Module 12

"Dynamic Types"

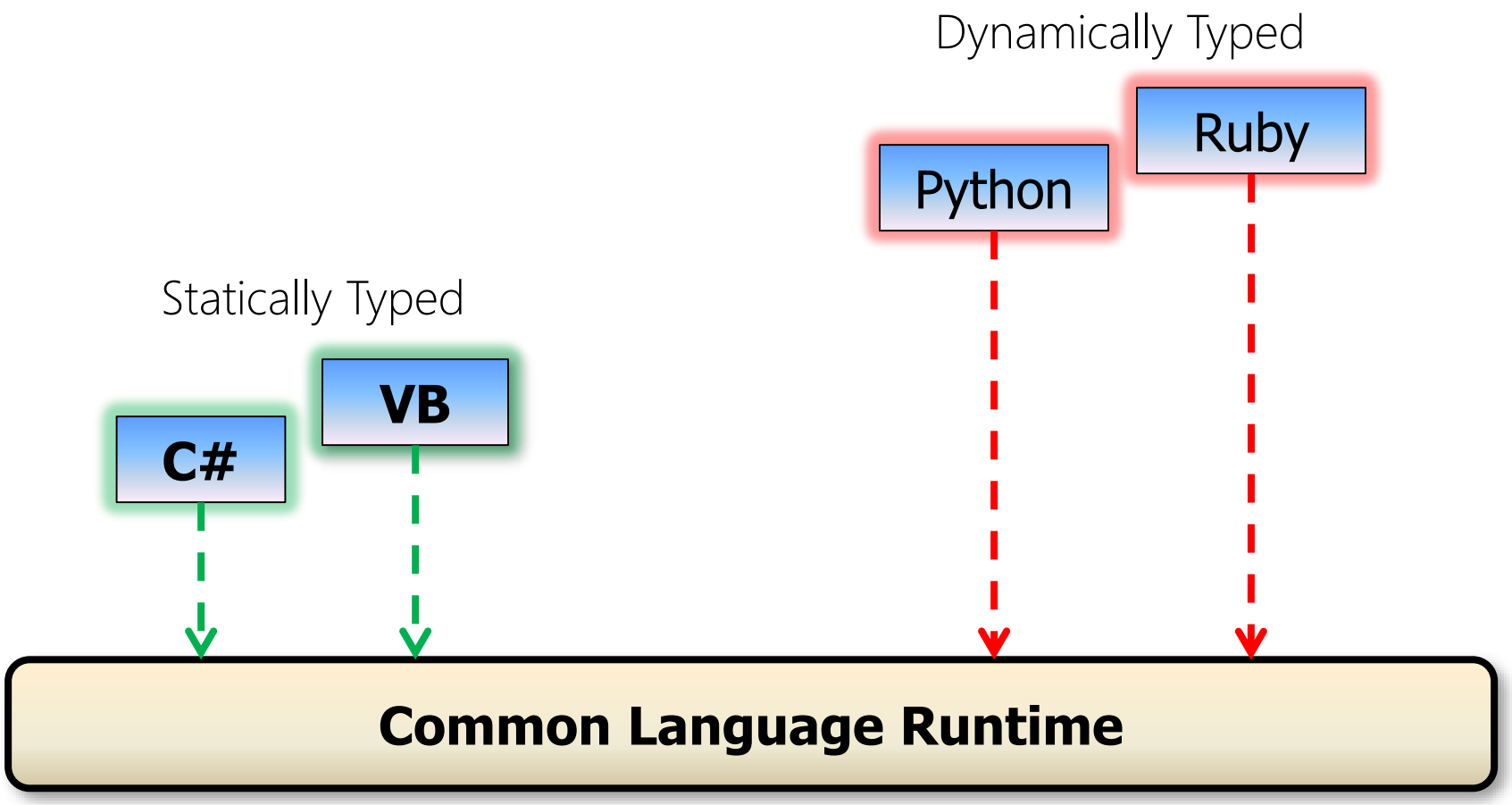
TEKNOLOGISK INSTITUT

Agenda

