Rappels

Université Toulouse Paul Sabatier - KEAT9AA1

Guilhem Saurel

2025-09-01

HELLO WORLD

Hello world: C++

```cpp
#include <iostream>

auto main() -> int {
  std::cout << "hello\n";
  return 0;
}
```

Hello world: C++, compilation

```
$ g++ hello.cpp && ./a.out
hello
```

Hello world: Python

```python
#!/usr/bin/env python

if __name__ == "__main__":
    print("hello")
```

Hello world: Python, execution

```
$ chmod +x hello.py && ./hello.py
hello
```

Types de données, C++

```
auto ga{3};                     // int
auto bu{3.14};                  // double
const auto *const zo{"tau"}; // const char *const
std::string meu{"pi"};
```

Types de données, Python

```
ga: int = 3
bu: float = 3.14
zo: str = "pi"
```

CONTRÔLE DU FLOT

Functions, C++

```
auto add(int first, int second) -> int {
  return first + second;
}
```

Functions, Python

```python
def add(first: int, second: int) -> int:
    return first + second
```

Conditions: C++

```cpp
if (temperature > 26) {
  std::cout << "Too hot\n";
  turn_cooler_on();
} else if (temperature < 16) {
  std::cout << "Too cold\n";
  turn_heater_on();
} else {
  std::cout << "Lucky people !\n";
}
```

Conditions: Python

```
if temperature > 26:
    print("Too hot")
    turn_cooler_on()
elif temperature < 16:
    print("Too cold")
    turn_heater_on()
else:
    print("Lucky people !")
```

Boucles: while, C++

```cpp
auto user_input{0};
while (user_input != 42) {
  std::cout << "guess: ";
  std::cin >> user_input;
}
```

Boucles: while, Python

```python
user_input: int = 0
while user_input != 42:
    user_input = int(input("guess: "))
```

Boucles: while, Python, :=

```
while (user_input := int(input("guess: "))) != 42:
    print("it's not", user_input)
```

Boucles: break - C++

```cpp
auto user_input{0};
while (true) {
  std::cout << "guess: ";
  std::cin >> user_input;
  if (user_input == 42) {
    std::cout << "Yes !" << "\n";
    break;
  }
  std::cout << "I's not " << user_input << "\n";
}
```

Boucles: break - Python

```python
user_input: int = 0
while True:
    user_input = int(input("guess: "))
    if user_input == 42:
        print("Yes !")
        break
    print("It's not", user_input)
```

Boucles: for / break - C++

```cpp
#include <cstdlib>

for (auto i{0}; i < 10000; i++) {
  std::cout << "iteration " << i << "\n";
  auto rand = static_cast<double>(std::rand());
  if (rand / RAND_MAX > 0.95) {
    break;
  }
}
```

Boucles: for / break - Python

```python
from random import random

for i in range(10000):
    print("iteration", i)
    if random() > 0.95:
        break
```

Boucles: continue, C++

```cpp
for (auto i{0}; i < 10; i++) {
  if (i % 2 == 0) {
    continue;
  }
  std::cout << "iteration " << i << "\n";
}
```

Boucles: continue, Python

```python
for i in range(10):
    if i % 2 == 0:
        continue
    print("iteration", i)
```

Boucles: conteneurs, C++

```
using Colors = std::vector<std::string>;
Colors colors{"orange", "blue", "pink"};
for (const auto &color: colors) {
  std::cout << color << "\n";
}
```

Boucles: conteneurs, Python

```python
colors = ["orange", "blue", "pink"]
for color in colors:
    print(color)
```

OBJETS

C++

```cpp
class Robot {
public:
  auto work() { battery -= 5; }
  auto get_battery() const -> int { return battery; }

protected:
  int battery{100};
};

auto main() -> int {
  auto robot = Robot{};
  std::cout << robot.get_battery() << "% remaining\n";
  robot.work();
  std::cout << robot.get_battery() << "% remaining\n";
  return 0;
}
```

Python

```python
class Robot:
    battery = 100

    def work(self):
        self.battery -= 5

    def get_battery(self) -> int:
        return self.battery


if __name__ == "__main__":
    robot = Robot()
    print(robot.get_battery(), "% remaining")
    robot.work()
    print(robot.get_battery(), "% remaining")
```

Héritage: C++

```cpp
class LeggedRobot : public Robot {
public:
  auto walk() { battery -= 10; }
};

auto main() -> int {
  auto robot = LeggedRobot{};
  std::cout << robot.get_battery() << "% remaining\n";
  robot.work();
  robot.walk();
  std::cout << robot.get_battery() << "% remaining\n";
  return 0;
}
```

Héritage: Python

```
class LeggedRobot(Robot):
    def walk(self):
        self.battery -= 10

if __name__ == "__main__":
    robot = LeggedRobot()
    print(robot.get_battery(), "% remaining")
    robot.work()
    robot.walk()
    print(robot.get_battery(), "% remaining")
```

Constructeurs: C++

```
class Robot {
public:
  Robot() {}
  Robot(int bat) : battery{bat} {}
  auto work() { battery -= 5; }
  auto get_battery() const -> int { return battery; }

protected:
  int battery{100};
};

auto main() -> int {
  auto robot = Robot{};
  auto bigRobot = Robot{350};
  return 0;
```

Constructeurs: Python

```
class Robot:
    battery = 100

    def __init__(self, bat=100):
        self.battery = bat

    def work(self):
        self.battery -= 5

    def get_battery(self) -> int:
        return self.battery


if __name__ == "__main__":
    robot = Robot()
    big_robot = Robot(350)
```

Destructeurs: C++

```
class Robot {
public:
  Robot(int bat) : battery{bat} {}
  ~Robot() { std::cout << "Destruction at " << battery << "\n"; }
  auto work() { battery -= 5; }
  auto get_battery() const -> int { return battery; }

protected:
  int battery{100};
};
```

Destructeurs: Python

Garbage collector

__del__