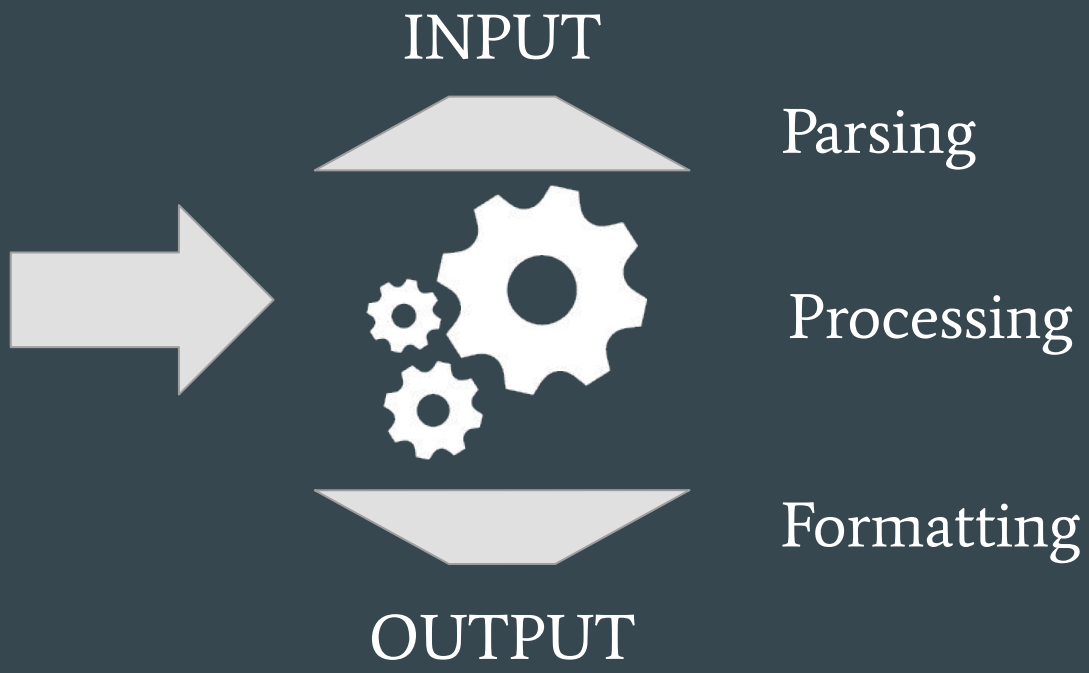
Algorithm in Python

...

Enter the magical land of algorithms

What are we going to code?



Basic notions : a variable

24

“This_is_a_word”

‘c’

{‘a’:2, ’b’:4}

[2,4,9]

True

Java, C, C++...

```
int n = 24;
```

Python

```
n = 24
n = “plop”
```

Basic notions : an instruction

```
A = B + 1

A[2] = 4

list.sort()

...
```

Not all as heavy as the others!

Lean the specific instruction of your languages (hidden functions)

Basic notions : conditions

- Can be evaluated to True/False
- Used inside evaluator like “if”, “while”

```
if condition:
   # code here
```

```
if condition == True:
   # code here
```

```
if condition == 1:
   # code here
```

```
if condition = True:
   # won’t work
```

Basic notions : loops

- Used when a bloc of code has to be evaluated multiple times

```
while condition:
    # code here
```

```
for var in list:
    # code here
```

```
a = 0
while a<5 :
    a += 1
    print(“hey”)
```

```
for k in range(5):
    print(“hey”)
```

Basic notions : functions

- Used to export and reuse a bloc of code
- Improves readability

```
def my_func():
    print(“a”)
my_func()
```

```
def my_func():
    return 0
var = my_func()
```

```
def my_func(a):
    return a + 1
var = my_func(var)

var += 1
```

Basic notions : lists

- Data structure, has limited size

```
my_list = []
my_list.append(5)
print(my_list)
--------
[5]
```

```
my_list = []
my_list.append(5)
my_list.append(“hey”)
print(my_list)
--------
[5,”hey”]
```

```
my_list = [0]*2
my_list[0] = 1
print(my_list)
--------
[0,1]
```

Quick python recap

- Indentation matters, be consistent !
- Use “#” for comments
- End “for”, “if” declarations with “:”
- No “;” at the end of instruction
- Don’t give a type to your variables
- Script language → do not try to compile
- You’re gonna love the garbage collector
- Learn as much about the built in function as possible, everything exists

Thanks, and once more don’t be afraid to ask for help !

Credits

Slides: Arthur Tondereau, INSAlgo