Object Oriented Programming 101
From Zero to Hero

Who am I ?

Radu Stoenescu
radu.s.toe@gmail.com

What Am I ?

Teaching Assistant
Object Oriented Programming
UPB

Too much freedom

enforcing usage rules is difficult

```
int undo_my_teacher(person_p teacher) {
    teacher->age = -1;
    return;
}
```

Consequences

1. Difficult to maintain large projects
2. Robust code reads due to the need to extensive documentation.

Summary

Procedural programming is about creating data and modifying it via functions (or procedures)!

Motivation

How To Stay

History

once upon a time there was procedural programming

We had data

```
typedef struct {
    char* name;
    int age;
} person_t, *person_p;
```

and means of manipulating it

```
int make_baby(person_p target) {
    target->age = 0;
    return;
}
```

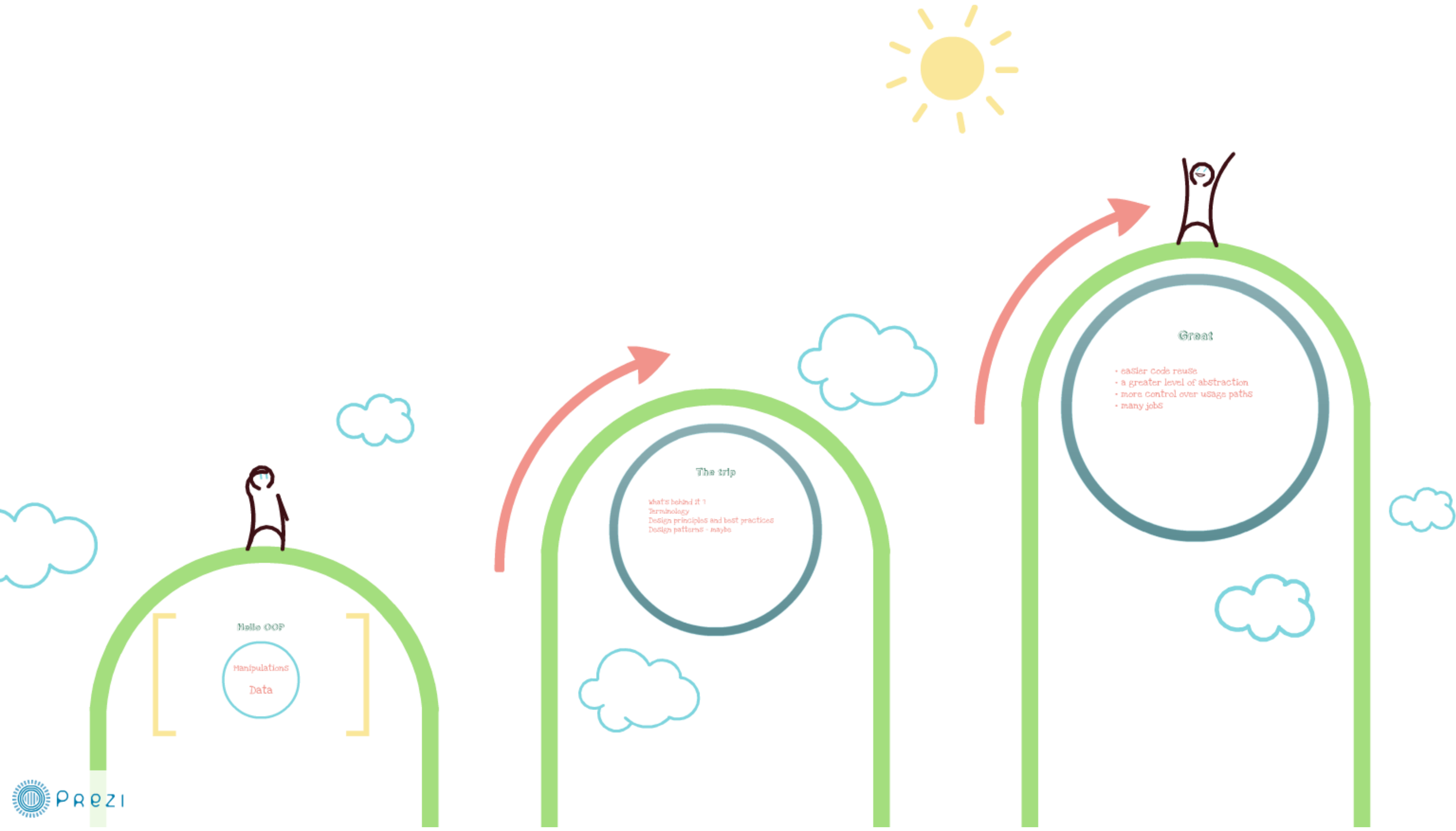
but they were separated

Problems ?

```
if (no problems) {
    exit(COMMON_BUGS);
}
```

Many thanks !

