IDE

Visual Studio Code	IntelliJ IDEA

Commentaar

```python
# 1-lijns-commentaar
```

```python
'''
    Multi-lijns
    Commentaar
'''
```

```python
lijst = [1, 2,
# 3,
 4, 5]

# s1 = "cat is" + " not" + " dead"
s2 = "cat is" + " dead"
```

```java
// 1-lijns-commentaar
```

```java
/*
    Multi-lijns
    Commentaar
*/
```

Midline comment

```java
int[] getallen = new int[] {1, 2, /*3 ,*/ 4};
String s = "cat is" + /* " not" + */ " dead";
```

https://stackoverflow.com/questions/5617159/mid-line-comment-in-python

https://stackoverflow.com/questions/69080019/how-to-comment-out-a-chunk-of-a-single-line-of-code-in-python

Printen naar consolevenster

```python
a = 10
b = 5
print('Waarden: ' + str(a) + " en " + str(b))
print('Waarden:', a, "en", b)
print(f'Waarden: {a} en {b}')
```

```python
print('eerste', end=' ')
print('volgende op zelfde lijn')
```

```python
import math
PI = math.pi
print(f'{PI:.2f}') # 3.14
```

```java
int a = 10, b = 5;
System.out.println("Waarden: " + a + " en " + b);
System.out.println(String.format("Waarden: %d en %d",
a, b));
System.out.printf("Waarden: %d en %d\n", a, b);
System.out.println(MessageFormat.format("Waarden: {0}
en {1}", a, b));
```

https://www.baeldung.com/java-string-interpolation

```java
System.out.print("eerste ");
System.out.println("volgende op zelfde lijn");
```

```java
double PI = Math.PI;
System.out.println(String.format("%.2f", PI));
```

Lezen van het toetsenbord

```python
naam = input("Geef je naam:")
leeftijd = int(input('Geef je leeftijd'))
print(f'{naam} {leeftijd}')
```

```java
import java.util.Scanner;
Scanner scan = new Scanner(System.in);
System.out.println("Geef je naam: ");
String naam = scan.nextLine();
System.out.println("Geef je leeftijd: ");
int leeftijd = scan.nextInt();
```

```
System.out.println(naam + " " + leeftijd);
```

Type conversie / Java wrapper klassen

```
a = 123
b = 'tekst'
print(b, type(b)) # tekst <class 'str'>
b = a
print(b, type(b)) # 123 <class 'int'>
```

Conversie int naar String:

```
a = 123
b = str(a)
print(b, type(b)) # 123 <class 'str'>
```

Conversie String naar int:

```
s = "123"
d = int(s)
print(d, type(d)) # 123 <class 'int'>
```

Conversie int naar String:

```
int a = 123;
String b = "tekst";
b = Integer.toString(a); //stringwaarde "123"
```

Conversie String naar int:

```
String s = "123";
int d = Integer.parseInt(s); //intwaarde 123
```

String methoden

```
streepjes = '-' * 10
print(streepjes) # ----------
print(len(streepjes)) # 10
print(type(s)) # <class 'str'>
```

```
naam = "Bob"
print(len(naam))
print(naam.upper())
naam = naam.strip()

idx1 = naam.find('i');   # 2
idx2 = naam.rfind('i');  # 5
idx3 = naam.find("is");  # 2
idx4 = naam.rfind("is"); # 2
```

Index van karakter in String:

```
c = 'a'
klinker = "aeiou".find(c) != -1
print(klinker)  # True
```

```
print(''.join(reversed(naam)))

print(naam[::-1])

print(''.join(naam[i] for i in
             range(len(naam)-1, -1, -1)))
```

```
System.out.println('-' * 10); //45 * 10 = 450
String streepjes = "-".repeat(10);
System.out.println(streepjes); //----------
System.out.println(streepjes.length()); //10
```

```
String naam = "Bob";
System.out.print(naam.length());
System.out.println(naam.toUpperCase());
naam = naam.trim();

int idx1 = naam.indexOf('i');     //2
int idx2 = naam.lastIndexOf('i'); //5
int idx3 = naam.indexOf("is");    //2
int idx4 = naam.lastIndexOf("is");//2
```

Index van karakter in String:

```
char c = 'a';
boolean klinker = "aeiou".indexOf(c) != -1;
System.out.println(klinker); // true
```