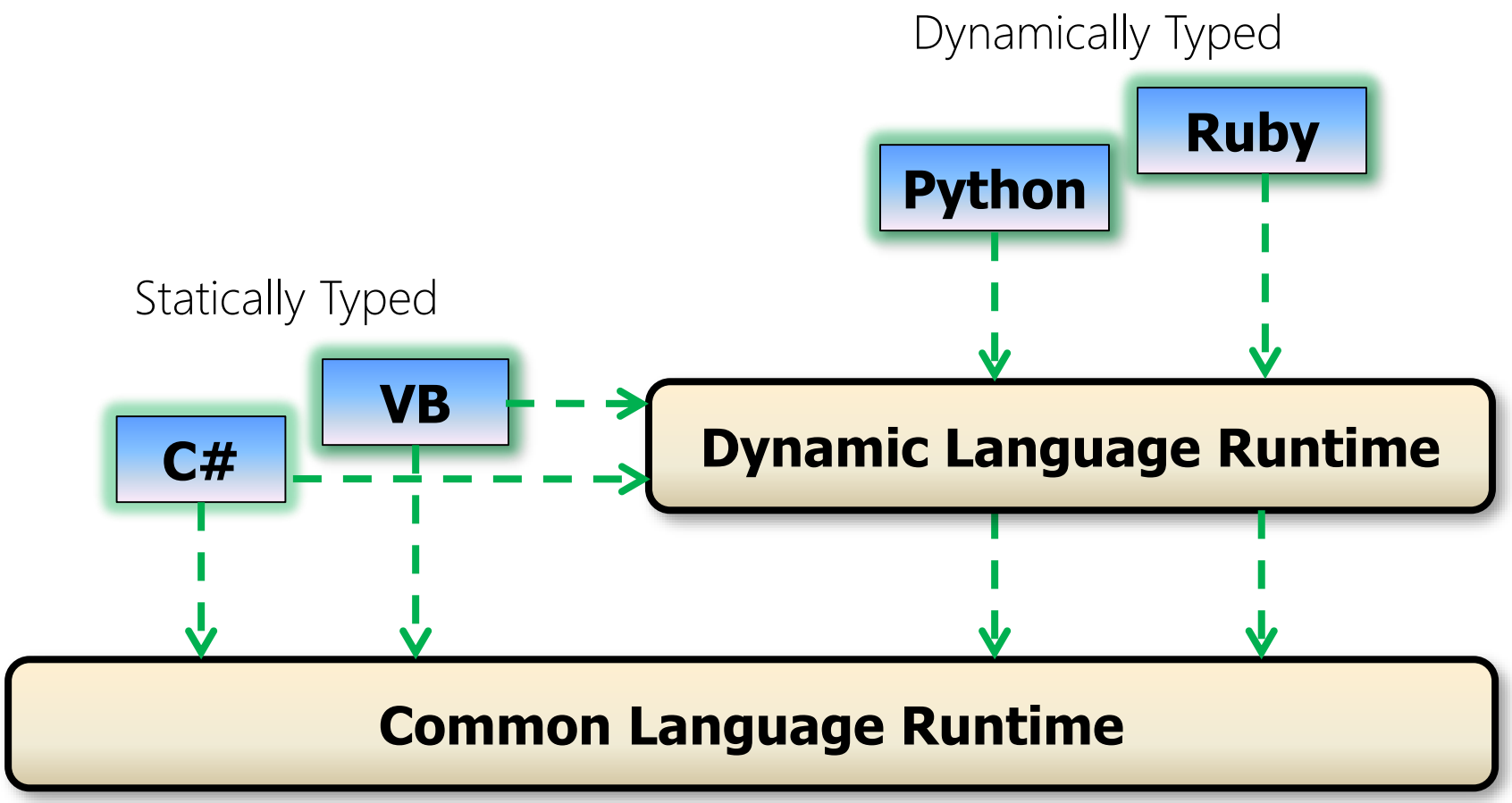
- Introducing Dynamic Types
- The `System.Dynamic` Namespace

