

Napredno programiranje i programski jezici

10 Python

Fakultet tehničkih nauka, Novi Sad
23-24/Z
Dunja Vrbaški

Svaki jezik - dovoljan editor
IDE - ubrzanje razvoja složenog softvera

C++:
CodeBlocks

Java:
Eclipse

Laboratorija → Izbor IDE

Python - izbor editora ili IDE programa proizvoljan

Visual Studio Code (editor)

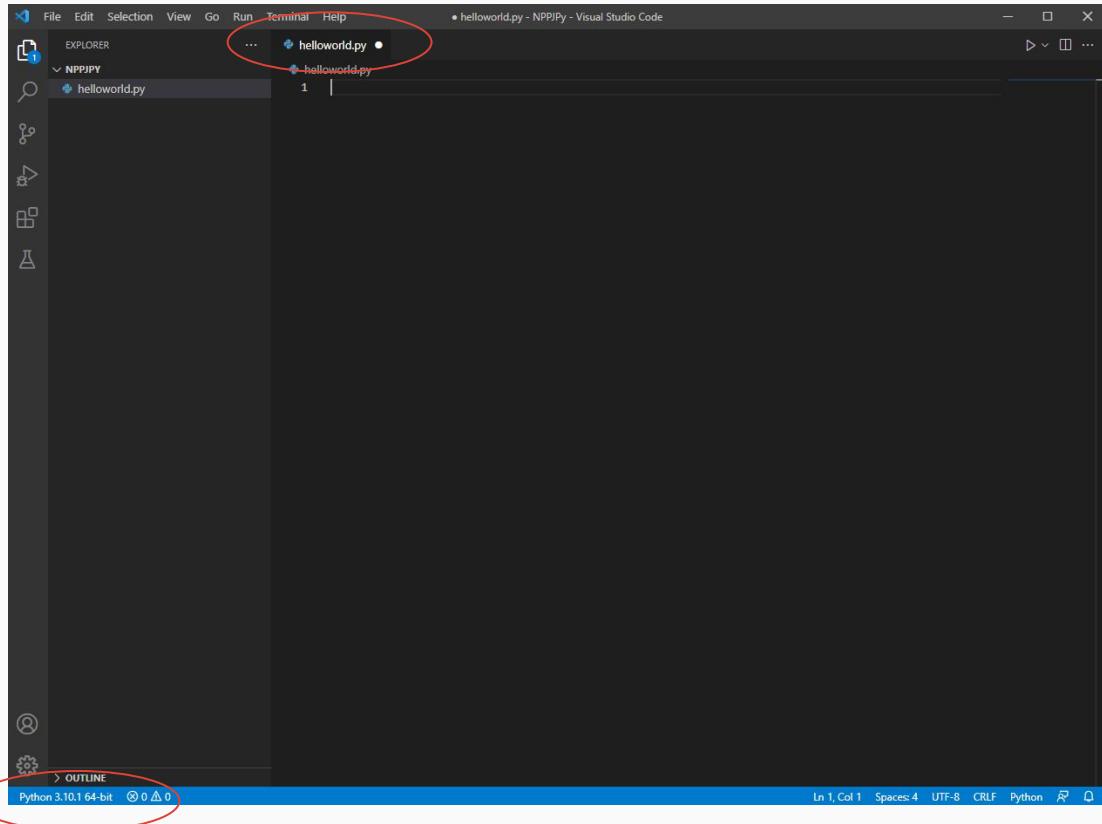
<https://code.visualstudio.com/docs/languages/python>

