

# Introduction

The INFDEV team

Hogeschool Rotterdam  
Rotterdam, Netherlands

# Introduction

## Lecture topics

- Intro to DEV3
- What have we learned so far?
- Basic notions of types and declarations
- Introduction to Java and C# with execution examples

# Introduction to DEV3

## Take pride in what you do

- The hardest part is over
- You have now really begun with learning to program
- We are proud of you and your results so far
- Remember to enjoy how much you are learning

## Exam

- written exam
- 4 open questions
- code, type system, and semantics
- no grade: go (score  $\geq 75$ ) or no go (otherwise)

## Homework

- homework to prepare step-by-step
- builds up to actual practicum
- there is no grade for this

## Practicum assignments

- a connected series of programming tasks
- build a simulation similar to that of DEV2
- use the additional structure and help offered by static typing and object orientation
- **mandatory**, but with no direct grade



## Oral

- the oral is entirely based on the practicum assignments
- we remove some pieces of code from the working solutions and you fill them back in
- the oral gives you the final grade for the course

## Expected study effort

- between 10 and 20 **net**<sup>a</sup> hours a week
- read every term on the slides and every sample
- if you do not understand it perfectly, either ask a teacher, google, or brainstorm with other students
- every sample of code on the slides you should both **understand** and **try out** on your machine

---

<sup>a</sup>No, 9gag does not count even if the slides are open on another monitor

# What have we learnt so far?

# What have we learnt so far?

Introduction

The INFDEV  
team

## Python in a nutshell

- How do **all** programming languages work underneath: PC, stack, and heap
- Basic code constructs: variables, conditionals, loops, primitive data types
- Customizable abstractions: functions, recursive functions, classes, methods

# Modern, object-oriented programming languages

## Introduction and motivation

- We will use Java and C#
- They are extremely similar in philosophy, syntax, type system, and semantics
- Each one apart is somewhat limited
- Together they cover a huge chunk of theory and practical applications

# Modern, object-oriented programming languages

Introduction

The INFDEV  
team

## Java

- Dominantly used in businesses
- Extremely Immense ecosystem of tools and libraries
- Great support on most platforms
- A large community means dozens of libraries for most common tasks

## C#

- Dominant in semi-high performance applications (games, simulations)
- Extremely clean and careful design of libraries and advanced language constructs
- Good support on most platform

# Modern, object-oriented programming languages

Introduction

The INFDEV  
team

## Java

- Slow to evolve, because of input from developers
- Less clean design with lots of historical corner cases

## C#

- Less adopted outside the Microsoft world, though Mono and .Net Core are helping
- Historical bad perception of the whole company polluted language reputation
- No immense collection of competing libraries and build systems



## Practicum and assignments

- Just choose whatever you like the most
- Both languages and all supported libraries are accepted
- Moreover, the differences between the two are minimal: learn one, but be aware that you are also learning the other
- We will point the differences out whenever needed

# From Python to Java/C#

## Where does the program go?

- In Python you can just begin writing code anywhere in a file
- This will not be true anymore in Java/C#
- Separate snippets of code cannot be just pasted in an empty file and tried out

## Where does the program go?

All snippets of Java and C# that we will see now cannot (until we see the Main) just be pasted in an empty file and run like we did for Python!!!

# From Python to Java/C#

Introduction

The INFDEV  
team

- Most basic Python constructs translate almost directly to Java/C#
- Lines and instructions always end with a semicolon (;)
- Variables are always declared before use, specifying their type.

```
1 x = (10 + 20)
```

The above Python becomes, in C#:

```
1 int x;  
2 x = (10 + 20);
```

or, alternatively:

```
1 int x = (10 + 20);
```

Which in Java then becomes:

```
1  int x;  
2  x = (10 + 20);
```

Which in Java then becomes:

```
1 int x = (10 + 20);
```

## Basic differences

This snippet (remember: we cannot just copy and paste it) produces the same execution in both Python and Java/C#!



