

Uvod v programska jezika Python in Java

Programska jezika **Python** in **Java** sta **splošnonamenska programska jezika** v katerih je moč sestaviti praktično katerikoli program.

Programski jezik **Python** je trenutno **najpopularnejši jezik** zaradi enostavne sintakse in obilice prosto-dostopnih programskih knjižnic, dočim pa ni najhitrejši jezik. Jezik se interpretira, kar pomeni, da program `demo.py` v ukazni vrstici izvedemo s `python demo.py`.

Programski jezik **Java** je bil razvit kot **varen jezik** za poljubno napravo. Sintaksa zahteva daljše programe, ki pa so lahko tudi desetkrat hitrejši kot programi v jeziku Python. Jezik se prevaja, kar pomeni, da program `Demo.java` v ukazni vrstici najprej prevedete z `javac Demo.java`, kar ustvari izvorno datoteko `Demo.class`, katero nato izvedete z `java Demo`.

Najkrajši program in izpis na zaslon

V programskem jeziku **Python bloke kode**, ki naj se izvedejo skupaj oziroma zaporedoma, določimo z *zamikanjem*. Vsak programski stavek praviloma zapišemo v svoji vrstici.

```
print('Pozdravljeni pri predmetu PR02!')
```

Python

V programskem jeziku **Java bloke kode**, ki naj se izvedejo skupaj oziroma zaporedoma, določimo z zavitimi oklepaji `{...}`. Vsak programski stavek zaključimo s podpičjem `;`, dočim je lahko celoten program v eni vrstici.

```
public class Demo {  
    public static void main(String[] args) {  
        System.out.println("Pozdravljeni pri predmetu PR02!");  
    }  
}
```

Java

Programske spremenljivke in konstante

V programskem jeziku **Python spremenljivke** definiramo in jim določimo začetno vrednost hkrati. Pri tem **ne določimo tipa** spremenljivke.

Python

```
x = 1
y = 1.23
z = True
ch = 'a'
st = "niz znakov"
# vrstični ali bločni komentar
```

V programskem jeziku **Python konstante** obravnavamo enako kot spremenljivke. Navadno jih označimo z velikimi črkami.

Python

```
G = 9.81
```

V programskem jeziku **Java spremenljivke** definiramo in jim *lahko* določimo začetno vrednost hkrati. Pri definiciji **moramo določiti tip** spremenljivke, ki ga ni moč spremeniti. Osnovni tipi spremenljivk so cela števila (tj. `byte`, `short`, `int`, `long`), realna števila (tj. `float`, `double`), logične vrednosti (tj. `boolean`) in nizi znakov (tj. `char` in razred `String`).

Java

```
int x = 1;
double y;
y = 1.23;
boolean z = true;
char ch = 'a';
String st = "niz znakov";
```

V programskem jeziku **Java konstante** označimo z besedo `final` in jih ne moremo naknadno spreminjati. Navadno jih označimo z velikimi črkami.

Java

```
final double G = 9.81;
```

Pogojni stavki in programske vejitve

Programske vejitve omogočajo **selektivno izvajanje** programske kode glede na logičen pogoj.

Najpogosteje se uporabljajo pogojni stavki (tj. `if else` stavki), dočim v večini programskih jezikih obstajajo tudi izbirni stavki (tj. `switch` stavki) in drugi. Vse pogojne stavke je moč gnezditi ipd.

V programskem jeziku **Python pogojne stavke** zapišemo kot je prikazano spodaj.

Python

```

if x < 1:
    print("Vrednost spremenljivke x je manjša od 1")
elif x < 2:
    print("Vrednost spremenljivke x je med 1 in 2")
else:
    print("Vrednost spremenljivke x je večja ali enaka 2")

```

V programskem jeziku **Java** **pogojne stavke** zapišemo kot je prikazano spodaj.

Java

```

if (x < 1) {
    System.out.println("Vrednost spremenljivke x je manjša od 1");
}
else if (x < 2) {
    System.out.println("Vrednost spremenljivke x je med 1 in 2");
}
else
    System.out.println("Vrednost spremenljivke x je večja ali enaka 2");

```

Iterativno izvajanje in programske zanke

Programske zanke omogočajo **ponovljeno izvajanje** programske kode dokler velja logičen pogoj.

Najpogostejše se uporabljajo osnovne zanke (tj. `for` in `while` zanke), dočim v večini programskih jezikih obstajajo tudi npr. `do while` zanke in druge. Zanke lahko predčasno zaključimo z uporabo ukaza `break`, novo iteracijo zanke pa lahko predčasno pričnemo z uporabo ukaza `continue`. Vse zanke je moč gnezditi ipd.

V programskem jeziku **Python** **for** **zanko** zapišemo kot je prikazano spodaj.

Python

```

for i in range(5):
    print("Vrednost spremenljivke i je enaka " + str(i))

```

Ekvivalentno lahko v programskem jeziku **Python** **while** **zanko** zapišemo kot je prikazano spodaj.

Python

```

i = 0
while i < 5:
    print("Vrednost spremenljivke i je enaka " + str(i))
    i += 1

```

V programskem jeziku **Java** **for** **zanko** zapišemo kot je prikazano spodaj.

Java

```
for (int i = 0; i < 5; i++)
    System.out.println("Vrednost spremenljivke i je enaka " + i);
```

Ekvivalentno lahko v programskem jeziku **Java** **while** **zanko** zapišemo kot je prikazano spodaj.

Java

```
int i = 0;
while (i < 5) {
    System.out.println("Vrednost spremenljivke i je enaka " + i);
    i++;
}
```

Programske metode, procedure in funkcije

Programske metode, procedure in funkcije omogočajo **ponovljeno izvajanje** enake programske kode upoštevajoč izbrane parametre. Dočim procedure ali metode zgolj izvedejo določeno programsko kodo, funkcije poleg tega vrnejo tudi rezultat z uporabo stavka `return`.

V programskem jeziku **Python metodo** zapišemo kot je prikazano spodaj.

Python

```
def method(i):
    print("Vrednost parametra i je enaka " + str(i))
```

V programskem jeziku **Python funkcijo** zapišemo kot je prikazano spodaj.

Python

```
def function(i):
    print("Vrednost parametra i je enaka " + str(i))
    i *= 7
    print("Vrednost rezultata funkcije je enaka " + str(i))
    return i
```

V programskem jeziku **Java metodo** zapišemo kot je prikazano spodaj.

Java

```
public static void method(int i) {
    System.out.println("Vrednost parametra i je enaka " + i);
}
```

V programskem jeziku **Java funkcijo** zapišemo kot je prikazano spodaj.

Java

```
public static int function(int i) {  
    System.out.println("Vrednost parametra i je enaka " + i);  
    i *= 7;  
    System.out.println("Vrednost rezultata funkcije je enaka " + i);  
    return i;  
}
```

V programskem jeziku **Python lambda funkcije** zapišemo kot je prikazano spodaj.

Python

```
def inc(x):  
    return x + 1  
# ali  
inc = lambda x: x + 1  
  
def pow(x, p):  
    return x**p  
# ali  
pow = lambda x, p: x**p
```