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

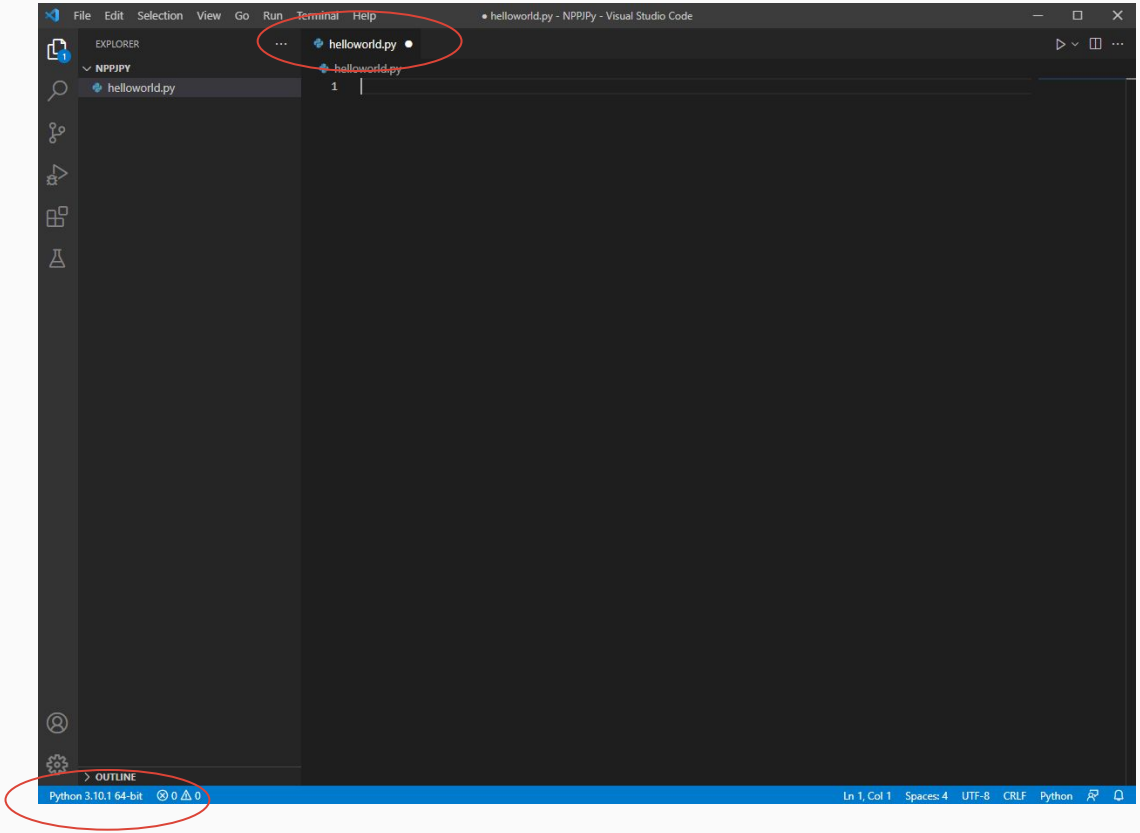
Python - izbor editora ili IDE programa proizvoljan

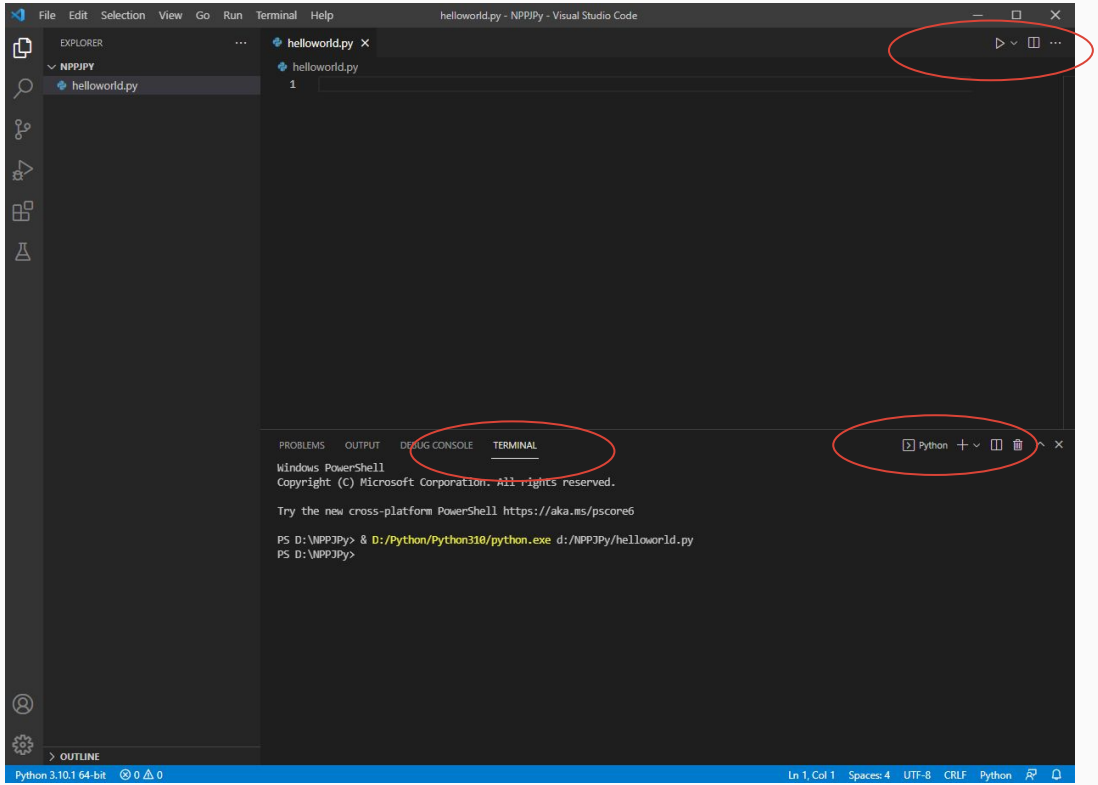
Visual Studio Code (editor)

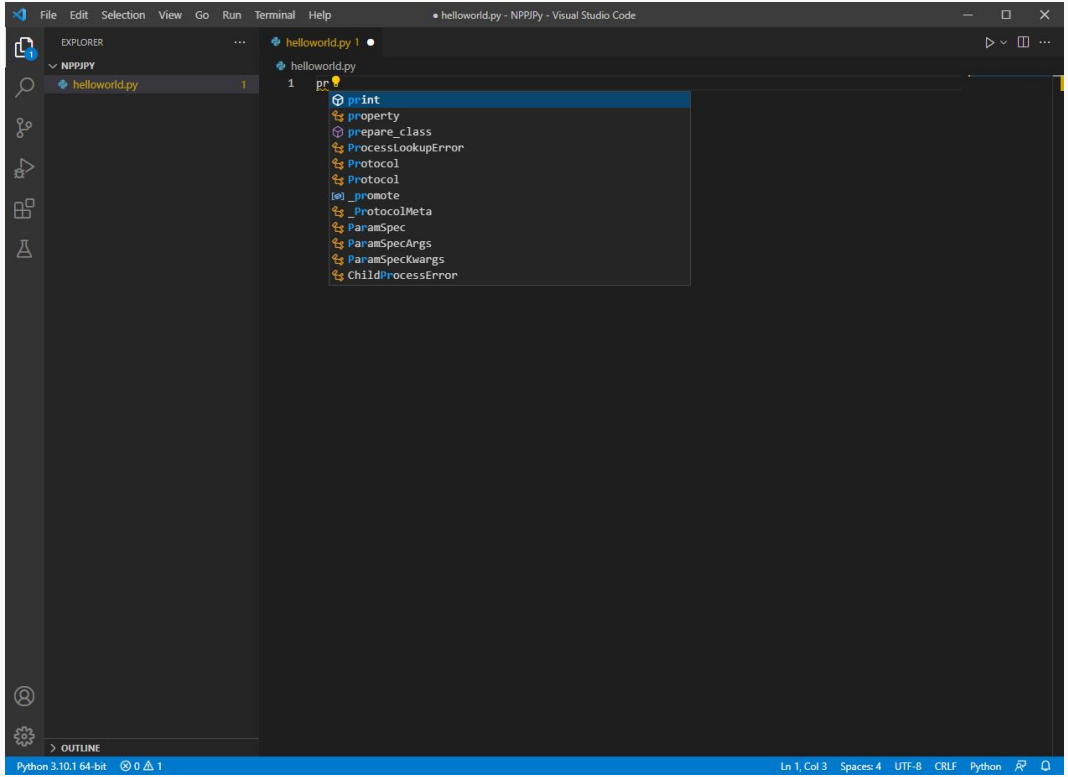
