

Conception Orientée Objet, 101

Université Toulouse Paul Sabatier - KEAR9RA1

2022-08-31

Guilhem Saurel

Available at

[https://homepages.laas.fr/gsaurel/
conception-orientee-objet-101.pdf](https://homepages.laas.fr/gsaurel/conception-orientee-objet-101.pdf)

Under License



<https://creativecommons.org/licenses/by-sa/4.0/>

Source

`https://gitlab.laas.fr/gsaurel/talks :
conception-orientee-objet-101.md`

Discussions

`https://matrix.to/#/#conception-orientee-objet:laas.fr`

Hello world

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

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

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

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

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

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

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

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

C++

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

C++

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

Python

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

Control flow

C++

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

C++

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

Python

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

```
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";  
}
```

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

C++

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

C++

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

Python

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

C++

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

Python

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

Walrus

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

```
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";  
}
```

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

```
#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;
    }
}
```

```
from random import random

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

C++

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

C++

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

Python

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

C++

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

C++

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

Python

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

Objects

```
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;  
}
```

```
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")
```

```
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;  
}
```

```
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")
```