Object Oriented Programming 101
From Zero to Hero



Hello OOP

Manipulations

Data

The trip

What's behind it ?
Terminology
Design principles and best practices
Design patterns - maybe

Great

- easier code reuse
- a greater level of abstraction
- more control over usage paths
- many jobs

Who am I ?

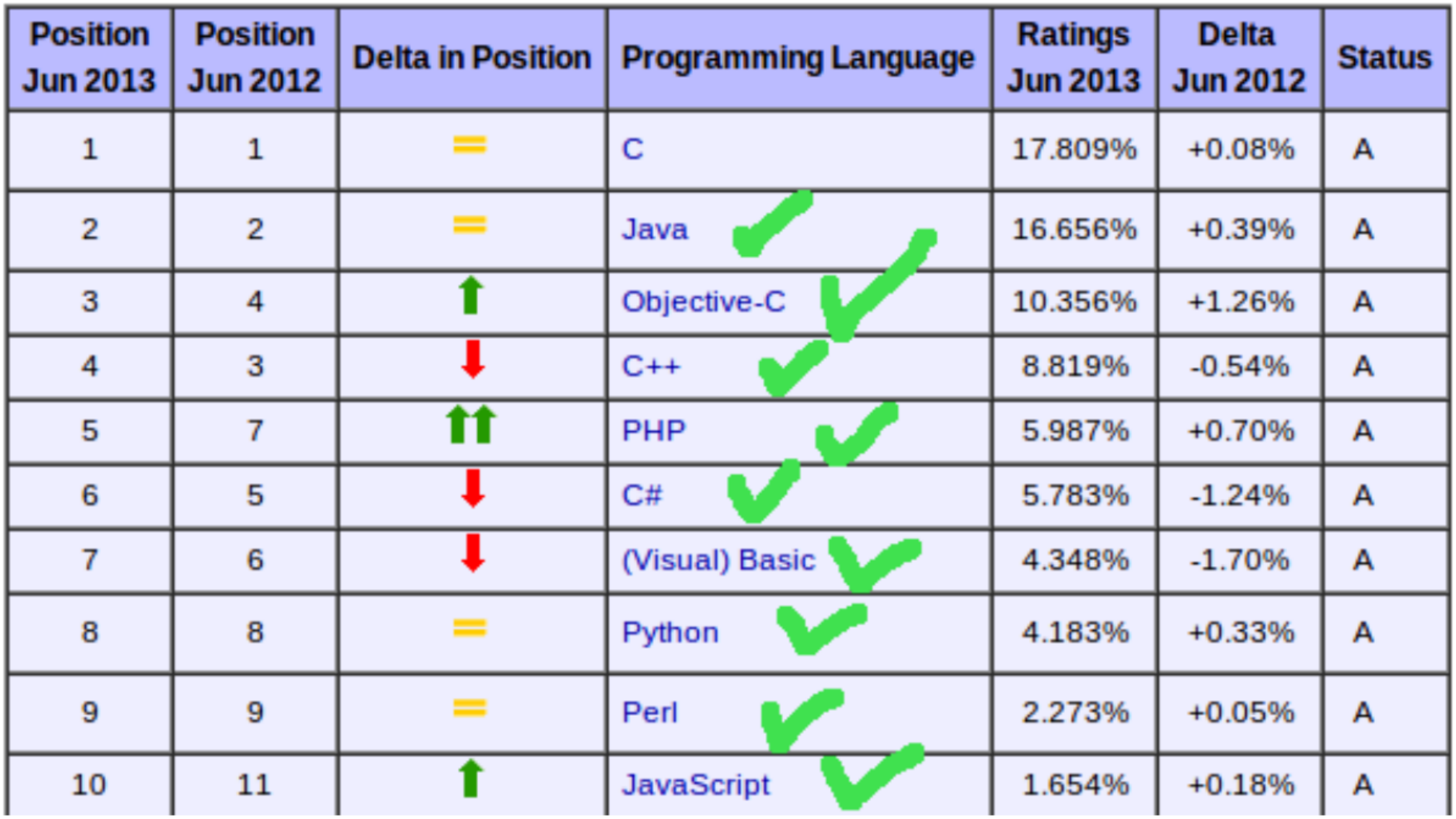
Radu Stoenescu
radu.s.toe@gmail.com

What Am I ?