https://code.visualstudio.com/docs/languages/python

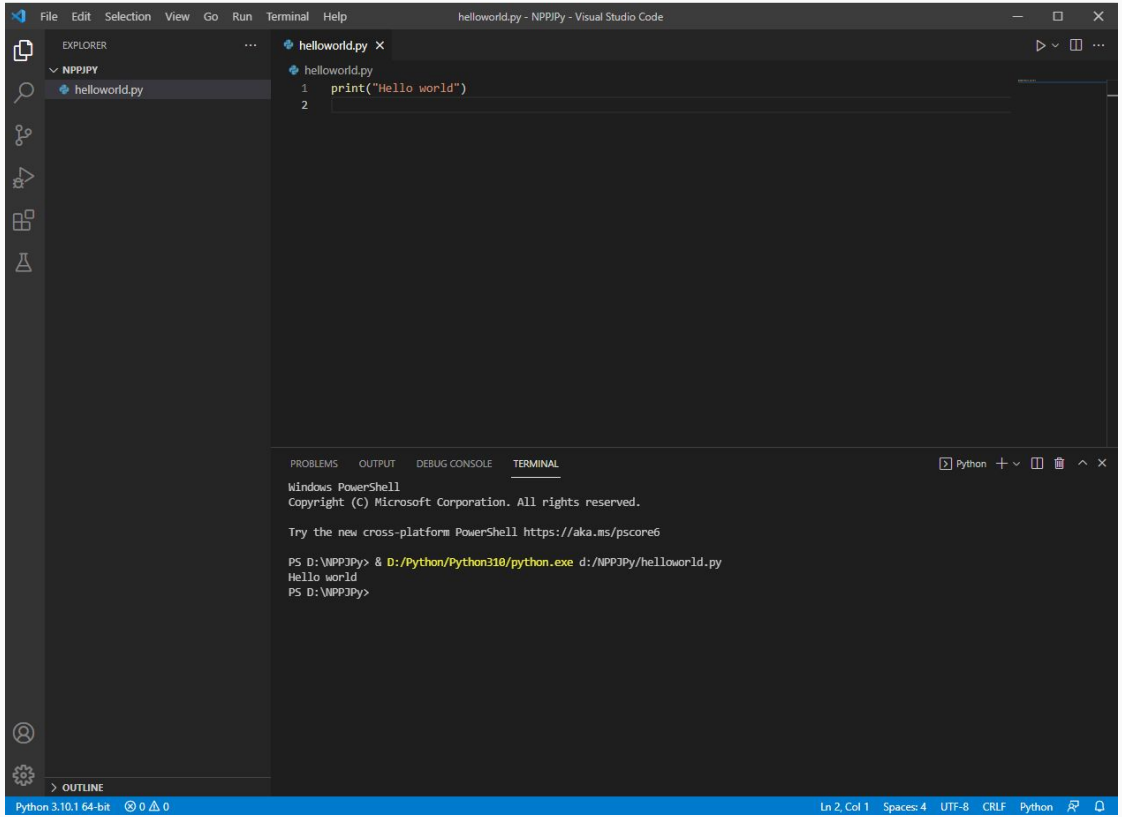
https://code.visualstudio.com/download
https://www.python.org/downloads/
https://marketplace.visualstudio.com/items?itemName=ms-python.python