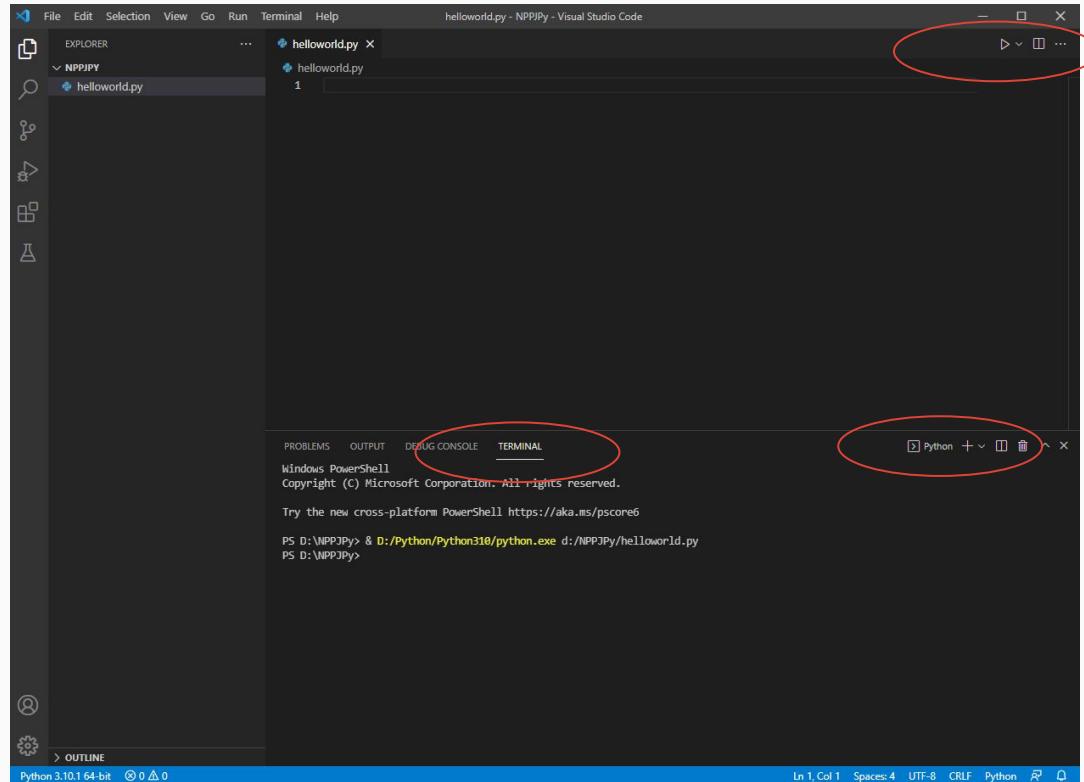
<https://code.visualstudio.com/download>

<https://www.python.org/downloads/>

<https://marketplace.visualstudio.com/items?itemName=ms-python.python>

Visual Studio != Visual Studio Code





A screenshot of the Visual Studio Code interface, showing a Python file named `helloworld.py`. The code editor displays the following line of code:

```
1 print
```

The word `print` is highlighted and has a yellow lightbulb icon above it, indicating a code completion or suggestion. A dropdown menu is open, listing several suggestions related to `print`:

- print
- property
- prepare_class
- ProcessLookupError
- Protocol
- Protocol
- _promote
- _ProtocolMeta
- ParamSpec
- ParamSpecArgs
- ParamSpecKwargs
- ChildProcessError

The rest of the code editor window is mostly blank, showing the dark theme of the IDE.

A screenshot of the Visual Studio Code interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar shows "helloworld.py - NPPJPy - Visual Studio Code". The left sidebar (EXPLORER) shows a folder "NPPJPy" containing a file "helloworld.py". The code editor displays the following Python script:

```
1 print("Hello world")
2
```

The bottom panel features a terminal window titled "TERMINAL" with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The terminal output is as follows:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\NPPJPy> & D:/Python/Python310/python.exe d:/NPPJPy/helloworld.py
Hello world
PS D:\NPPJPy>
```

The status bar at the bottom indicates "Python 3.10.1 64-bit" and "Ln 2, Col 1 Spaces: 4 UTF-8 CRLF Python".

The image shows two instances of Visual Studio Code side-by-side, both displaying a file named `helloworld.py`.

Left Instance:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** helloworld.py - NPPPy - Visual Studio Code.
- Explorer:** Shows a folder structure with `NPPPy` expanded, containing `helloworld.py`.
- Editor:** Displays the Python code:

```
1 print("Hello world")
2
```
- Contextual Menu (F12):** Options include Go to Definition, Go to Declaration, Go to Type Definition, Go to References, Peek, Find All References, Show Call Hierarchy, Rename Symbol, Change All Occurrences, Format Document, Format Document With..., Refactor..., Source Action..., Cut, Copy, Paste, Run Current File in Interactive Window, Run From Line in Interactive Window, Run Selection/Line in Interactive Window, Run to Line in Interactive Window, Run Python File in Terminal (highlighted), Run Selection/Line in Python Terminal, Shift+Enter, Sort Imports, Command Palette..., Ctrl+Shift+P.
- Bottom Status Bar:** Python 3.10.1 64-bit, In 2, Col 1, Sp 0.

Right Instance:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** helloworld.py - NPPPy - Visual Studio Code.
- Explorer:** Shows a folder structure with `NPPPy` expanded, containing `helloworld.py`.
- Editor:** Displays the Python code:

```
1 print("Hello world")
2
```
- Contextual Menu (F12):** Options include Open to the Side, Open With..., Reveal in File Explorer, Open in Integrated Terminal, Select for Compare, Cut, Copy, Copy Path, Copy Relative Path, Rename, Delete, F2, Run Current File in Interactive Window, Run Python File in Terminal (highlighted), Run Selection/Line in Python Terminal, Shift+Enter.
- Terminal:** Shows the command line output:

```
PS D:\NPPPy> & D:/Python/Python310/python.exe d:/NPPPy/helloworld.py
Hello world
PS D:\NPPPy>
```
- Bottom Status Bar:** Python 3.10.1 64-bit, In 2, Col 1, Spaces 4, UTF-8, CR LF, Python, R, Sp 0.

The screenshot shows a Visual Studio Code interface with a dark theme. In the top left, the menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar indicates "helloworld.py - NPPJPy - Visual Studio Code".

The Explorer sidebar on the left shows a project structure with a folder "NPPJPy" containing a file "helloworld.py". The code editor window displays the following Python script:

```
1 print("Hello world")
2
3 a = 5
4 b = 3
5 a = a + b
6 print(a)
7
```

Below the code editor is a terminal window titled "TERMINAL". It shows a Windows PowerShell session with the following output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\NPPJPy> & D:/Python/Python310/python.exe d:/NPPJPy/helloworld.py
Hello world
PS D:\NPPJPy> & D:/Python/Python310/python.exe d:/NPPJPy/helloworld.py
Hello world
8
PS D:\NPPJPy>
```

The status bar at the bottom of the terminal window shows "Python 3.10.1 64-bit" and "Ln 7, Col 1 Spaces:4 UTF-8 CRLF Python".

A screenshot of the Visual Studio Code interface, showing a context menu open over a line of Python code in the editor. The menu is triggered by a right-click on the line containing 'print("Hello world")'. The menu items are:

- Go to Definition F12
- Go to Declaration Shift+F12
- Go to Type Definition
- Go to References Shift+Alt+F12
- Peek >
- Find All References Shift+Alt+F12
- Show Call Hierarchy Shift+Alt+H
- Rename Symbol F2
- Change All Occurrences Ctrl+F2
- Format Document Shift+Alt+F
- Format Document With...
- Format Selection Ctrl+K Ctrl+F
- Refactor... Ctrl+Shift+R
- Source Action...

The bottom part of the menu shows additional options:

- Cut Ctrl+X
- Copy Ctrl+C
- Paste Ctrl+V
- Run Current File in Interactive Window
- Run From Line in Interactive Window
- Run Selection/Line in Interactive Window Shift+Enter **(highlighted)**
- Run To Line in Interactive Window
- Run Python File in Terminal
- Run Selection/Line in Python Terminal Shift+Enter
- Sort Imports
- Command Palette... Ctrl+Shift+P

The status bar at the bottom displays: Python 3.10.1 64-bit, 0 ▲ 0, Ln 3, Col 1 (5 selected), Spaces:4, UTF-8, CRLF, Python.