Teaching Assistant
Object Oriented Programming
UPB

Motivation

Here to Stay



Position Jun 2013	Position Jun 2012	Delta in Position	Programming Language	Ratings Jun 2013	Delta Jun 2012	Status
1	1	=	C	17.809%	+0.08%	A
2	2	=	Java ✓	16.656%	+0.39%	A
3	4	⬆	Objective-C ✓	10.356%	+1.26%	A
4	3	⬇	C++ ✓	8.819%	-0.54%	A
5	7	⬆⬆	PHP ✓	5.987%	+0.70%	A
6	5	⬇	C# ✓	5.783%	-1.24%	A
7	6	⬇	(Visual) Basic ✓	4.348%	-1.70%	A
8	8	=	Python ✓	4.183%	+0.33%	A
9	9	=	Perl ✓	2.273%	+0.05%	A
10	11	⬆	JavaScript ✓	1.654%	+0.18%	A

History

once upon a time
there was procedural
programming

We had data

```
typedef struct {
    char* name;
    int age;
} person_t, *person_p;
```

and means of manipulating it

```
int make_baby(person_p target) {
	target->age = 0;
	return;
}
```

but they were separated

functions | data

Problems ?

```
if (no problems) {
      exit(COMMON_GUYS);
}
```

Too much freedom

enforcing usage rules
is difficult

```
int undo_my_teacher(person_p teacher) {
      teacher->age = -1;
      return;
}
```