# From Python to Java/C#

Introduction

The INFDEV  
team

```
1 int x = (10 + 20);
```

Stack:

PC
1

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1 int x = (10 + 20);
```

Stack:

PC	x
2	30

# From Python to Java/C#

Introduction

The INFDEV  
team

Java/C# support similar sets of primitive data types

- integers in various sizes: byte, short, int, long, and many others
- floats in various sizes: float and double
- strings: string

These types are richer than Python, because we can specify their size, and thus precision, instead of the one-size-fits-all solution of Python

# From Python to Java/C#

Introduction

The INFDEV  
team

Each primitive data type has a different range and uses more or less memory

- byte is 1 byte, and it goes from -128 to 127
- short is 2 bytes, and it goes from -32,768 to 32,767
- int is 4 bytes, and it goes from  $-2^{31}$  to  $2^{31} - 1$
- float is 4 bytes, and it has a very wide range **with non-uniform steps between adjacent values!**...

Some bugs may depend on attempts to write beyond the range or at a higher precision than supported by the type.

# From Python to Java/C#

Introduction

The INFDEV  
team

- Python operators translate almost directly to Java/C#
- Only exception are the logical operators
- not becomes (!), or becomes (|||), and becomes (&&)

```
1 b = (((10 + 20) / 2) > 5)
```

The above Python becomes, in C#:

```
1 bool b = (((10 + 20) / 2) > 5);
```

Which in Java then becomes:

```
1 bool b = (((10 + 20) / 2) > 5);
```

## Operators and expressions

This snippet (remember: we cannot just copy and paste it) produces the same execution in both Python and Java/C#!

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1 bool b = (((10 + 20) / 2) > 5);
```

Stack:

PC
1



# From Python to Java/C#

Introduction

The INFDEV  
team

```
1 bool b = (((10 + 20) / 2) > 5);
```

Stack:

PC	b
2	true

# From Python to Java/C#

Introduction

The INFDEV  
team

- Python function calls translate directly to Java/C#
- Only difference is, again, the semicolon
- Behaviour remains precisely the same

```
1 print(int(input()))
```

The above Python becomes, in C#:

```
1 Console.WriteLine(Int32.Parse(Console.ReadLine()))
```

Which in Java then becomes:

```
1 System.out.println(Integer.parseInt(new Scanner(System.in).nextLine()))
```

## Function calls

This snippet (remember: we cannot just copy and paste it) produces the same execution in both Python and Java/C#!

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1 Console.WriteLine(Int32.Parse(Console.ReadLine()))
```

Stack:

PC
1

Input:

100
-----

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1 Console.WriteLine(Int32.Parse(Console.ReadLine()))
```

Stack:

PC
1

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1 Console.WriteLine(Int32.Parse(Console.ReadLine()))
```

Stack:

PC
2

Output: 100

## Control flow statements

- Java and C# are curly-bracket languages
- This means that any block of code must now appear between curly brackets { and }
- There are no more colons (:) to delimit declarations
- Indentation remains important for the reader<sup>a</sup>, but the languages do not care
- Programs in Java/C# tend to be longer in part because of this

---

<sup>a</sup>And the student aiming for a passing grade!



# From Python to Java/C#

Introduction

The INFDEV  
team

- Python statements translate almost directly to Java/C#
- Only difference are the brackets and the lack of semicolon
- Behaviour remains precisely the same

```
1 x = int(input())
2 if (x > 0):
3     print("greater")
4 else:
5     print("smaller_or_equal")
```

The above Python becomes, in C#:

```
1 int x = Int32.Parse(Console.ReadLine());
2 if((x > 0)) {
3     Console.WriteLine("greater");
4 } else {
5     Console.WriteLine("smaller_or_equal");
6 }
```

Which in Java then becomes:

```
1  int x = Integer.parseInt(Console.ReadLine());  
2  if (x > 0) {  
3      Console.WriteLine("greater");  
4  } else {  
5      Console.WriteLine("smaller_or_equal");  
6  }
```

## Control flow statements

This snippet (remember: we cannot just copy and paste it) produces the same execution in both Python and Java/C#!

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  if((x > 0)) {  
3      Console.WriteLine("greater")  
4  } else {  
5      Console.WriteLine("smaller_or_equal")  
6  }
```

Stack:

PC
1

Input:

100

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  if((x > 0)) {  
3      Console.WriteLine("greater")  
4  } else {  
5      Console.WriteLine("smaller_or_equal")  
6  }
```

Stack:

PC
1

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  if((x > 0)) {  
3      Console.WriteLine("greater")  
4  } else {  
5      Console.WriteLine("smaller_or_equal")  
6  }
```

Stack:

PC	x
2	100

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  if((x > 0)) {  
3      Console.WriteLine("greater")  
4  } else {  
5      Console.WriteLine("smaller_or_equal")  
6  }
```

Stack:

PC	x
3	100

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1 x = int(input())
2 cnt = 0
3 while (x > 0):
4     cnt = (cnt + 1)
5     x = (x / 2)
```

The above Python becomes, in C#:

```
1 int x = Int32.Parse(Console.ReadLine());
2 int cnt = 0;
3 while((x > 0)) {
4     cnt = (cnt + 1);
5     x = (x / 2);
6 }
```



Which in Java then becomes:

```
1  int x = Integer.parseInt(Console.ReadLine());  
2  int cnt = 0;  
3  while (x > 0) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }
```

## Control flow statements

This snippet (remember: we cannot just copy and paste it) produces the same execution in both Python and Java/C#!

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC
1

Input:

32

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC
1

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	x
2	32

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
3	0	32

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
4	0	32

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
5	1	32



# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
3	1	16

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
4	1	16

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
5	2	16

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
3	2	8

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
4	2	8

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
5	3	8

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
3	3	4

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
4	3	4



# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
5	4	4

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
3	4	2

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
4	4	2

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
5	5	2

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
3	5	1

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
7	5	1

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  int x = Int32.Parse(Console.ReadLine());  
2  int cnt = 0;  
3  while((x > 1)) {  
4      cnt = (cnt + 1);  
5      x = (x / 2);  
6  }  
7  Console.WriteLine(("Result is " + cnt.ToString()))
```

Stack:

PC	cnt	x
8	5	1

Output: "Result is 5"

## Classes

- Java/C# are object-oriented languages
- This means that (almost) everything is an **object**, that is an instance of a **class**
- All Java/C# programs will therefore begin with a class definition



# From Python to Java/C#

Introduction

The INFDEV  
team

A class in Java/C# looks very much like a Python class, with some minor differences:

- `__init__` is a method with the name of the class itself
- all fields must be declared, like variables, within the body of the class
- `self` is now called `this`

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1 class Counter:
2     def __init__(self):
3         self.cnt = 0
```

The above Python becomes, in C#:

```
1 class Counter {
2     private int cnt;
3     public Counter() {
4         this.cnt = 0;
5     }
6 }
```

Which in Java then becomes:

```
1  class Counter {  
2      private int cnt;  
3      public Counter() {  
4          this.cnt = 0;  
5      }  
6  }
```

## Visibility

- We can limit visibility of attributes (and methods) in a class in Java/C#;
- This means we can prevent a user of a class from accidentally using something in the wrong way
- Most important attributes are
  - public, means every part of the program can access it
  - private, means it can only be accessed from inside the class
- We assume for the moment that the constructor will always be public

Assuming `x` being an instance of `C`, this would be an invalid program:

```
1  class C {  
2      private int a;  
3      public int b;  
4      public C() {  
5          a = 0;  
6          b = 0;  
7      }  
8  }  
9  ...  
10 Console.WriteLine(x.a);
```

*In what sense invalid?*

# From Python to Java/C#

Introduction

The INFDEV  
team

Assuming `x` being an instance of `C`, this would be an invalid program:

```
1  class C {  
2      private int a;  
3      public int b;  
4      public C() {  
5          a = 0;  
6          b = 0;  
7      }  
8  }  
9  ...  
10 Console.WriteLine(x.a);
```

*In what sense invalid?*

The **compiler** will literally refuse to run the program by saying that `a` is private, and thus may not be accessed.

Which in Java then becomes:

```
1  class C {  
2      private int a;  
3      public int b;  
4      public C() {  
5          a = 0;  
6          b = 0;  
7      }  
8  }  
9  ...  
10 Console.WriteLine(x.a);
```

# From Python to Java/C#

Introduction

The INFDEV  
team

Assuming `x` being an instance of `C`, this would be a valid program, just like in Python:

```
1  class C {  
2      private int a;  
3      public int b;  
4      public C() {  
5          a = 0;  
6          b = 0;  
7      }  
8  }  
9  ...  
10 Console.WriteLine(x.b);
```

This suggests that Python is like Java/C# where all class attributes are automatically declared as public.