Veel stringmanipulaties: Gebruik klasse StringBuilder
>> append(), delete(), insert(), reverse(), indexOf(), lastIndexOf(), charAt(), isEmpty(), length(), substring(), trimToSize(), …

```
StringBuilder sb = new
StringBuilder("schipper").reverse();
```

Tip: Chainen van methoden

String indexering

String indexering en slicing

```python
naam = "Bob De Bouwer"
idx = naam.find(' ')
voornaam = naam[0:idx]
familienaam = naam[idx+1:]
print(voornaam,familienaam)
```

String indexering

```java
String naam = "Bob De Bouwer";
int idx = naam.indexOf(' ');
String voornaam = naam.substring(0, idx);
String familienaam = naam.substring(idx+1);
System.out.println(voornaam + " " + familienaam);
```

Basis types

Integers (built-in type):

```python
i = 100_000
print(i, type(i))
# 100000 <class 'int'>
```

Floats (built-in type):

```python
f = 3.14
print(f, type(f))
# 3.14 <class 'float'>
```

Booleans (built-in type):

```python
b = True
print(b, type(b))
# True <class 'bool'>
```

Strings (built-in type):

```python
s = 'Tekst'
print(s, type(s))
# Tekst <class 'str'>
```

None (built-in type):

```python
x = None
print(x, type(x))
# None <class 'NoneType'>
```

Complex (built-in type):

```python
c = 5 + 3j
print(c, type(c))
# (5+3j) <class 'complex'>
```

8 Java primitieve types & bijhorende Java wrapper klasse

Gehele getallen (4 primitieve types):

```java
/* byte > Byte: */ byte b = 127;
/* short > Short: */ short s = 12;
/* int > Integer: */ int i = 100000;
/* long > Long: */ long l = 123456;
```

Reële getallen (2 primitieve types):

```java
/* double > Double: */ double d = 12.34;
/* float > Float: */ float f = 3.1415f;
```

Booleans (primitief type):

```java
/* boolean > Boolean: */ boolean bl = true;
```

Karakters (primitief type):

```java
/* char > Character: */ char c = 'c';
```

De klasse String (object type):

```java
String tekst1 = "Tekst";
String tekst2 = new String("Tekst");
boolean refsGelijk = tekst1 == tekst2; //false
boolean inhoudGelijk = tekst1.equals(tekst2); //true
```

Java klasse Math

```python
print(type(pow)) # <class 'builtin_function_or_method'>
print(type(round)) # <class 'builtin_function_or_method'>

afgerond = round(7.5) # 8
afgerond = round(1.23456, 4) # 1.2346

res = pow(2,8) # 256
res = pow(4,2,8) # 0
```

```java
long afgerond = Math.round(7.5); // 8
double afgerond2 = Math.round(1.23456 * 10000) /
10000.0; // 1.2346

double d = Math.pow(2,8);
```

```python
import math as m
print(type(m)-) # <class 'module'>
print(m.sqrt(m.pi))

import math
```

```java
System.out.println(Math.sqrt(Math.PI));

System.out.println(Math.sin(Math.PI/2));
double s90 = Math.sin(Math.toRadians(90));
```

```python
print(math.sin(math.pi/2)) # 1.0
print(math.sin(math.radians(90))) # 1.0

from math import sin, radians as rad
print(sin(rad(90))) # 1.0

from math import *
print(sqrt(pi))
print(sin(pi/2)) # 1.0

from math import pi, sqrt, factorial as fac
print(sqrt(pi))
print(fac(3))

print(type(pi)) # <class 'float'>
print(type(sqrt)) # <class 'builtin_function_or_method'>
print(type(fac)) # <class 'builtin_function_or_method'>
```

```java
System.out.println(s90);
```

```python
from random import Random
random = Random().random() #[0, 1[

random = Random().randint(10,20)
# [10, 20]
random = Random().randrange(10,100,5)
# [10,15,20, ..., 100]
random = choice(range(10,100,5))
# [10,15,20, ..., 100]
```

```java
double r1 = Math.random(); //[0,1[

int max = 20, min = 10;
long r2 = Math.round(((Math.random() * (max - min)) + min)); //[10,20[
```

Python modules random en string

```python
import random, string

leesteken = string.punctuation[random.randint
          (0,len(string.punctuation)-1)]
leesteken = random.choice(string.punctuation)
letter_a_z =
random.choice(string.ascii_lowercase)
cijfer_0_9 = random.choice(string.digits)
```

