

# **APIP #1**

## **Introductie in Java & IntelliJ**



# Agenda

- De switch van Python naar Java
  - verschillen en overeenkomsten
- Waarom Java leren?
- IntelliJ, onze nieuwe IDE
- De taal Java
  - Datatypes
  - Output
  - Input
  - Operatoren
- Introductie oefenen in Stepik

# Overeenkomsten Python en Java

- Allebei geïnterpreteerde talen
  - Daarom beide sterk platform onafhankelijk
- Uitgebreide libraries (bibliotheken) voor beide
  - Python heeft Java op sommige vlakken wel ingehaald
- In Python is alles een object en in Java is bijna alles een object
- Zijn allebei afstammelingen van de Algol / C / C++ familie
  - Waarbij Python het meest afwijkt

Wat is  
geïnterpreteerd  
vs  
gecompileerd?

Later leren we  
wat een object  
en  
objectoriëntatie  
is

# Belangrijk verschil tussen Java & Python

**Manier van programmeren**

**Typing**

Wat is imperatief?



**Imperatief / Procedureel  
Object Oriented**

**Dynamically  
typed**



**Object Oriented**

**Statically  
typed**

**Gaan we  
zometeen  
over  
praten**

**DE HAAGSE**  
HOGESCHOOL

# Dynamische typing in Python

- Run de volgende code eens:

```
x = "Hello, world!"  
print(type(x))
```

```
x = 18  
print(type(x))
```

```
x = ['haha', 1, 'hihi']  
print(type(x))
```

# Dynamische typing in Python

- Run de volgende code eens:

```
x = "Hello, world!"  
print(type(x))
```

```
x = 18  
print(type(x))
```

```
x = ['haha', 1, 'hihi']  
print(type(x))
```

```
<class 'str'>  
<class 'int'>  
<class 'list'>  
> |
```

# Dynamische typing in Python

- In Python heeft een variabele geen vast data type
- Python regelt **achter de schermen** dat dit goed gaat
- Dit noemen we **dynamically typed**.

Wat zou **statically typed** dan kunnen betekenen?

# Statische typing in Java

Iedere soort data heeft zijn eigen statische / vaste ‘*type*’ waardoor Java weet welke data er verwacht wordt.

Java wil daarom bij declaratie van een variabele *expliciet* het type van een variabele weten.

Zal de variabele voor altijd een getal zijn? Dan geven we aan Java de “int” type mee.

Zal de variabele voor altijd een stuk tekst zijn? Dan geven we aan Java de String type mee.



# Statische typing in Java 2

## Python

```
message = 'Hello, world!'
```

```
age = 18
```

```
pi = 3.14
```

```
isBatman = false
```

## Java:

```
String message = "Hello, world!";
```

```
int age = 18;
```

```
double pi = 3.14;
```

```
boolean isBatman = false;
```

# Even mee oefenen

**Vul de ... in. Denk er niet alleen over na,  
maar *schrijf* je antwoorden op!**

**1. Welk datatype hoort bij de data?    2. Welke data hoort in het datatype?**

... value = 10;

String value = ...;

... value = false;

double value = ...;

... value = 13.87;

int value = ...;

... value = "Testing, 1, 2.";    boolean value = ...;

# Antwoorden

## 1. Welk datatype hoort bij de data?

```
int value = 10;  
boolean value = false;  
list value = 13.87;  
String value = "Testing, 1, 2.";
```

## 2. Welke data hoort in het datatype?

```
String value = "Goed";  
double value = 42.0;  
int value = 42;  
boolean value = true;
```