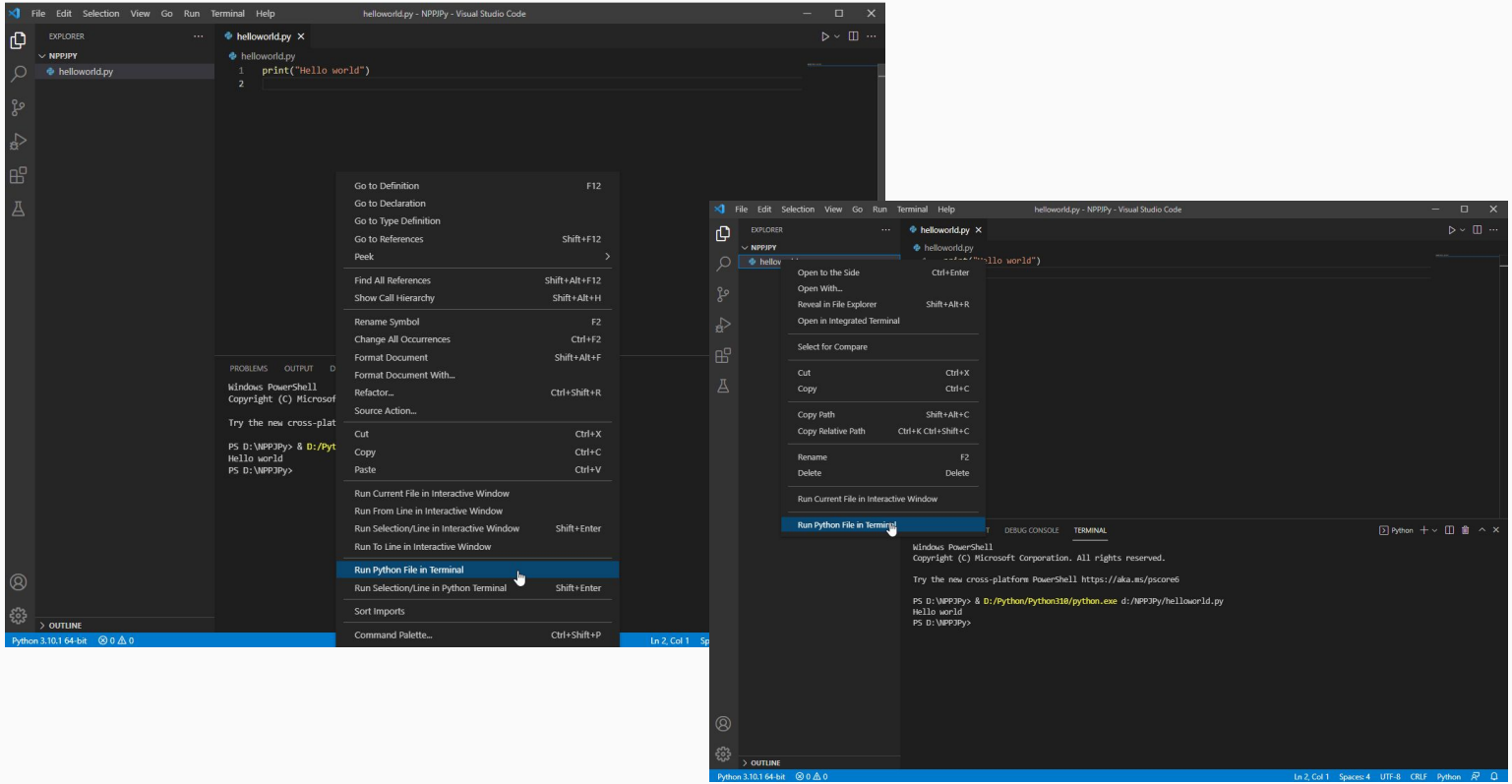
Visual Studio != Visual Studio Code

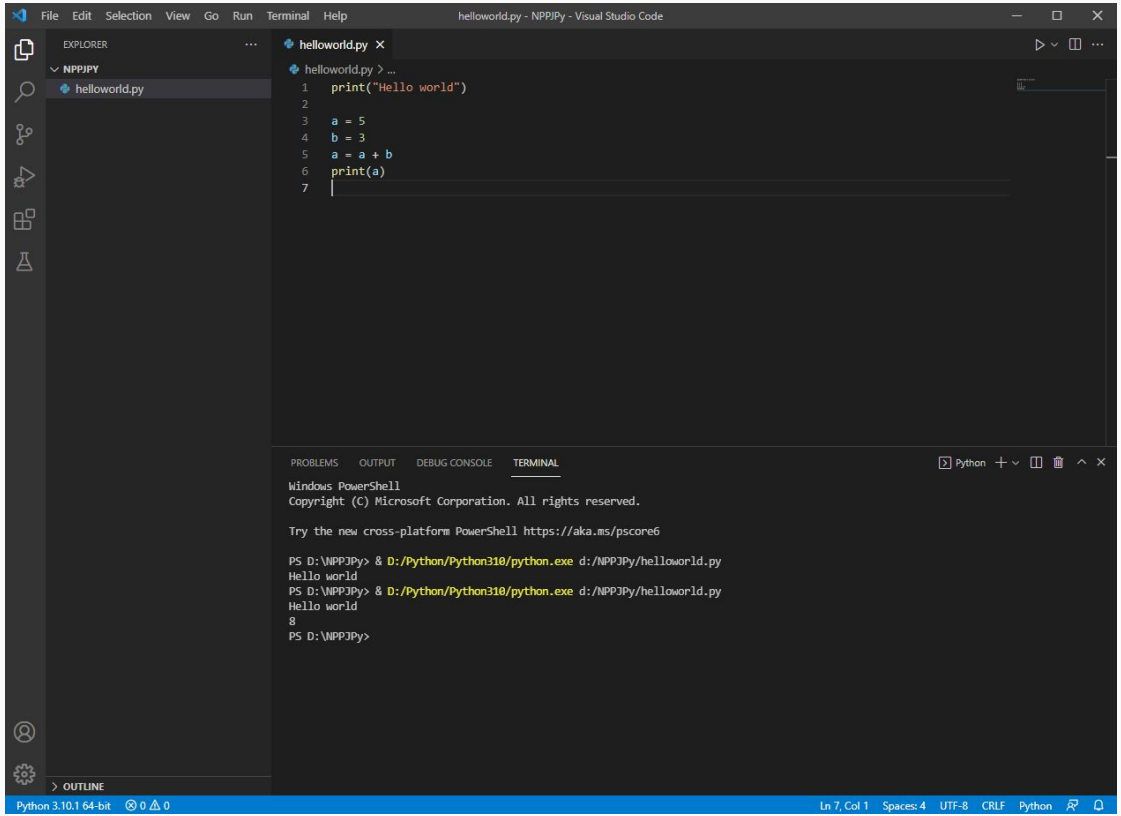

File Edit Selection View Go Run Terminal Help
• helloworld.py - NPPJPy - Visual Studio Code
EXPLORER
NPPJPY