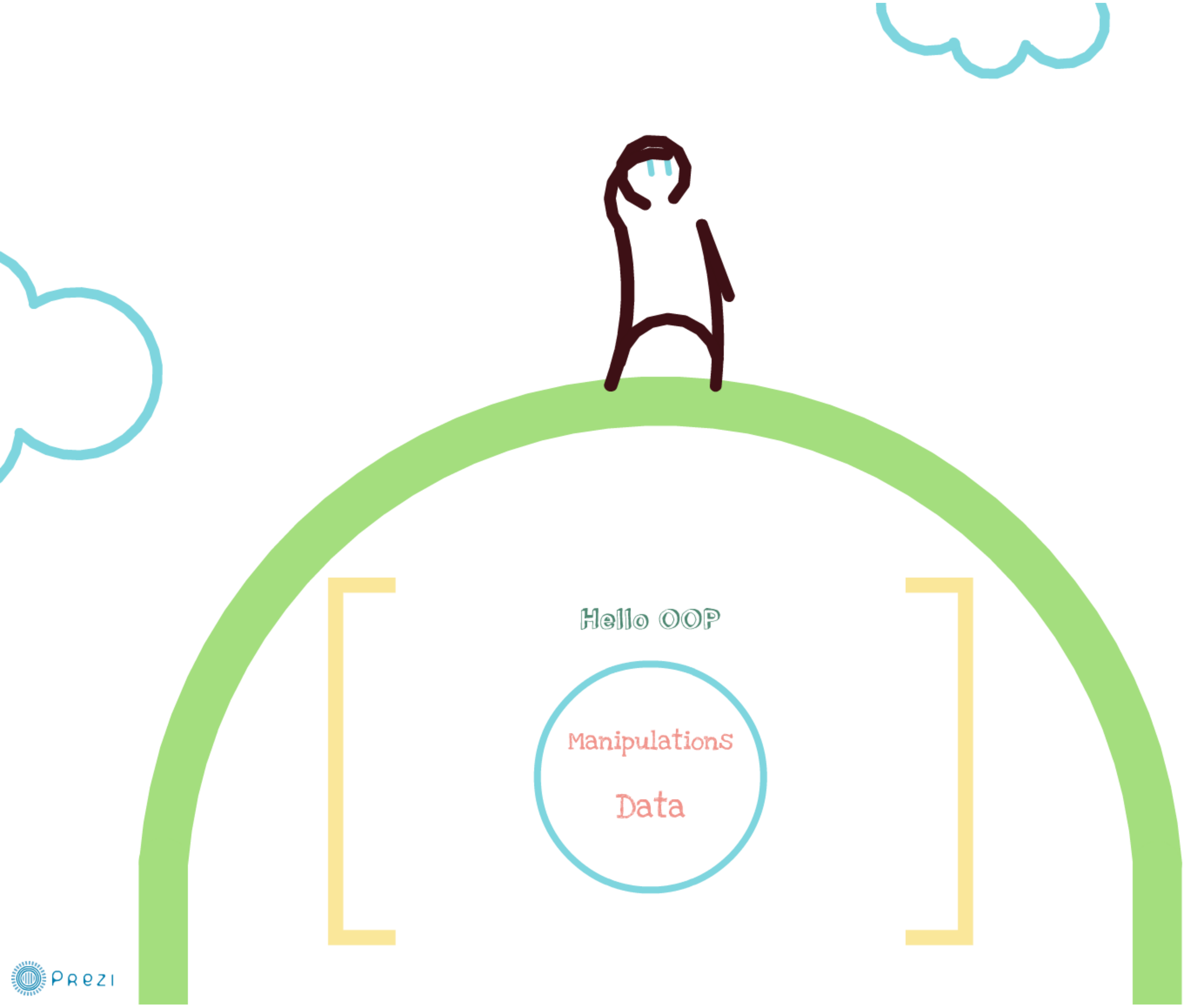
Consequences

1. Difficult to maintain large projects
2. Tedious code reuse due to the need to extensive documentation.

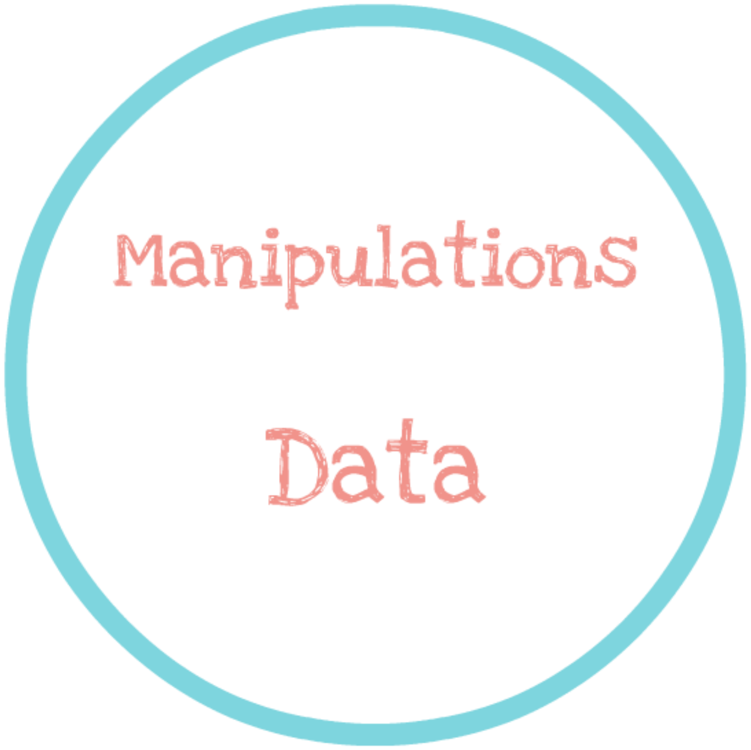
Summary

Procedural programming is about creating data and modifying it via functions (or procedures)

Hello OOP



Hello OOP



The trip

What's behind it ?
Terminology
Design principles and best practices
Design patterns - maybe

The trip

What's behind it ?
Terminology
Design principles and best practices
Design patterns - maybe

What is it ?

A programming paradigm that represents computation as a series of interactions between instances of classes.

Yeah ... right .. everything's clear now

1. Every participating entity is an instance of a class.
2. A class is a blueprint of an instance (object)
3. A class brings together:
   - data (attributes, instance variables, state)
   - functions (computation, methods, behavior)
4. An instance shares methods but NOT data.
5. An instance get life via a special method (constructor) that initializes data.
6. An interaction (message passing between objects) is represented by a method invocation.