/

Operatoren

```python
a = 1
b = a   # b = 1
a += 1  # a = 2
```

Analoog voor a -= 1

postfix increment:

```java
int a = 1;
int b = a++; // b = 1 en a = 2
```

Analoog voor postfix decrement : a--

```python
a = 1
a += 1  # a = 2
b = a   # b = 2
```

Analoog voor a -= 1

prefix increment:

```java
int a = 1;
int b = ++a; // b = 2 en a = 2
```

Analoog voor prefix decrement : --a

```python
a += 1
b -= 1
c *= 2
```
Analoog voor andere: +=, -=, *=, /=

Rekenkundige operatoren:

```java
a += 1;
b -= 1;
c *= 2 ;
```
Analoog voor andere: +=, -=, *=, /=

```python
i = 11
res_geheel = i // 2   # 5
rest = i % 2          # 1
res_met_komma = i / 2 # 5.5
```

```java
int i = 11;
int resGeheel = i / 2; // 5
int rest = i % 2;      // 1
double resMetKomma = ((double)i) / 2; // 5.5
```

```python
a = 12
d = a ** 2 # 144
```

```
>> zie methode pow in Math klasse
```

```python
b1 = False
b2 = True
res = b1 and b2 # False
res = b1 or b2  # True
res = not b1    # True
```

Logische operatoren: && || en !

```java
boolean b1_ = false, b2_ = true, res;
res = b1_ && b2_; //false
res = b1_ || b2_; //true
res = !b1_; //true
```

https://realpython.com/python-bitwise-operators/

Bitwise operatoren: & | ^ ~ << en >>

Selecties

```python
if getal // 100 > 0:
    print('Hondertal ')
elif getal // 10 > 0:
    print('Tiental')
else:
    print('Eenheid')
```

```java
if (getal / 100 > 0) {
    System.out.println("Hondertal");
}
else if (getal / 10 > 0) {
    System.out.println("Tiental");
}
else {
    System.out.println("Eenheid");
}
```

```python
match(dag):
    case 6: print('zaterdag')
    case 7: print('zondag')
    case default: print('weekdag')
```

```java
switch (dag) {
   case 6: res = "zaterdag";
           break;
   case 7: res = "zondag";
           break;
   default: res = "weekdag";
}
```

```python
match(dag):
    case 1|2|3|4|5: print('weekdag')
    case 6|7: print('weekend')
    case _: print('ongeldig')
```

```java
switch(dag) {
   case 1,2,3,4,5: res = "weekdag";
                   break;
   case 6,7: res = "weekend";
```

```
            break;
    default: res = "ongeldig";
}
```

For-lussen

```
for i in range(10):
    print(i)
```

```
for(int i = 0; i < 10; i++) {
     System.out.println(i);
}
```

```
lijst = [12,24,36,48,60]
for h in lijst:
     print(h)
```

```
ArrayList<Hond> al = new ArrayList<>();
al.add(hondje);
for (Hond h : al) {
   System.out.println(h);
}
```

```
woord = 'test'
for letter in woord:
     print(letter, end = '')
```

```
String woord = "test";
for (char c : woord.toCharArray()) {
   System.out.print(c);
}
```

```
waarden = [1,2,3,4,5]
som = 0
for waarde in waarden:
     som += waarde
```

```
int[] waarden = new int[] {1,2,3,4,5};
int som = 0;
for (int waarde : waarden) {
   som += waarde;
}
```

While-lussen

```
while voorwaarde:
    # doe iets
    # wijzig voorwaarde
```

```
while (voorwaarde) {
   // doe iets
   // wijzig voorwaarde
}
```

```
do {
   // doe iets
   // wijzig voorwaarde
} while (voorwaarde);
```

break – continue – return

```
def verwerk():
    result = ''
    for i in range(10):
        result += str(i)
        if i == 4: return
        if i == 3: break
```

```
public static String verwerk() {
   String result = "";
   for (int i = 0; i < 10; i++) {
      result += i;
      if (i == 4) return result;
      if (i == 3) break;
      if (i == 2) continue;
    }
```

```python
        if i == 2: continue
    return result

print(verwerk()) # 0123
```

```java
    return result;
}

System.out.println(verwerk()); // 0123
```

```python
def verwerk():
    result = 0
    for i in range(10):
        result += i
        if i == 4: return
        if i == 3: break
        if i == 2: continue
    return result

print(verwerk()) # 6
```

```java
public static int verwerk() {
   int result = 0;
   for (int i = 0; i < 10; i++) {
      result += i;
      if (i == 4) return result;
      if (i == 3) break;
      if (i == 2) continue;
   }
   return result;
}
System.out.println(verwerk()); //6
```

