Lecture -2

Introduction to Python and computer programming

Instructor : AALWAHAB DHULFIQAR

Advisor : Dr. Tejfel Mate

python

What you will learn:

how the computer works

how the program is executed

compilation and interpretation

what Python

Python's Versions

what Python literals, operators, and expressions are;

what variables are and what are the rules that govern them;

PYTHON INSTITUTE

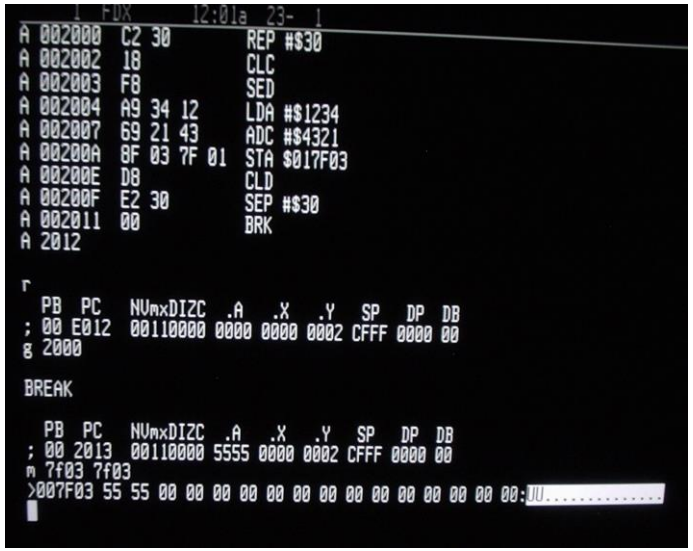
Open Education & Development Group

CISCO™

How the computer works

Find the average speed:

- accept a number representing the distance;
- accept a number representing the travel time;
- divide the former value by the latter and store the result in the memory;
- display the result (representing the average speed) in a readable format.



Natural languages vs. programming languages

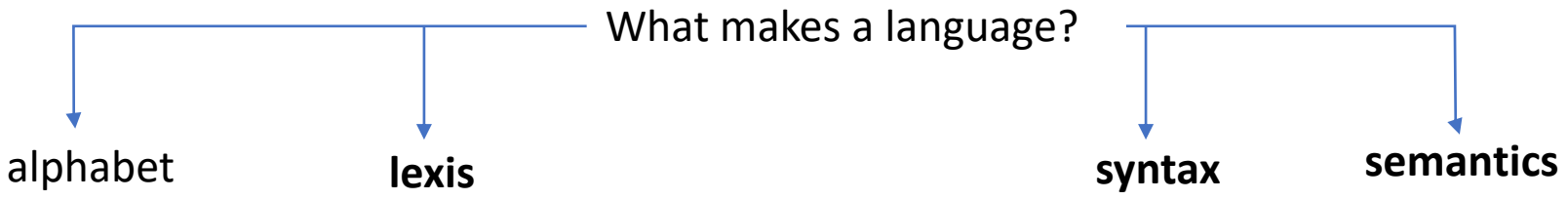
Computers have their own language, too, called machine language, which is very rudimentary.

The commands it recognizes are very simple. We can imagine that the computer responds to orders like "take that number, divide by another and save the result".

A complete set of known commands is called an instruction list, sometimes abbreviated to IL.

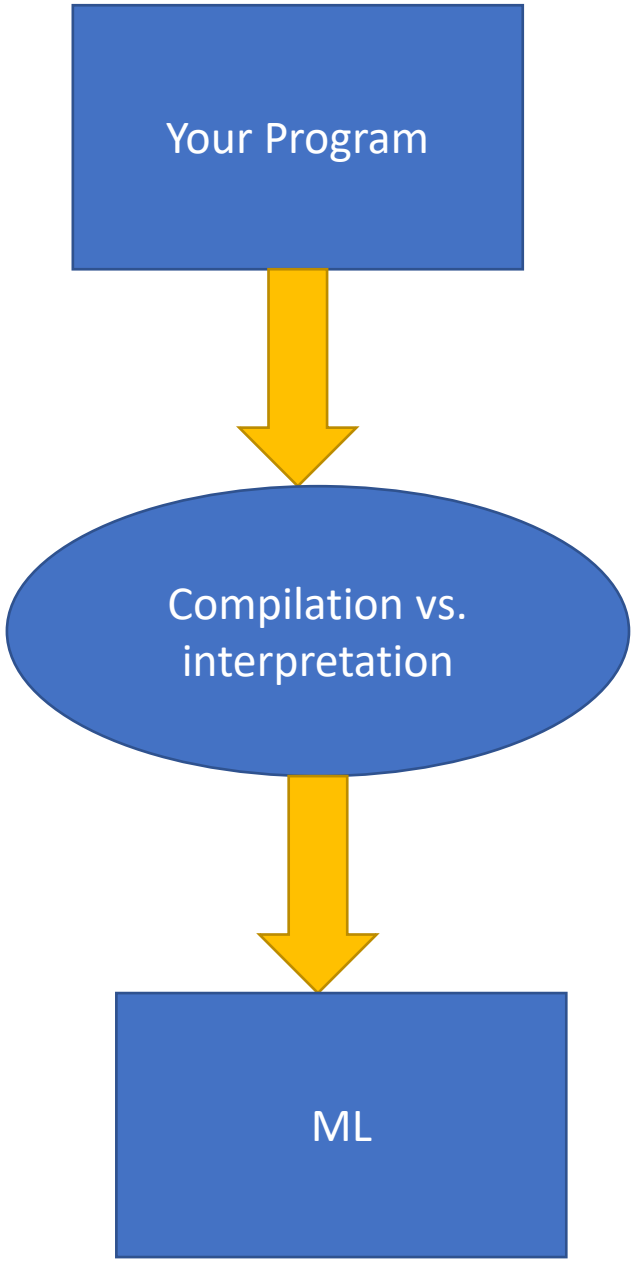
No computer is currently capable of creating a new language.