TEKNOLOGISK INSTITUT

Dynamic Languages vs. the CLR

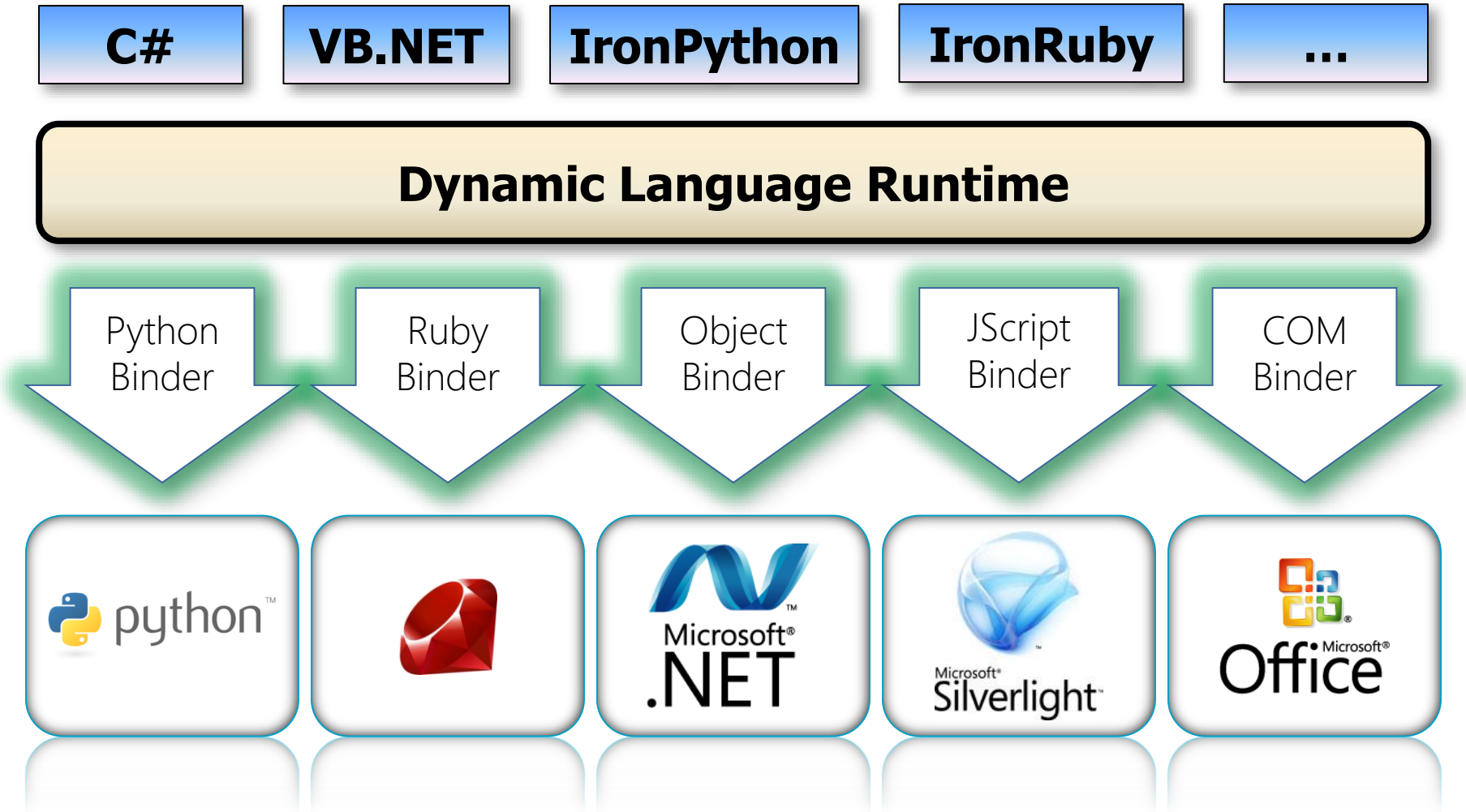


Dynamic Language Runtime (DLR)