Eigen functies/methoden

Eigen functies documenteren via Python docstring

```python
def maal(getal1, getal2):
    """Geeft het produkt van getal1 en getal2 terug."""
    produkt = getal1 * getal2
    return product
```

Eigen methoden documenteren via Javadoc

```java
/**
 * Geeft het produkt van 2 getallen terug
 * @param getal1 eerste getal
 * @param getal2 tweede getal
 * @return product van de 2 getallen
 */
public static int maal(int getal1, int
getal2) {

   int produkt = getal1 * getal2;
   return produkt;
}
```

Functie definitie:

```python
def is_klinker(c):
    return "aeiou".find(c) != -1
```

```python
def is_klinker(c):
    return c in 'aeiou'
```

Functie oproep:

```python
print("klinker" if is_klinker('a') else
"medeklinker")
```

Methode definitie:

```java
public static boolean isKlinker(char c) {
   return "aeiou".indexOf(c) != -1;
}
```

Methode oproep:

```java
System.out.println(isKlinker('a') ?
"klinker" : "medeklinker");
```

Ternaire operator

```python
ltr = 'a'
res = "kl" if is_klinker(ltr) else "mdkl"
print(res)
```

```java
char ltr = 'a';
String res = isKlinker(ltr) ? "kl" : "mdkl";
System.out.println(res);
```

```python
zin = "Hier ga ik mee aan de slag"
zin_klinkers_naar_X = ''.join('X' if i in
'aeiou' else i for i in zin)
print(zin_klinkers_naar_X)
```

Uitvoer: HXXr gX Xk mXX XXn dX slXg

```java
String zin = "Hier ga ik mee aan de slag";
String zinKlinkersNaarX = "";
for (char c : zin.toCharArray()) {
   zinKlinkersNaarX += isKlinker(c) ? 'X': c;
}
System.out.println(zinKlinkersNaarX);
```

Uitvoer: HXXr gX Xk mXX XXn dX slXg

Kan evt. ook via String.join:

```java
zinKlinkersNaarX = String.join("",
            zinKlinkersNaarX, isKlinker(c) ?
                                "X": c + "");
```

Java klasse LocalDate

```python
from datetime import datetime
curJaar = datetime.now().year
```

```java
import java.time.LocalDate;
LocalDate today = LocalDate.now();
System.out.println(today); //2024-01-05
int curJaar = today.getYear();
System.out.println(curJaar); //2024

int dag = 29, maand = 2, jaar = 2028;
LocalDate datum = LocalDate.of(jaar, maand, dag);
System.out.println(datum);
```

Binaire getallen / Java klasse Integer:

```python
getal = 15
print(bin(getal))
print(f'{getal:b}')

print(bin(getal)[2:].zfill(8))
print(f'{getal:b}'.zfill(8))
```

```java
int getal = 15;
String binGetal = Integer.toBinaryString(getal);
String nullen = "0".repeat(8 - binGetal.length());
System.out.println(nullen + binGetal);
```

Onthoud:

- Java is net als Python hoofdlettergevoelig
- In java worden variabelen gedeclareerd : je bepaalt van welk type elke variabele is vóór gebruik
- Java kent method overloading: meerdere methoden met zelfde naam, maar andere parameterlijst
- Een methode geeft steeds 1 type terug, dat kan een primitief type zijn, een object type, of void wanneer niets teruggegeven wordt
- Voor de naamgeving van Java klassen gebruik je *PascalCasing*, voor Java methoden en Java variabelen gebruik je *camelCasing*
- Een constante definieer je als een `final` variabele. Als naam gebruik je enkel hoofdletter waarbij tussen de deelwoorden een underscore wordt gebruikt
- Java kent als operatoren: de rekenkundige, de relationele, de logische en de bitoperatoren
- Java kent logische expressies en ternaire operatoren: `var =(voorwaarde ? waardeIfTrue : waardeIfFalse);`
- ...