human languages developed naturally.



A program written in a high-level programming language is called a source code (in contrast to the machine code executed by computers). Similarly, the file containing the source code is called the source file.

Compilation vs. interpretation




```
class StudentsDataException(Exception):
    pass

class BadLine(Exception):
    def __init__(self, line, message=''):
        Exception.__init__(self, message)
        self.line = line

class FileEmpty(Exception):
    def __init__(self, message=''):
        Exception.__init__(self, message)
```

	COMPILATION	INTERPRETATION
ADVANTAGES	•the execution of the translated code is usually faster; •only the user has to have the compiler - the end-user may use the code without it; •the translated code is stored using machine language - as it is very hard to understand it, your own inventions and programming tricks are likely to remain your secret.	•you can run the code as soon as you complete it - there are no additional phases of translation; •the code is stored using programming language, not the machine one - this means that it can be run on computers using different machine languages; you don't compile your code separately for each different architecture.
DISADVANTAGES	•the compilation itself may be a very time-consuming process - you may not be able to run your code immediately after any amendment; •you have to have as many compilers as hardware platforms you want your code to be run on.	•don't expect that interpretation will ramp your code to high speed - your code will share the computer's power with the interpreter, so it can't be really fast; •both you and the end user have to have the interpreter to run your code.

Note: languages designed to be utilized in the interpretation manner are often called scripting languages, while the source programs encoded using them are called scripts.

What is Python?

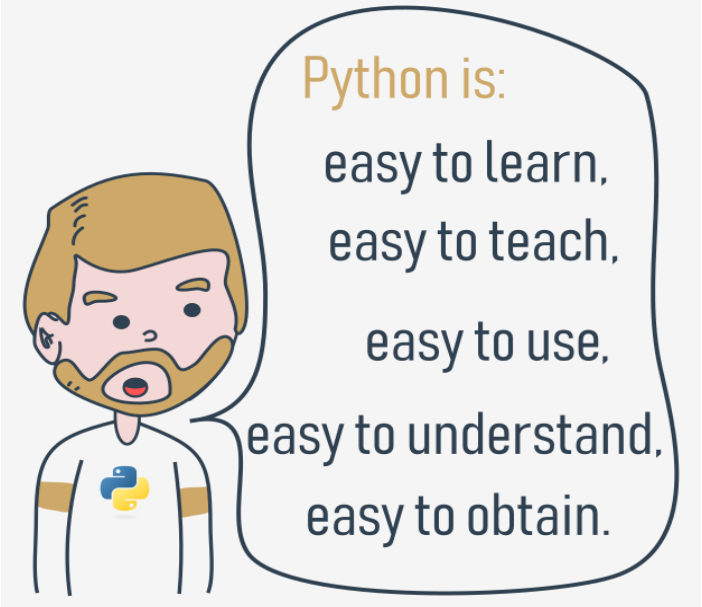
Python was created by Guido van Rossum, born in 1956 in Haarlem, the Netherlands.

In December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office (...) would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus).
Guido van Rossum

Python goals

- an **easy and intuitive** language just as powerful as those of the major competitors;
- **open source**, so anyone can contribute to its development;
- code that is as **understandable** as plain English;
- **suitable for everyday tasks**, allowing for short development times.

What makes Python special?