Which in Java then becomes:

```
1  class C {  
2      private int a;  
3      public int b;  
4      public C() {  
5          a = 0;  
6          b = 0;  
7      }  
8  }  
9  ...  
10 Console.WriteLine(x.b);
```

# From Python to Java/C#

Introduction

The INFDEV  
team

If we want to add methods, we also need to be aware of the type of each of their parameter and of the type they return.

```
1 class Counter:
2     def __init__(self):
3         self.cnt = 0
4     def incr(self,diff):
5         self.cnt = (self.cnt + diff)
```

The above Python becomes, in C#:

```
1 class Counter {
2     private int cnt;
3     public Counter() {
4         cnt = 0;
5     }
6     public void Incr(int diff) {
7         this.cnt = (this.cnt + diff);
8     }
9 }
```

Which in Java then becomes:

```
1  class Counter {  
2      private int cnt;  
3      public Counter() {  
4          cnt = 0;  
5      }  
6      public void Incr(int diff) {  
7          this.cnt = (this.cnt + diff);  
8      }  
9  }
```

## Methods

- Methods can, just like attributes, either `private` or `public`
- `public` methods can be called from anywhere
- `private` methods may only be called from inside the class itself

# From Python to Java/C#

Introduction

The INFDEV  
team

Now that we have a class, we can instantiate it and call its methods.

```
1 class Counter:
2     def __init__(self):
3         self.cnt = 0
4     def incr(self,diff):
5         self.cnt = (self.cnt + diff)
6 c = Counter()
7 c.incr(5)
```

The above Python becomes, in C#:

```
1 class Counter {
2     private int cnt;
3     public Counter() {
4         cnt = 0;
5     }
6     public void incr(int diff) {
7         this.cnt = (this.cnt + diff);
8     }
9 }
10 ...
11 Counter c = new Counter();
12 c.incr(5);
```

Which in Java then becomes:

```
1  class Counter {  
2      private int cnt;  
3      public Counter() {  
4          cnt = 0;  
5      }  
6      public void incr(int diff) {  
7          this.cnt = (this.cnt + diff);  
8      }  
9  }  
10 ...  
11 Counter c = new Counter();  
12 c.incr(5);
```

## Methods

This snippet (remember: we cannot just copy and paste it) produces the same execution in both Python and Java/C#!

```
1  class Counter {  
2      private int cnt;  
3      public Counter() {  
4          this.cnt = 0;  
5      }  
6      public void incr(int diff) {  
7          this.cnt = (this.cnt + diff);  
8      }  
9  }  
10 ...  
11 Counter c = new Counter();  
12 c.incr(5);
```

Stack:

PC
1



```
1  class Counter {  
2      private int cnt;  
3      public Counter() {  
4          this.cnt = 0;  
5      }  
6      public void incr(int diff) {  
7          this.cnt = (this.cnt + diff);  
8      }  
9  }  
10 ...  
11 Counter c = new Counter();  
12 c.incr(5);
```

Stack:

PC
11

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  class Counter {  
2      private int cnt;  
3      public Counter() {  
4          this.cnt = 0;  
5      }  
6      public void incr(int diff) {  
7          this.cnt = (this.cnt + diff);  
8      }  
9  }  
10 ...  
11 Counter c = new Counter();  
12 c.incr(5);
```

Stack:

PC
11

Heap:

1
cnt=

# From Python to Java/C#

Introduction

The INFDEV  
team

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public void incr(int diff) {
7          this.cnt = (this.cnt + diff);
8      }
9  }
10 ...
11 Counter c = new Counter();
12 c.incr(5);

```

Stack:

PC	...		PC	ret	this
11	...		4	null	ref 1

Heap:

1
cnt=

# From Python to Java/C#

Introduction

The INFDEV  
team

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public void incr(int diff) {
7          this.cnt = (this.cnt + diff);
8      }
9  }
10 ...
11 Counter c = new Counter();
12 c.incr(5);

```

Stack:

PC	...		PC	ret
11	...		4	null

Heap:

1
cnt=0

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  class Counter {  
2      private int cnt;  
3      public Counter() {  
4          this.cnt = 0;  
5      }  
6      public void incr(int diff) {  
7          this.cnt = (this.cnt + diff);  
8      }  
9  }  
10 ...  
11 Counter c = new Counter();  
12 c.incr(5);
```

Stack:

PC	c
12	ref 1

Heap:

1
cnt=0

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  class Counter {  
2      private int cnt;  
3      public Counter() {  
4          this.cnt = 0;  
5      }  
6      public void incr(int diff) {  
7          this.cnt = (this.cnt + diff);  
8      }  
9  }  
10 ...  
11 Counter c = new Counter();  
12 c.incr(5);
```

Stack:	PC	...		PC	ret	diff	this
	12	...		7	null	5	ref 1

Heap:	1
	cnt=0

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  class Counter {  
2      private int cnt;  
3      public Counter() {  
4          this.cnt = 0;  
5      }  
6      public void incr(int diff) {  
7          this.cnt = (this.cnt + diff);  
8      }  
9  }  
10 ...  
11 Counter c = new Counter();  
12 c.incr(5);
```

Stack:	PC	...		PC	ret
	12	...		7	null

Heap:	1
	cnt=5

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1 class Counter {  
2     private int cnt;  
3     public Counter() {  
4         this.cnt = 0;  
5     }  
6     public void incr(int diff) {  
7         this.cnt = (this.cnt + diff);  
8     }  
9 }  
10 ...  
11 Counter c = new Counter();  
12 c.incr(5);
```

Stack:

PC	c
13	ref 1

Heap:

1
cnt=5



# From Python to Java/C#

Introduction

The INFDEV  
team

Method access determines where they can be called. Suppose `x` is of type `C`:

```
1  class C {  
2      private int a;  
3      public int b;  
4      public C() {  
5          this.a = 0;  
6          this.b = 0;  
7      }  
8      public void IncrA(int diff) {  
9          this.a = (this.a + diff);  
10     }  
11     private void IncrB(int diff) {  
12         this.b = (this.b + diff);  
13     }  
14 }  
15 ...  
16 x.IncrA(10);
```

*Will this program be allowed to run?*

# From Python to Java/C#

Introduction

The INFDEV  
team

Method access determines where they can be called. Suppose `x` is of type `C`:

```
1  class C {  
2      private int a;  
3      public int b;  
4      public C() {  
5          this.a = 0;  
6          this.b = 0;  
7      }  
8      public void IncrA(int diff) {  
9          this.a = (this.a + diff);  
10     }  
11     private void IncrB(int diff) {  
12         this.b = (this.b + diff);  
13     }  
14 }  
15 ...  
16 x.IncrA(10);
```

*Will this program be allowed to run?*

Yes, because `IncrA` is a public method.

Which in Java then becomes:

```
1  class C {  
2      private int a;  
3      public int b;  
4      public C() {  
5          this.a = 0;  
6          this.b = 0;  
7      }  
8      public void IncrA(int diff) {  
9          this.a = (this.a + diff);  
10     }  
11     private void IncrB(int diff) {  
12         this.b = (this.b + diff);  
13     }  
14 }  
15 ...  
16 x.IncrA(10);
```

# From Python to Java/C#

Introduction

The INFDEV  
team

Method access determines where they can be called. Suppose `x` is of type `C`:

```
1  class C {  
2      private int a;  
3      public int b;  
4      public C() {  
5          this.a = 0;  
6          this.b = 0;  
7      }  
8      public void IncrA(int diff) {  
9          this.a = (this.a + diff);  
10     }  
11     private void IncrB(int diff) {  
12         this.b = (this.b + diff);  
13     }  
14 }  
15 ...  
16 x.IncrB(10);
```

*Will this program be allowed to run?*

# From Python to Java/C#

Introduction

The INFDEV  
team

Method access determines where they can be called. Suppose `x` is of type `C`:

```
1 class C {  
2     private int a;  
3     public int b;  
4     public C() {  
5         this.a = 0;  
6         this.b = 0;  
7     }  
8     public void IncrA(int diff) {  
9         this.a = (this.a + diff);  
10    }  
11    private void IncrB(int diff) {  
12        this.b = (this.b + diff);  
13    }  
14 }  
15 ...  
16 x.IncrB(10);
```

*Will this program be allowed to run?*

No, because `IncrB` is a private method.

Which in Java then becomes:

```
1  class C {  
2      private int a;  
3      public int b;  
4      public C() {  
5          this.a = 0;  
6          this.b = 0;  
7      }  
8      public void IncrA(int diff) {  
9          this.a = (this.a + diff);  
10     }  
11     private void IncrB(int diff) {  
12         this.b = (this.b + diff);  
13     }  
14 }  
15 ...  
16 x.IncrB(10);
```

## Static methods

Surprisingly, both Java and C# miss simple functions like those of Python: this means that they need to be emulated as methods.

# From Python to Java/C#

Introduction

The INFDEV  
team

Simple Python functions become *static methods* in both Java and C#.

