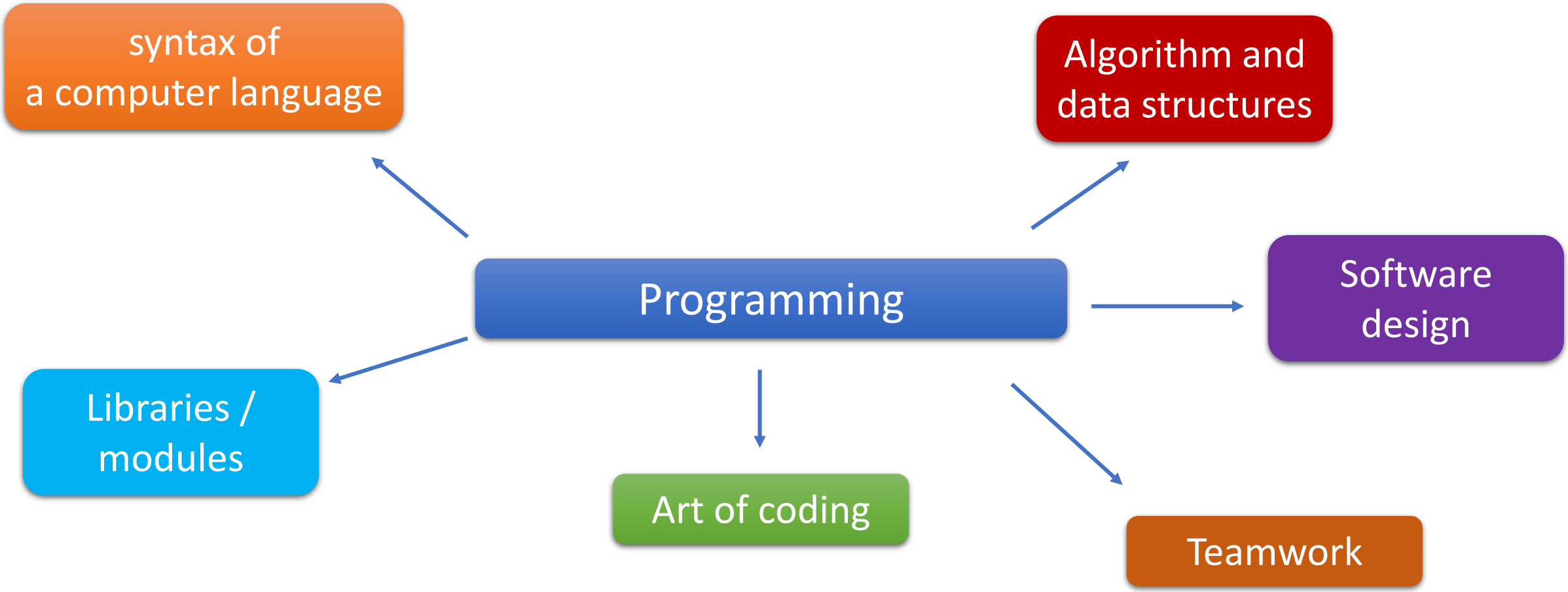
Python for beginners

06/08/2020



Tips:

- No secret sauce
- Specialize
- Google

- Programming is problem-solving, not typing code
- Just build it

Programming languages for science

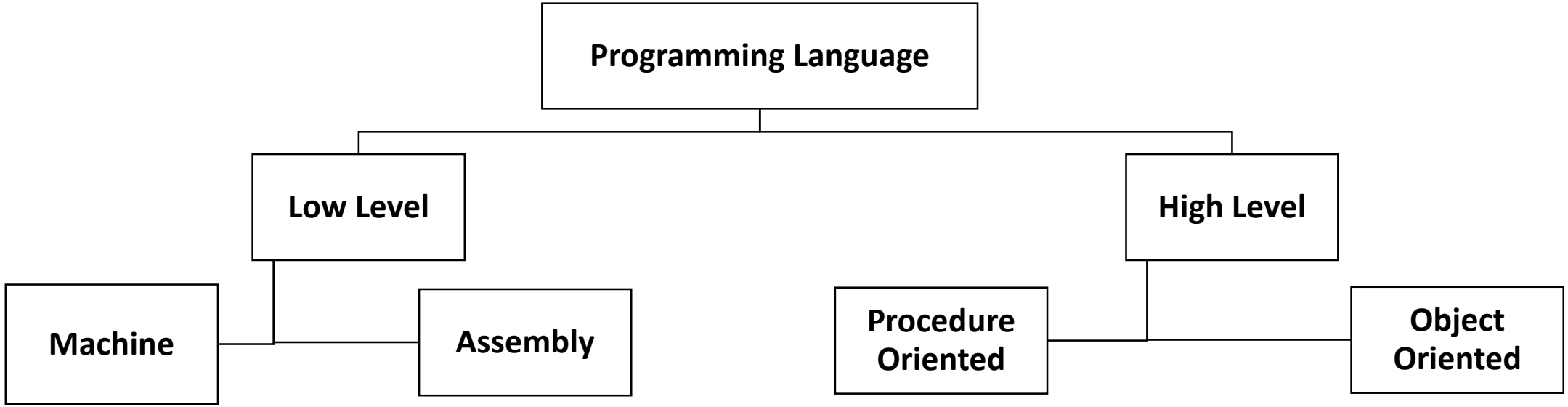
Most programming languages used by scientists were designed from the beginning to handle numerical and scientific tasks:

- R
- MATLAB
- Igor
- Mathematica
- IDL

Advantages of specialized languages:

- Usually the first to support high-level functionality needed in science
- Language and programming environment are tailored to meet the needs of scientific developers
- Lower learning curve

General-purpose languages



useful for any type of programming, regardless of the topic or functionality needed.