Python has two direct competitors

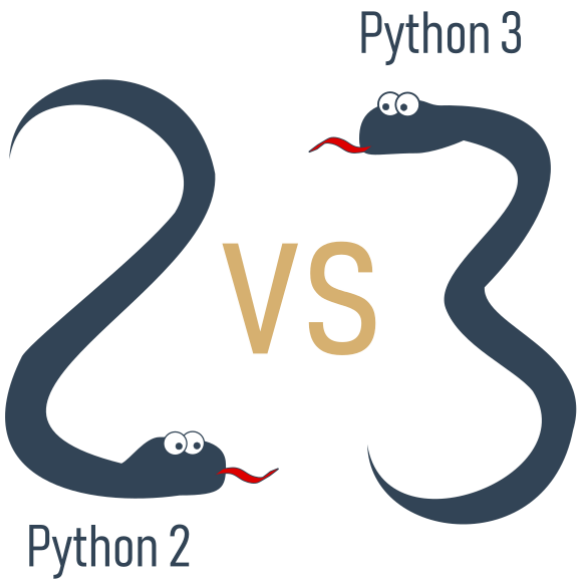
Perl - a scripting language originally authored by Larry Wall;
Ruby - a scripting language originally authored by Yukihiro Matsumoto.

Where can we see Python in action?

Internet services like search engines, cloud storage and tools, social media and so on.

Many **developing tools** are implemented in Python. More and more **everyday use applications** are being written in Python. Lots of **scientists** have abandoned expensive proprietary tools and switched to Python. Lots of IT project **testers** have started using Python to carry out repeatable test procedures.

Python's versions



maintained by the people gathered around the PSF (Python Software Foundation)

https://www.python.org/psf-landing/

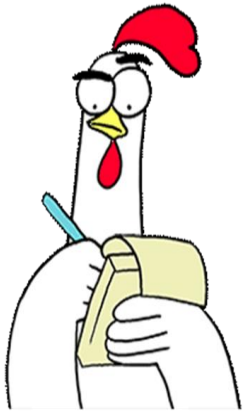
Guido van Rossum used the "C" programming language to implement the very first version of his language and this decision is still in force. All Pythons coming from the PSF are written in the "C" language. This is why the PSF implementation is often referred to as **CPython**.

Jython

pypy

Checkpoint

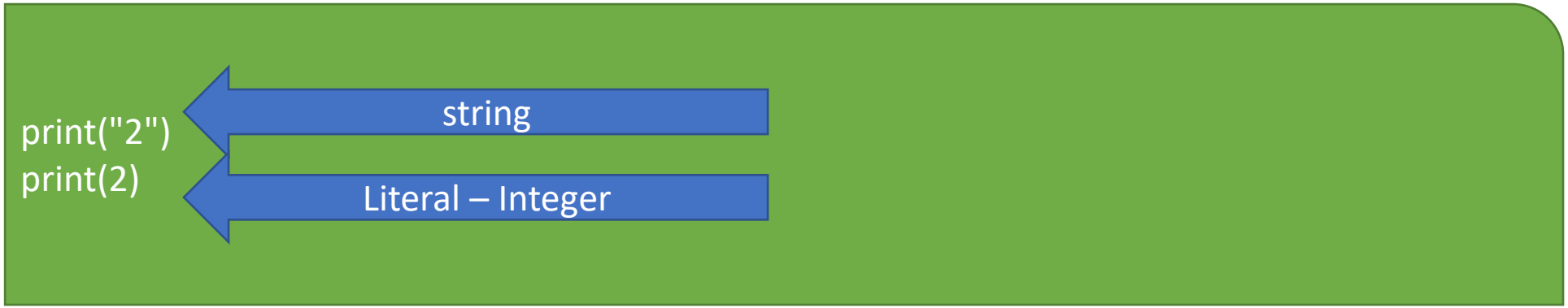
- ✓ the fundamentals of computer programming, i.e., how the computer works, how the program is executed, how the programming language is defined and constructed;
- ✓ the difference between compilation and interpretation;
- ✓ the basic information about Python and how it is positioned among other programming languages, and what distinguishes its different versions;
- ✓ the study resources and different types of interfaces you will be using in the course.



Literals - the data in itself

A literal is data whose values are determined by the literal itself.

Example: 123 is a literal, and c is not.



Integers

English	Python
11,111,111 11.111.111 11 111 111	11111111 11_111_111 +or - 11_111_111

Integers: octal and hexadecimal numbers

If an integer number is preceded by an 0O or 0o prefix (zero-o), it will be treated as an octal value.
0o123 is an octal number with a (decimal) value equal to 83

If 0x or 0X (zero-x). 0x123 is a hexadecimal number with a (decimal) value equal to 291.

Floats

In Python 0.4 .4 4.

When you want to use any numbers that are very large or very small, you can use scientific notation

$300000000 = 3 \times 10^{8}$

Strings

Strings are used when you need to process text (like names of all kinds, addresses, novels, etc.), not numbers.

Example : "I am a string."

How to print this ? -> I like "Monty Python"

print("I like \"Monty Python\"")

print('I like "Monty Python"')

Boolean values

English	Python
Yes, this is true;	True
No, this is false	False

What are variables?