The screenshot shows the Visual Studio Code interface with a dark theme. On the left is the Explorer sidebar, which contains a tree view with a single item: 'NPPJPy' expanded, showing 'helloworld.py'. The main area is divided into two panes. The left pane displays the code editor with the file 'helloworld.py' open. The code is as follows:

```
1 print("Hello world")
2
3 a = 5
4 b = 3
5 a = a + b
6 print(a)
7
```

The right pane is the 'Interactive' terminal, titled 'Interactive-1'. It shows the output of the code execution: 'a = 5 ...'. At the bottom of the screen, there is a status bar with the following information: 'Python 3.10.1 64-bit', 'Ln 3, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', and a few small icons.

A screenshot of the Visual Studio Code interface. On the left is the Explorer sidebar with a tree view showing a folder named 'NPPJPy' containing a file 'helloworld.py'. The main editor area displays the following Python code:

```
1 print("Hello world")
2
3 a = 5
4 b = 3
5 a = a + b
6 print(a)
7
```

To the right of the editor is the 'Interactive-1' terminal window, which shows the output of the code execution:

```
a = 5 ...
```

At the bottom of the terminal window, there is a status bar with the text 'Jupyter Server: local'.

The screenshot shows two instances of Visual Studio Code running side-by-side. Both instances have the same file structure in the Explorer sidebar:

- File
- Edit
- Selection
- View
- Go
- Run
- Terminal
- Help

The Explorer sidebar shows a folder named "NPPJPy" containing a file named "helloworld.py". The code editor shows the following Python script:

```
1 print("Hello world")
2
3 a = 5
4 b = 3
5 a = a + b
6 print(a)
7
```

The terminal window titled "Interactive-1" shows the output of the script:

```
a = 5 ...
```

The second instance of Visual Studio Code shows the same setup. The code editor has the same script. The terminal window titled "Interactive-1" shows the output:

```
a = 5 ...
c = 7
[4] ✓ 0.4s
```

Jupyter notebooks

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

```
print("Hello world")
```

```
public class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello world");
    }
}
```

C++ - gcc kompajler

Java - kompajler, JVM

Python - kompajler, PVM

```
print("Hello world")  
  
a = 5  
b = 3  
a = a + b  
print(a)
```

c++ - funkcije, klase

java - klase

python ?

```
print("Hello world")  
  
a = 5  
b = 3  
a = a + b  
print(a)
```

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

<https://www.python.org/>

```
import this
```

The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

```
print("Hello world")  
  
a = 5  
b = 3  
a = a + b  
print(a)
```

Nema ;

Gde su tipovi?

```
print("Hello world")  
  
a = 5  
b = 3  
a = a + b  
print(a)  
  
a = "Dobar dan"  
print(a)
```

```
Hello world  
8  
Dobar dan
```

```
print("Hello world")  
  
a = 5  
b = 3  
a = a + b  
print(a)  
  
a = "Dobar dan"  
print(a)
```

```
Hello world  
8  
Dobar dan
```

Sve vrednosti su objekti
Objekti imaju nepromenljiv tip

Promenljive → imena, reference
Mogu da menjaju objekte “na koje pokazuju” (koje referenciraju)

```
print("Hello world")
```

```
a = 5
```

```
b = 3
```

```
a = a + b
```

```
print(a)
```

```
a = "Dobar dan"
```

```
print(a)
```

Svaki objekat ima svoj ID - nepromenljivo

```
print("Hello world")

a = 5
b = 3
a = a + b
print(a)

a = "Dobar dan"
print(a)

print(a, b)
print(id(a), id(b))
```

```
Hello world
8
Dobar dan
Dobar dan 3
2045520525040 2045515202864
```

```
print("Hello world")

a = 5
b = 3
a = a + b
print(a)

a = "Dobar dan"
print(a)

print(a, b)
print(id(a), id(b))
```