Advantages of general-purpose languages:

- More flexibility
- Languages require a lot of effort to learn; general-purpose languages offer greater return on that investment
- Much larger community of developers, greater longevity
- Better language design

Why python?

```
#include<iostream.h>
#include<conio.h>
void main()
{
  clrscr();
  int a,n;
  cout<<"enter n ";
  cin>>n;
  for(a=1;a<=n;a++)
   cout<<a<<" ";
   getch();
}
```

INPUT=>10

OUTPUT=>1 2 3 4 5 6 7 8 9 10

INPUT = >>print(list(range(1,11)))

OUTPUT=> 1 2 3 4 5 6 7 8 9 10

Python is a community of developers

Commercial development environments like MATLAB or Igor benefit from a monolithic community structure:

- One official source for core software packages
- One default integrated development environment (IDE) used by everybody
- One cohesive community of developers and staff for support
- Usually excellent, comprehensive documentation

Design philosophy
Created by Guido van Rossum and first released in 1991, Python was designed to be a highly readable language and community based.

PEP stands for Python Enhancement Proposal. A PEP is a design document providing information to the Python community, or describing a new feature for Python or its processes or environment. The PEP should provide a concise technical specification of the feature and a rationale for the feature (https://www.python.org/dev/peps/)

“There should be one—and preferably only one—obvious way to do it” – PEP20 (The Zen of Python)

Python is an *interpreted* programming language.

Programming languages can be compiled, interpreted, or a hybrid of the two.

Compiled languages like C, C++, Java, and Julia take the program you write and convert it into optimized, machine-executable code. Often, compiled languages are both **faster to execute** and **more difficult to use**.

Interpreted languages like Python, MATLAB, Igor, and PHP use a pre-compiled interpreter to read your program code and execute it, one step at a time. Often, interpreted languages are **slower to execute** and **easier to use**.

Question: Is Python a slow language?

Answer: It depends on how you use it.

Where to start with Python

Where to get help

Python Documentation
https://www.python.org/doc/

Tutorial
https://docs.python.org/3/tutorial

Library Reference
https://docs.python.org/3/library/index.html

Python cheat sheets
https://www.datacamp.com/community/data-science-cheatsheets

Syllabus

- Python Programming Basics (week 1)
  - Variables
  - Operations
  - Numeric Types
  - Basic Arithmetic
  - Comparisons and Logic
  - String Type
  - Indexing/Slicing Basics
  - Flow control
  - Useful Data Structures - Dictionaries/Tuples
  - Functions and Modules

- Python libraries (week 2 and 3)
  - Introduction to Numpy
  - Scientific Computing with Numpy
  - A (very brief) Introduction to Pandas
  - Basic Plotting with Matplotlib

- Working with data (week 4)

Homework:

- Installation and Setup locally (week 1)
  - Running Python commands interactively
  - Running scripts from the command line
  - The Spyder IDE
  - Jupyter Notebooks
  - Git

- Team projects (week 2 and 3)
  - Choosing data set
  - Importing data
  - Descriptive statistics
  - Data visualization

- Team project presentation (week 4)

Intro to Python

Python Syntax

A Python program is divided into a number of logical lines and every logical line is terminated by the token NEWLINE.

A comment begins with a hash character(**#**) which is not a part of the string literal and ends at the end of the physical line. All characters after the **#** character up to the end of the line are part of the comment and the Python interpreter ignores them.

Code:

```
# This is a comment
# print out Hello
print('Hello’)  # now print
```

Shell

```
Hello
>>>
```

Python Coding Style:

- Use 4 spaces per indentation and no tabs.
- Do not mix tabs and spaces. Tabs create confusion and it is recommended to use only spaces.
- Maximum line length : 79 characters which help users with a small display.
- Use blank lines to separate top-level function and class definitions and single blank line to separate methods definitions inside a class and larger blocks of code inside functions.
- When possible, put inline comments.
- Use spaces around expressions and statements.

Python Variable

A variable is a named location used to store data in the memory. It is helpful to think of variables as a container that holds data that can be changed later in the program.

```
X=10
```

Python Variable Name Rules:

- Must begin with a letter (a - z, A - B) or underscore (_)
- Other characters can be letters, numbers or _
- Case Sensitive
- Can be any (reasonable) length

pneumonoultramicroscopicsilicovolcanoconiosis (Finnish = type of silicosis*)*

Keywords in Python programming language or reserved words - they cannot be used as identifiers

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

You can always get the list of keywords in your current version by typing the following in the prompt.

```
>>> import keyword
>>> print(keyword.kwlist)

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

Data types

Every value in Python has a datatype. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.

Numbers: int, float and complex

2, 2.0, 1+2j

List: ordered sequence of items

[1, 2.2, 'python’]

Tuple: an ordered sequence of items same as a list, is defined within parentheses (), cannot be modified

(1, 2.2, 'python’)

String: sequence of Unicode characters

"This is a string“

Set: an unordered collection of unique items, is defined within braces { }, indexing has no meaning

{1, 2, 3, 4, 5}

Dictionary: an unordered collection of key-value pairs, defined within braces {} with each item being a pair in the form key:value, - can be of any type

{1:'value','key':2}

use the **type()** function to know which class a variable or a value belongs to.
convert between different data types by using different type conversion functions **like int(), float(), str()**

Time to open Colab