## Waarom Java leren?

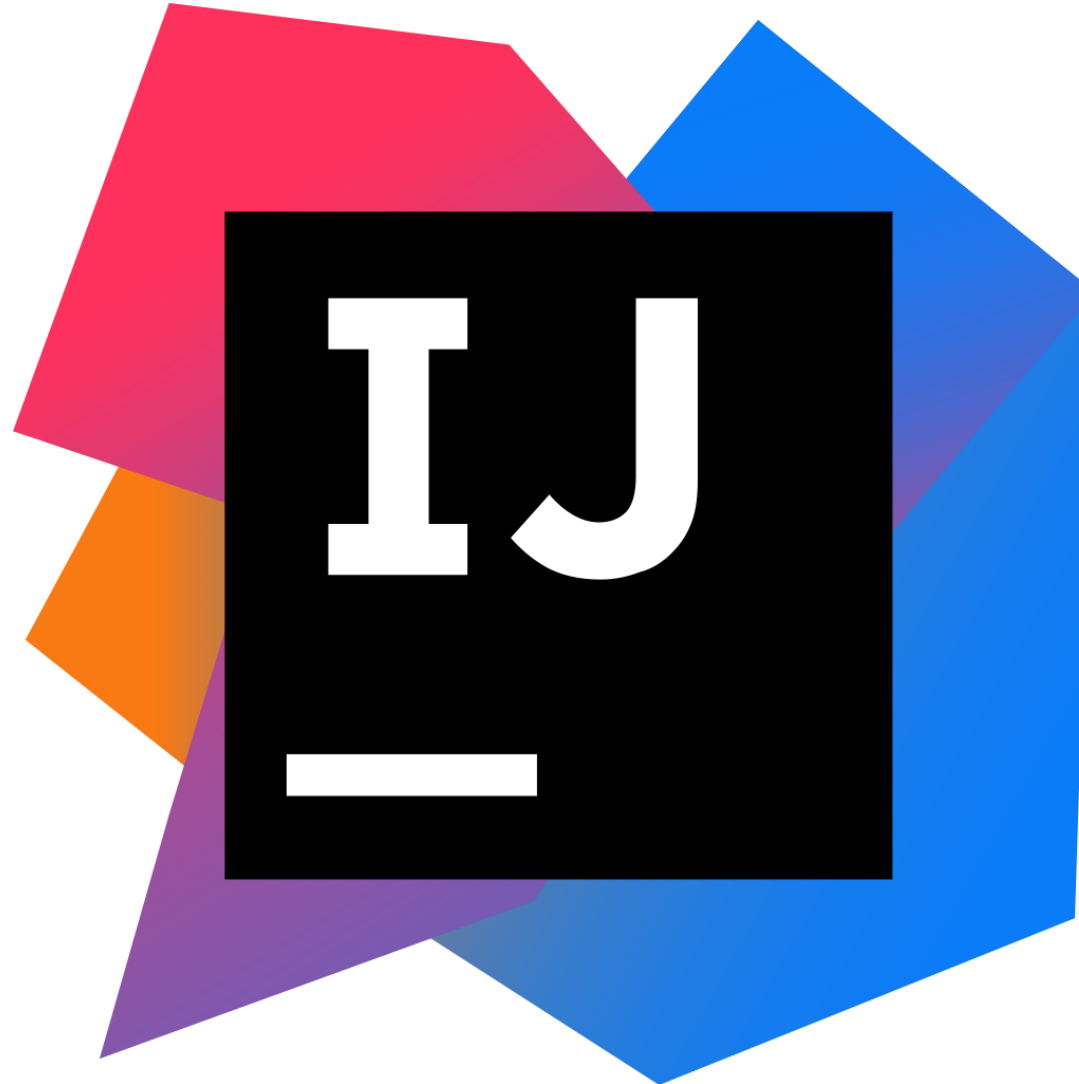
- Voelt super omslachtig dit allemaal
- Ik ken Python al
- Waar heb ik Java nog voor nodig?
- Etc...

# Waarom Java leren?

- Voelt super omslachtig dit allemaal
  - Ik ken Python al
  - Waar heb ik Java nog voor nodig?
  - Etc...
- 
- Java is al 20 jaar consistent in de top 3  
<https://www.tiobe.com/tiobe-index/>
  - Wat is de kans dat je de rest van je carrière alleen maar met Python werkt?
  - Een goede programmeur beheerst vaak meerdere talen

# Onze nieuwe IDE IntelliJ van JetBrains

(integrated development environment)



# Het begin van onze applicatie

Dit allemaal is nu niet relevant...

```
public class Main {  
  
    public static void main(String[] args) {  
        // write your code here  
    }  
}
```



Tijdens het begin van APIP is  
alleen tussen { } interessant

Later kijken we naar alle  
andere dingen

Imperatief?

# Output 1

Output is alles wat onze applicatie op het scherm zet.