Class

constructors
data
methods

Encapsulation

- hiding implementation (black-boxing)
- every piece of data is hidden
- setters/getters

Exposing an API

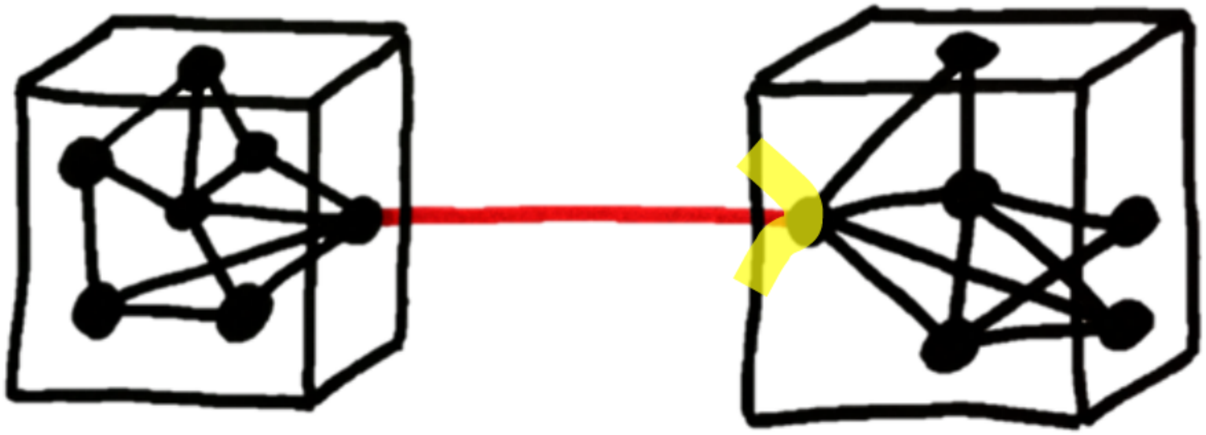
- things you expect others should use
- good enough code

Terminology alert:
a method caller = client
interface ~ API

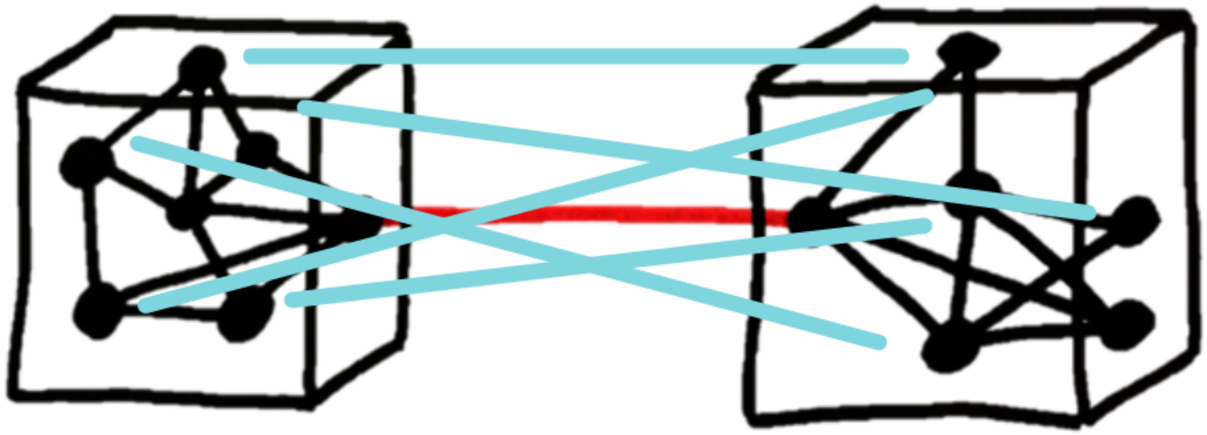
Tight cohesion

- a class should do one thing and one thing only
- favors code reuse★

Loose coupling



vs.



code reuse

what is it ?

java

inheritance
composition and delegation
generics

inheritance

- a new class that does what its ancestor does and something extra

- establishes an "is a" relationship between two classes

composition and delegation

- a class uses one or more of its members to implement a certain behavior
- establishes a 'has a' relationship between two classes

Which is better ?