What does every Python variable have?

a name;
a value (the content of the container)

If you want to give a name to a variable, you must follow some strict rules:

1. the name of the variable must be composed of upper-case or lower-case letters, digits, and the character _ (underscore)
2. the name of the variable must begin with a letter;
3. the underscore character is a letter;
4. upper- and lower-case letters are treated as different (a little differently than in the real world - Alice and ALICE are the same first names, but in Python they are two different variable names, and consequently, two different variables);
5. the name of the variable must not be any of Python's reserved words (the keywords - we'll explain more about this soon).

Correct and incorrect variable names

Here are some correct, but not always convenient variable names:

MyVariable, i, t34, Exchange_Rate, counter, days_to_christmas, TheNameIsSoLongThatYouWillMakeMistakesWithIt, _.

These variable names are also correct:

Adiós_Señora, sûr_la_mer, Einbahnstraße, переменная.

incorrect names:

10t (does not begin with a letter), Exchange Rate (contains a space)

Keywords

['False', 'None', 'True', 'and', 'as', 'assert', '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']

Creating variables

```
var = 1
print(var)
```

Note: False Not equals false

Using variables

```
var = 1
account_balance = 1000.0
client_name = 'John Doe'
print(var, account_balance, client_name)
print(var)
```

```
var = "3.8.5"
print("Python version: " + var)
```

Assigning a new value to an already existing variable

```
var = 1
print(var) # OT = 1
var = var + 1
print(var) # OT = 2
```

```
var = 100
var = 200 + 300
print(var )# OT = 500
```

Shortcut operators

x = x * 2 ⇒ x*=2

Sheep = Sheep + 1 ⇒ Sheep+=1

i = i + 2 * j ⇒ i += 2 * j

var = var / 2 ⇒ var /= 2

rem = rem % 10 ⇒ rem %= 10

j = j - (i + var + rem) ⇒ j -= (i + var + rem)

x = x ** 2 ⇒ x **= 2

Leaving comments in code: why, how, and when

You may want to put in a few words addressed not to Python but to humans, usually to explain to other readers of the code how the tricks used in the code work, or the meanings of the variables, and eventually, in order to keep stored information on who the author is and when the program was written.

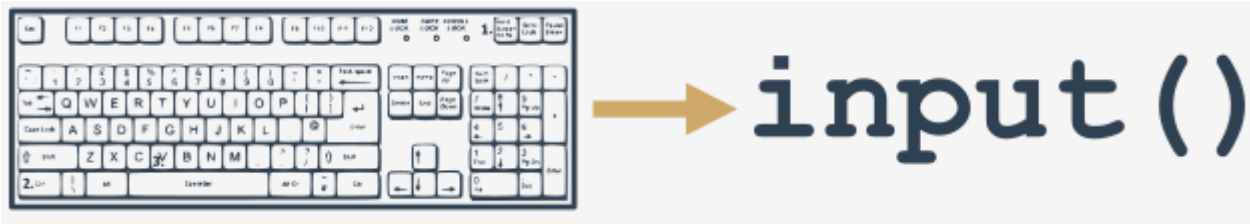
A remark inserted into the program, which is omitted at runtime, is called a comment

```
# This program evaluates the hypotenuse c.
# a and b are the lengths of the legs.
a = 3.0
b = 4.0
c = (a ** 2 + b ** 2) ** 0.5  # We use ** instead of square root.
print("c =", c)
```

The `input()` function

The input() function is able to read data entered by the user and to return the same data to the running program.

```
print("Tell me anything...")
anything = input()
print("Hmm...", anything, "... Really?")
```




```
anything = input("Tell me anything...")
print("Hmm...", anything, "...Really?")
```

The result of the `input()` function

```
anything = input("Enter a number: ")
something = anything ** 2.0
print(anything, "to the power of 2 is", something)
```

TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'float'

Type casting

Python offers two simple functions to specify a type of data and solve this problem - here they are: int() and float().

```
anything = float(input("Enter a number: "))
something = anything ** 2.0
print(anything, "to the power of 2 is", something)
```

```
leg_a = float(input("Input first leg length: "))
leg_b = float(input("Input second leg length: "))
hypo = (leg_a**2 + leg_b**2) ** .5
print("Hypotenuse length is", hypo)
```

String operators - introduction

Concatenation : string + string

```
fnam = input("May I have your first name, please? ")
lnam = input("May I have your last name, please? ")
print("Thank you.")
print("\nYour name is " + fnam + " " + lnam + ".")
```

Replication : string * number
number * string

```
print("+" + 10 * "-" + "+")
print(("|" + " " * 10 + "|\n") * 5, end="")
print("+" + 10 * "-" + "+")
```

See you in the lab session