helloworld.py
helloworld.py
helloworld.py
1
OUTLINE
Python 3.10.1 64-bit
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Python

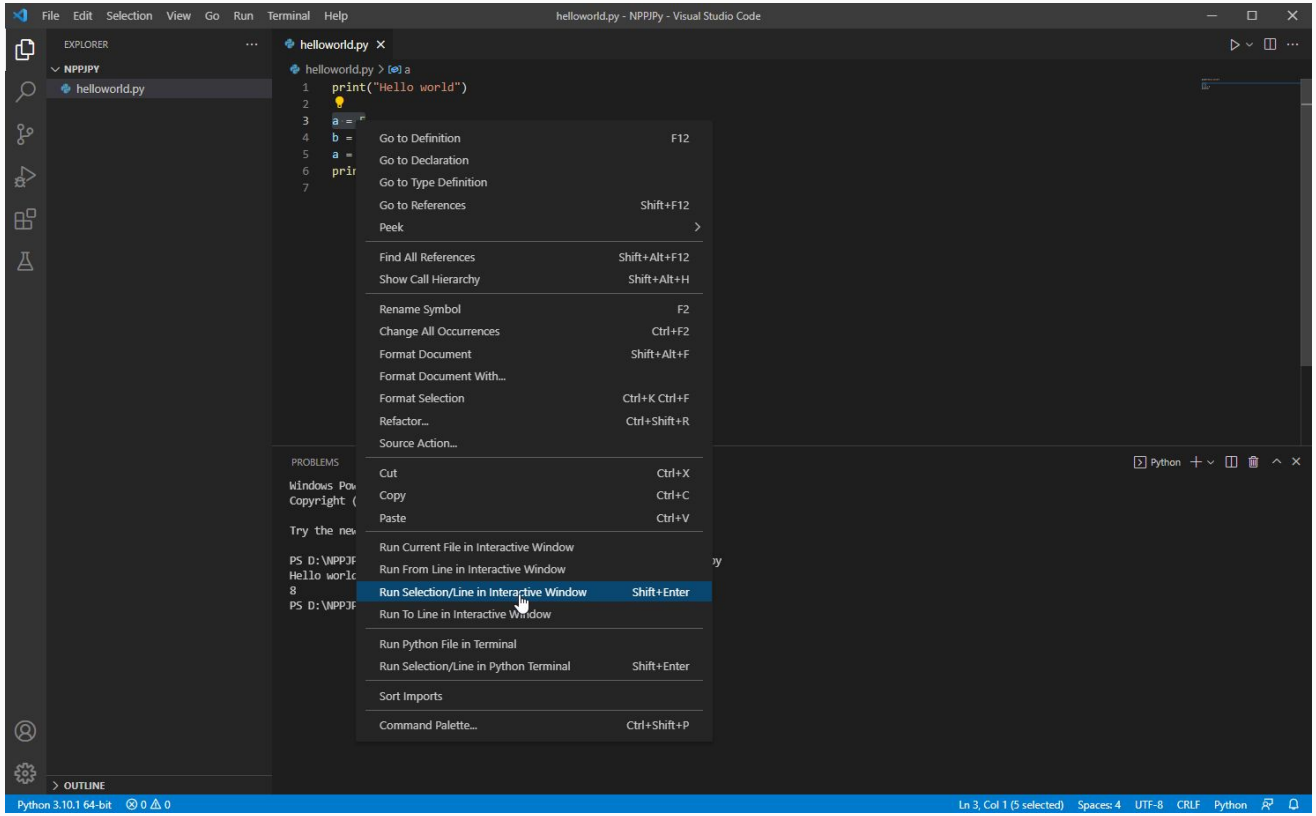

File Edit Selection View Go Run Terminal Help