Inheritance

- behavior proliferation
- statically bound behavior
- only one superclass
- prone to problems caused by superclass interface changes

composition

- verbosity
- performance penalty
- can't benefit from polymorphism

how to decide ?

- does it pass the 'is-a' test ?
- does it adhere to Liskov's substitution principle ?
- is this a case where polymorphism is desired ?

Sorry what ?

Liskov's substitution principle

It states that, in a computer program, if S is a subtype of T, then objects of type T may be replaced with objects of type S (i.e., objects of type S may be substituted for objects of type T) without altering any of the desirable properties of that program (correctness, task performed, etc.).

via Wikipedia

Can a subclass
substitute an instance
of the superclass ?

Polimorphism

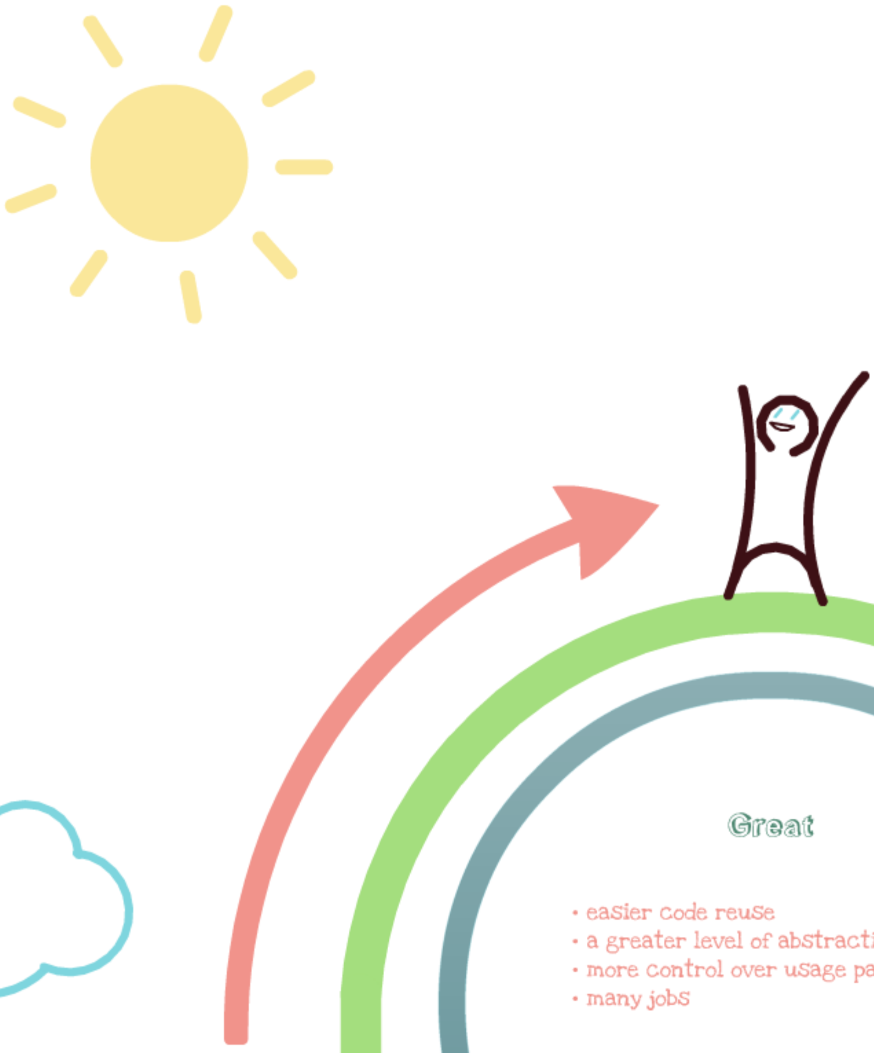
Ability of a class A to act as an instance of a superclass.

```
Weapon wpn = new BFG();
```

But why ?

facilitates behavior variance

```
DBConnection myConn = new OracleConnection();
vs
OracleConnection myConn = new OracleConnection();
```