Kad pokrećemo skriptu sa programom

```
a = "Dobar dan"
b = 3
a, b
[13] ✓ 0.6s
... ('Dobar dan', 3)

a = "Dobar dan"
b = 3
id(a), id(b)
[14] ✓ 0.6s
... (2883614595440, 2883524624688)
```

Interaktivno

```
print("Hello world")

a = 5
b = 3
a = a + b
print(a)

a = "Dobar dan"
print(a)

print(a,b)
print(id(a), id(b))
print(type(a), type(b))
```

Svaki objekat ima svoj tip - nepromenljivo

```
print("Hello world")

a = 5
b = 3
a = a + b
print(a)

a = "Dobar dan"
print(a)

print(a, b)
print(id(a), id(b))

print(type(a), type(b))
```

```
Hello world
8
Dobar dan
Dobar dan 3
2021368215280 2021366890800
<class 'str'> <class 'int'>
```

```
print("Hello world")

a = 5
b = 3
a = a + b
print(a)

a = "Dobar dan"
print(a)

print(a, b)
print(id(a), id(b))
```

Svaki objekat ima svoju vrednost - može biti promenljivo
(int, string, tuple - immutable; list - mutable)

Objekti mogu imati i druge podatke

Tipovi

- numerički (int, float, complex)
- tekst sekvence - stringovi (str)
- bool
-
- sekvence (list, tuple, range)
- rečnik (dict)

//

- binarne sekvence (bytes, bytearray, memoryview)
- skupovi (set, frozenset)
- iteratori
- funkcije, metode
- klase, instance
- ...

Operation	Meaning	Operation	Result	Notes	Operation	Result	Notes
<	strictly less than	x + y	sum of x and y		x y	bitwise or of x and y	(4)
<=	less than or equal	x - y	difference of x and y		x ^ y	bitwise exclusive or of x and y	(4)
>	strictly greater than	x * y	product of x and y		x & y	bitwise and of x and y	(4)
>=	greater than or equal	x / y	quotient of x and y		x << n	x shifted left by n bits	(1)(2)
==	equal	x // y	floored quotient of x and y	(1)	x >> n	x shifted right by n bits	(1)(3)
!=	not equal	x % y	remainder of x / y	(2)	~x	the Operator (expressions...), the [expressions...], {key: value...}, {expressions...}	Description
is	object identity	-x	x negated		x[index]	Subscription, slicing, call, attribute reference	Binding or parenthesized expression, list display, dictionary display, set display
is not	negated object identity	+x	x unchanged		x[index:index]	Await expression	
		abs(x)	absolute value or magnitude of x	abs()	x(arguments...), x.attribute	Exponentiation [5]	
		int(x)	x converted to integer	(3)(6)	await x	Positive, negative, bitwise NOT	
		float(x)	x converted to floating point	(4)(6)	**	Multiplication, matrix multiplication, division, floor division, remainder [6]	
		complex(re, im)	a complex number with real part re, imaginary part im. im defaults to zero.	(6)	float()	*, @, /, //, %	
		c.conjugate()	conjugate of the complex number c		+, -	Addition and subtraction	
		divmod(x, y)	the pair (x // y, x % y)	(2)	<<, >>	Shifts	
		pow(x, y)	x to the power y	(5)	&	Bitwise AND	
		x ** y	x to the power y	(5)	^	Bitwise XOR	
						Bitwise OR	
					in, not in, is, is not, <, <=, >, >=, !=, ==	Comparisons, including membership tests and identity tests	
					not x	Boolean NOT	
					and	Boolean AND	
					or	Boolean OR	
					if - else	Conditional expression	
					lambda	Lambda expression	
					:=	Assignment expression	

<https://docs.python.org/3/library/stdtypes.html>

<https://docs.python.org/3/reference/expressions.html>