Advanced Topics in Programming Language - Duck Typing in Python

Tommaso Puccetti
Studente presso Universita degli studi di Firenze

April 15, 2019

Contents

List of Tables

List of Figures

1 Possible references

- You Tube video;
- Python duck typing (or automatic interfaces)- how change the depen-decy injection;
- Simple example;
- Something more technical;
- Ultimate guide to Python's type checking;

2 Type checking: recall

Type checking is the process of verifying and enforces the typing rules of a language. In other words the **type checker** (the type checking algorithm of the language) is used to prove the **type safety** of a program.
It may occur either at:

1. compile time (**Static**);
2. run time (**Dynamic**).

Let's see in details what's the difference:

- **Static type checking:** is the process of verifying the type safety of a program based on the analysis of a program text. If a program passes a static type checker, then the program is guaranteed to satisfy some set of type safety properties for all possible inputs.
- **Dynamic type checking:** is the process of verifying the type safety of a program at runtime. It may cause a program to fail at runtime.

There are also two different ways to classify the type check:

- **Explicitly typed:** each variables is annotaded in source code with type's information. In this case the *type check is simple but the language is more difficult* (from the programmers point of view).
- **Implicitly typed:** the data types of source code are automatically detected. It is also rederred as **type inference**. The language *is easier but the type check algorithm is far more complex.*

2.1 Example: type inference vs. dynamic typing

These two kind of typings could be confused. Here an example to clarify the differences:

```
var1 = 10
var2 = "astring"
var3 = var1 + var2
```

1. In **dynamically typed** language this code run without errors: at runtime the *var1* is forced to be a string and the result is *"10astring"*;
2. By the other side, in **inferred type language** the compiler *throw an error*.