Great

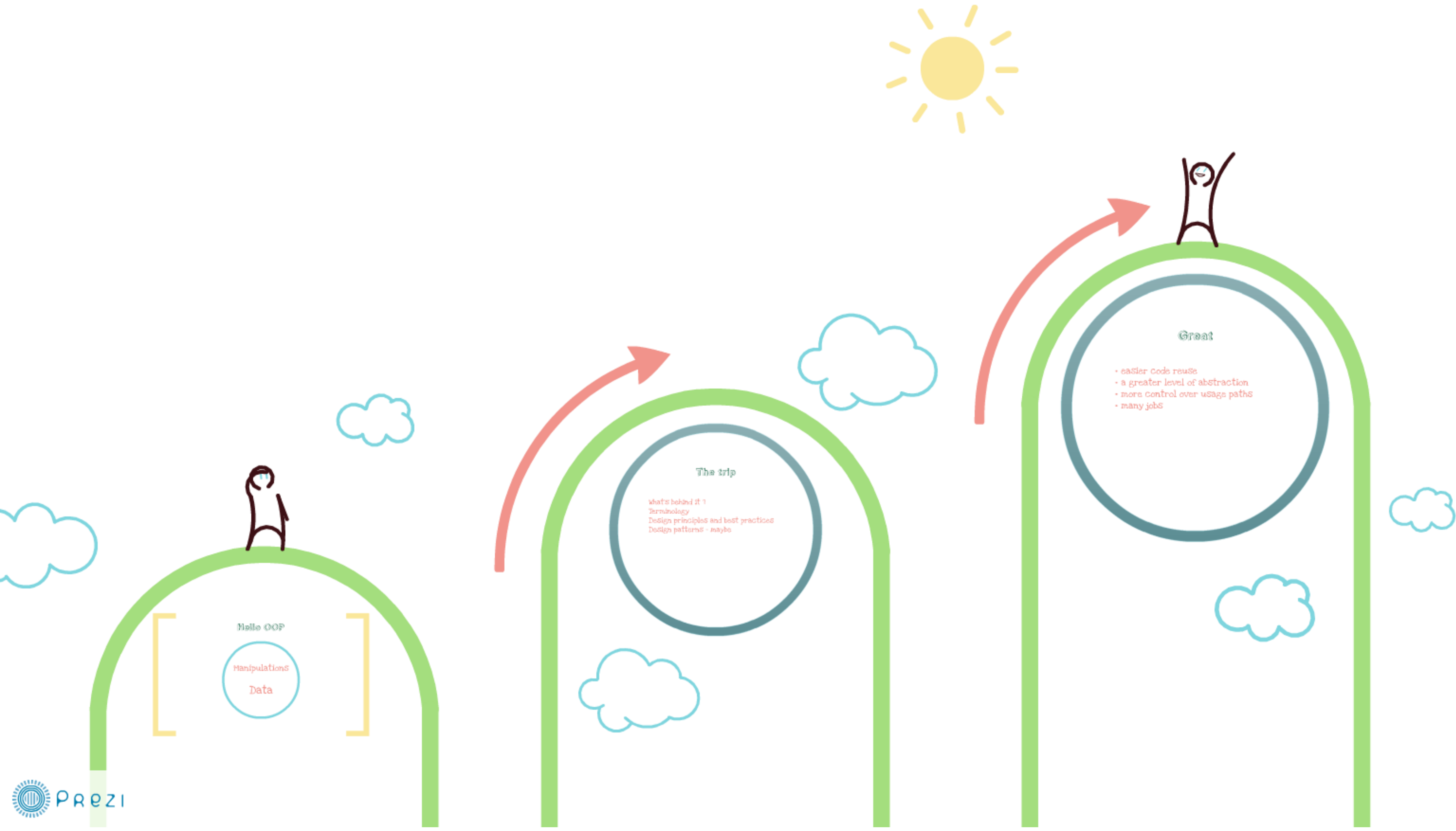
- easier code reuse
- a greater level of abstraction
- more control over usage paths
- many jobs

st practices

Great

- easier code reuse
- a greater level of abstraction
- more control over usage paths
- many jobs

Object Oriented Programming 101
From Zero to Hero



Hello OOP

Manipulations

Data

The trip

What's behind it ?
Terminology
Design principles and best practices
Design patterns - maybe

Great

- easier code reuse
- a greater level of abstraction
- more control over usage paths
- many jobs

Many thanks !