.NET Dynamic Programming

TEKNOLOGISK INSTITUT



The `dynamic` Keyword

TEKNOLOGISK INSTITUT

- Dynamic types are facilitated through the dynamic keyword

```
dynamic calculator = new Calculator();
int result = calculator.Add( 42, 87 );

Console.WriteLine( result );
```

- "**`dynamic`**" is <u>not</u> a specific new type such as **`object`**, **`string`**, **`int`**, …
  - It just means "dynamically typed"
  - Checks at *runtime* instead of *compile-time*
  - No IntelliSense in Visual Studio 2012

- Dynamic data is <u>not</u> statically typed!

dynamic vs. var

- var
  - Lets the compiler figure out the type
  - Once determined by initialization, the type never changes
  - Statically typed

```
var i = 87;
i = 42;
i = "Does this compile? No!";
```

- dynamic
  - Compiler performs no type check
    - Type may change
  - Lets the runtime figure out the type when invoking
  - Dynamically typed

```
dynamic d = 87;
d = "Does this compile? Yes!";
d.DoesThisCompile();
```

Usage of Dynamic Types

TEKNOLOGISK INSTITUT

- Use dynamic typing for
  - Interoperating with dynamic languages such as Python or Ruby
  - Interoperating with COM, e.g. Office, Speech, …
- Use **`dynamic`** keyword to allow for the dynamic behavior

```
dynamic d = new MyDynamicClass();
d.SomeDynamicMethod( 42, 87 );

MyDynamicClass v = new MyDynamicClass();
v.SomeDynamicMethod();
```

- Note: Otherwise use static typing as usual…!

- The **`dynamic`** keyword cannot be used with
  - Lambda expressions and LINQ
  - `try-catch-finally`

Agenda

- Introducing Dynamic Types
- The **`System.Dynamic`** Namespace

Creating Types with Dynamic Behavior

TEKNOLOGISK INSTITUT

- There is no "`class dynamic`" keywords ☺

- To create a C# type with dynamic behavior you must implement the `IDynamicMetadataProvider` interface
  - In `System.Dynamic` namespace
  - Hard and burdensome to implement…
- Alternatively, you can derive your class from `DynamicObject`

```
public class MyDynamicClass : DynamicObject
{
    public override bool TryInvoke(
        InvokeMemberBinder binder,
         object[] args,
         out object result )
    {
         ...
    }
}
```

The ExpandoObject

TEKNOLOGISK INSTITUT

- .NET 4.5 ships with a built-in dynamic object in **System.Dynamic** called the **ExpandoObject**
  - Members can be added and removed at runtime

```
dynamic contact = new ExpandoObject();
contact.FirstName = "Jesper";
contact.LastName = "Gulmann";

Console.WriteLine( contact.FirstName + " " +
                   contact.LastName );
```

- Excellent for creating wrapper classes to e.g. XML and JavaScript etc.

- Not described in the book. See:
  - http://blogs.msdn.com/b/csharpfaq/archive/2009/10/01/dynamic-in-c-4-0-introducing-the-expandoobject.aspx

Quiz: Dynamic – Compile-time or Runtime?

TEKNOLOGISK INSTITUT

```
var s = 87;
s = "I see dead code";
Console.WriteLine( s.Length );
```

```
dynamic s = 87;
s = "I see dead code";
Console.WriteLine( s.Length );
```

```
dynamic s = 87;
s = new Car();
Console.WriteLine( s.Length );
```

```
dynamic genius = new ExpandoObject();
genius.Name = "Anders Hejlsberg";
genius.IQ = 220;
Console.WriteLine( genius.Name );
```

Summary

- Introducing Dynamic Types
- The `System.Dynamic` Namespace

Question

TEKNOLOGISK INSTITUT

- You are creating an application with C# 5.0. You need to initialize a dynamic object such that you can add properties to the object at runtime.

  Which code segment should you use?

a) `object exp = new ExpandoObject();`

b) `var exp = new ExpandoObject();`

c) `dynamic exp = new ExpandoObject();`

d) `ExpandoObject exp = new ExpandoObject();`

TEKNOLOGISK
INSTITUT