```
1 def f(x):  
2     return (x + 10)
```

The above Python needs to be put inside a class and be marked as *static*, in both Java and C#:

```
1 class MyClass {  
2     static public int f(int x) {  
3         return (x + 10);  
4     }  
5 }
```



Which in Java then becomes:

```
1  class MyClass {  
2      static public int f(int x) {  
3          return (x + 10);  
4      }  
5  }
```

*static methods* are called without an instance of the class left of the dot, but rather with the name of the class they are declared in

```
1 class MyClass {  
2     static public int f(int x) {  
3         return (x + 10);  
4     }  
5 }  
6 ...  
7 Console.WriteLine(MyClass.f(10))
```

Which in Java then becomes:

```
1 class MyClass {  
2     static public int f(int x) {  
3         return (x + 10);  
4     }  
5 }  
6 ...  
7 System.out.println(MyClass.f(10))
```

## Static methods

This snippet (remember: we cannot just copy and paste it) produces the same execution in both Python and Java/C#!

```
1 class MyClass {  
2     static public int f(int x) {  
3         return (x + 10);  
4     }  
5 }  
6 ...  
7 Console.WriteLine(MyClass.f(10))
```

Stack:

PC
1

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1 class MyClass {  
2     static public int f(int x) {  
3         return (x + 10);  
4     }  
5 }  
6 ...  
7 Console.WriteLine(MyClass.f(10))
```

Stack:

PC
7

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1 class MyClass {  
2     static public int f(int x) {  
3         return (x + 10);  
4     }  
5 }  
6 ...  
7 Console.WriteLine(MyClass.f(10))
```

Stack:

PC	...		PC	ret	x
7	...		3	null	10

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1 class MyClass {  
2     static public int f(int x) {  
3         return (x + 10);  
4     }  
5 }  
6 ...  
7 Console.WriteLine(MyClass.f(10))
```

Stack:

PC	...		PC	ret
7	...		3	20



# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  class MyClass {  
2      static public int f(int x) {  
3          return (x + 10);  
4      }  
5  }  
6  ...  
7  Console.WriteLine(MyClass.f(10))
```

Stack:

PC
8

Output:

20

## The exttttMain method

- Java and C# programs do not just begin at the top of a file.
- The program is a class with a special static method, called `main`.
- The arguments to this method are an array of strings, the command line parameters<sup>a</sup>.

---

<sup>a</sup>Just ignore, it is mostly not used.

# From Python to Java/C#

Introduction

The INFDEV  
team

Here is our first actual Java/C# program of the day!

```
1  class Program {  
2      static public void Main(String[] args) {  
3          Console.WriteLine("Hello_world!")  
4      }  
5  }
```

# From Python to Java/C#

Introduction

The INFDEV  
team

Here is our first actual Java/C# program of the day!

```
1  class Program {  
2      static public void Main(String[] args) {  
3          Console.WriteLine("Hello_world!")  
4      }  
5  }
```

We will now run it: this is the first program we could copy in a file and just compile and run!

Which in Java then becomes:

```
1  class Program {  
2      static public void Main(String[] args) {  
3          System.out.println("Hello_world!")  
4      }  
5  }
```

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1  class Program {  
2      static public void Main(String[] args) {  
3          Console.WriteLine("Hello_world!");  
4      }  
5  }
```

Stack:

PC
1

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1 class Program {  
2     static public void Main(String[] args) {  
3         Console.WriteLine("Hello_world!")  
4     }  
5 }
```

Stack:

PC	...		PC	ret	args
6	...		3	null	null

# From Python to Java/C#

Introduction

The INFDEV  
team

```
1 class Program {  
2     static public void Main(String[] args) {  
3         Console.WriteLine("Hello world!")  
4     }  
5 }
```

Stack:

PC	...		PC	ret
6	...		3	null

Output: "Hello world!"



# Conclusion

## What have we seen so far?

- Intro to DEV3
- What we have learned so far: Python, from variables to basic classes
- Primitive types and declarations: an intuition about the type system
- Introduction to Java and C#: from variables to basic classes, with execution examples

# This is it!

Introduction

The INFDEV  
team

The best of luck, and thanks for the  
attention!