In **Python** kennen jullie `print('Hello, world!')`

In **Java** kunnen we dit ook! Dit doen we met  
`System.out.println("Hello, world!");`

De output komt in de **console van onze IDE**. IntelliJ heeft voor deze lange print statement ook een shortcut: `sout` + enter



## Output 2

```
String message = "Hello!";  
System.out.println(message);  
Output: Hello!
```

```
String otherMessage = "Hello!";  
System.out.println(otherMessage + "world!");  
Output: Hello!world!
```

### Exercise

System.out.println(10);	0: ???
System.out.println("10");	0: ???
System.out.println(10 + 10);	0: ???
System.out.println("10" + 10);	0: ???
System.out.println(20 + 20 + "20" + 20 + 20);	0: ???

## Output 2

```
String message = "Hello!";  
System.out.println(message);  
Output: Hello!
```

```
String otherMessage = "Hello!";  
System.out.println(otherMessage + "world!");  
Output: Hello!world!
```

### Exercise

System.out.println(10);	0: 10
System.out.println("10");	0: 10
System.out.println(10 + 10);	0: 20
System.out.println("10" + 10);	0: 1010
System.out.println(20 + 20 + "20" + 20 + 20);	0: 40202020

## Output 3: println vs print

```
System.out.println("Hello, ");  
System.out.println("World!");
```

```
System.out.print("Hello, ");  
System.out.print("World!");
```

```
System.out.print("Hello, ");  
System.out.println();  
System.out.print("World!");
```

```
System.out.print("Hello, \n");  
System.out.print("World!");
```


# Output 3: println vs print antwoorden

```
System.out.println("Hello, ");  
System.out.println("World!");  
// Hello,  
// World!
```

```
System.out.print("Hello, ");  
System.out.print("World!");  
// Hello, World!
```

```
System.out.print("Hello, ");  
System.out.println();  
System.out.print("World!");  
// Hello,  
// World!
```

```
System.out.print("Hello, \n");  
System.out.print("World!");  
// Hello,  
// World!
```

**Korte pauze**  
**Terug over ? minuten** 

# Imports

- Imports werken in Java en Python op een soortgelijke manier

```
import java.util.Scanner; import matplotlib.pyplot as plt  
import java.lang.Math;    import numpy as np
```

# Input in Python

Input is de actie als een gebruiker iets moet invoeren bij onze applicatie.

In Python:

```
my_input = input()  
print(my_input + 5)  
print(int(my_input) + 5)
```

Relatief makkelijk. Alles wordt als een string ingelezen en jij mag dit zelf later naar andere types converteren.

In Java is het iets gecompliceerder.

# Input in Java

- In Java eerst een zogenaamde **Scanner** aanmaken die input kan 'scannen'
- Vervolgens moeten we de **Scanner** vertellen wat hij moet "inscannen". Een "String"? Een int? Een double? Een boolean? Etc.

```
import java.util.Scanner;
```

```
public Class Main {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        String textMessage = scanner.nextLine(); // tekst inlezen  
        int number = scanner.nextInt(); // geheel getal inlezen  
    }  
}
```

**Wat zouden we dan moeten gebruiken om een  
double of een boolean in te lezen?**



# Input classic bug

Als je een getal inleest met `nextInt();` en direct daarna een `nextLine();` gebruikt, werkt dit dan naar verwachting?

```
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
  
    int number = scanner.nextInt();  
    String message = scanner.nextLine();  
  
    System.out.println(number + message);  
}
```

# Input classic bug fix

`nextInt();` pakt het eerste gehele getal, maar **niet** de newline.

`nextLine();` pakt op zijn beurt alles tot en met de newline

Doe daarom een extra `nextLine();` om de “inputbuffer” te legen

```
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
  
    int number = scanner.nextInt();  
    scanner.nextLine();  
    String message = scanner.nextLine();  
  
    System.out.println(number + message);  
}
```

# Operatoren

Een operator is een karakter of een combinatie van karakters. Samen bepalen ze wat voor actie er genomen moet worden.

```
int a = 5 + 6 * 30;  
System.out.println(a); // wat wordt er hier geprint?
```

# Operatoren voor deze week

Multiplicative	* / %
Additive	+ -
Assignment	=

# Modulo 1

- Naast de standaard  $+$ ,  $-$ ,  $/$ ,  $*$  wiskundige operatoren hebben we ook nog de modulo operator:

**%**

- Modulo berekent de rest waarde van het delen:

$10 \% 5 = 0$  (want 5 past 2 keer als geheel in 10, en dan houden we 0 over)

$12 \% 5 = 2$  (want 5 past 2 keer als geheel in 12, en dan houden we 2 over)

$3 \% 2 = 1$  (want 2 past 1 keer als geheel in 3, en dan houden we 1 over)

$5 \% 2 = 1$  (want 2 past 2 keer als geheel in 5, en dan houden we 1 over)

# Modulo 2

## Oefeningen

$$10 \% 5 = 0$$

$$12 \% 5 = 2$$

$$12 \% 4 = ?$$

$$12 \% 6 = ?$$

$$12 \% 7 = ?$$

$$38 \% 9 = ?$$

$$50 \% 50 = ?$$

$$50 \% 0 = ?$$

# Modulo 2

## Oefeningen

$$10 \% 5 = 0$$

$$12 \% 5 = 2$$

$$12 \% 4 = ?$$

$$12 \% 6 = ?$$

$$12 \% 7 = ?$$

$$38 \% 9 = ?$$

$$50 \% 50 = ?$$

$$50 \% 0 = ?$$

## Antwoorden

$$10 \% 5 = 0$$

$$12 \% 5 = 2$$

$$12 \% 4 = 0$$

$$12 \% 6 = 0$$

$$12 \% 7 = 5$$

$$38 \% 9 = 2$$

$$50 \% 50 = 0$$

# Oefenen met operatoren: output en type?

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println(10 / 3);  
        System.out.println(10.0 / 3);  
        System.out.println(10 / 3.0);  
  
        System.out.println(13 + 10 % 3);  
        System.out.println(13 + 10 % 3 + "8");  
        System.out.println(13 + 10 % 3 + '8');  
  
        System.out.println(19 + 8 * 10 / 7 % 5 + 10);  
        System.out.println(19 + 8 * 10 / 7 % 5 + 10 + "8");  
        System.out.println(19 + 8 * 10 / 7 % 5 + 10 + '8');  
        // hoe moeten we machtsverheffen in Java?  
    }  
}
```



# Oefenen met operatoren: output en type?

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println(10 / 3);           // 3  
        System.out.println(10.0 / 3);         // 3.3333333333333335  
        System.out.println(10 / 3.0);         // 3.3333333333333335  
  
        System.out.println(13 + 10 % 3);       // 14  
        System.out.println(13 + 10 % 3 + "8"); // 148  
        System.out.println(13 + 10 % 3 + '8'); // 70 ('8' heeft ASCII  
                                                waarde 56)  
  
        System.out.println(19 + 8 * 10 / 7 % 5 + 10); // 30  
        System.out.println(19 + 8 * 10 / 7 % 5 + 10 + "8"); // 308  
        System.out.println(19 + 8 * 10 / 7 % 5 + 10 + '8'); // 86  
  
        // hoe moeten we machtsverheffen in Java?  
    }  
}
```

# printf


Printf is een functie die overweg kan met zogenaamde format specifiers, placeholders die het type weergeven. De variabelen die je geprint wil hebben moet je ook meegeven aan printf:

```
public class Main {  
    public static void main(String[] args) {  
        String s1 = "World";  
        System.out.printf("Hello %s\n", s1);  
        System.out.printf("Er zitten %d studenten in de klas\n", 50);  
        System.out.printf("De student heeft een %f behaald voor zijn toets\n", 8.3);  
    }  
}
```

# printf

Je kan ook weergeven hoeveel decimalen je achter de komma wilt hebben. Of hoeveel ruimte een getal überhaupt moet innemen. Of links en rechts uitlijnen

```
public class Main {  
    public static void main(String[] args) {  
        System.out.printf("De student heeft een %f behaald voor zijn toets\n", 8.3);  
        System.out.printf("De student heeft een %.1f behaald voor zijn toets\n", 8.3);  
        System.out.printf("Dit kost € % 10.2f\n", 41.83);  
        System.out.printf("Dit kost € % -10.2f\n", 41.83);  
        System.out.printf("%c is de eerste letter van het alfabet\n", 'A');  
    }  
}
```

**Korte pauze**  
**Terug over ? minuten** 

# Hoe oefenen?

## Intro Stepik by Jaap

# Vragen?



*“That’s all Folks!”*