make your life easier

```
# First comment  
print("Hello, Python!") # second comment
```

Instruments

Pip

Pip is a package manager for Python. It helps you to install components to use them in your code. Also, you can use it to manage your libraries, dependencies which are not a part of standard Library

```
➔ ~ python3 -m pip --version  
pip 21.1.3 from /usr/local/lib/python3.9/site-packages/pip (python 3.9)  
➔ ~
```

Virtualenv & venv

The venv module provides support for creating lightweight “virtual environments” with their own site directories, optionally isolated from system site directories.

```
python3 -m venv /path/to/new/virtual/environment
```