helloworld.py - NPPJPy - Visual Studio Code
EXPLORER
NPPJPY
helloworld.py
helloworld.py
1
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS D:\NPPJPy> & D:/Python/Python310/python.exe d:/NPPJPy/helloworld.py
PS D:\NPPJPy>
OUTLINE
Python 3.10.1 64-bit
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python

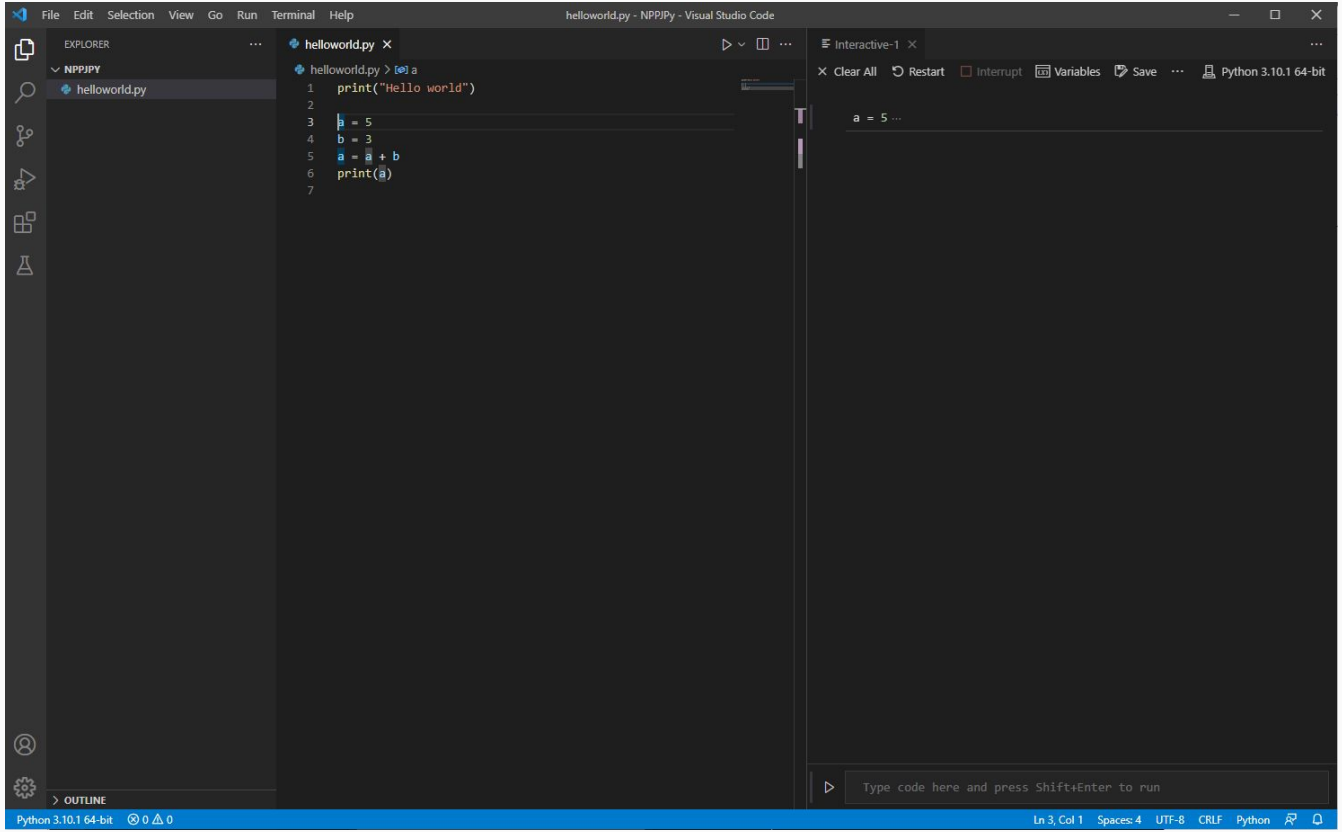

File Edit Selection View Go Run Terminal Help
• helloworld.py - NPPJPy - Visual Studio Code
EXPLORER
NPPJPY
helloworld.py 1
helloworld.py 1 •
helloworld.py
1 pr
print
property
prepare_class
ProcessLookupError
Protocol
Protocol
_promote
_ProtocolMeta
ParamSpec
ParamSpecArgs
ParamSpecKwargs
ChildProcessError
OUTLINE
Python 3.10.1 64-bit ⊗ 0 ⚠ 1
Ln 1, Col 3 Spaces: 4 UTF-8 CRLF Python

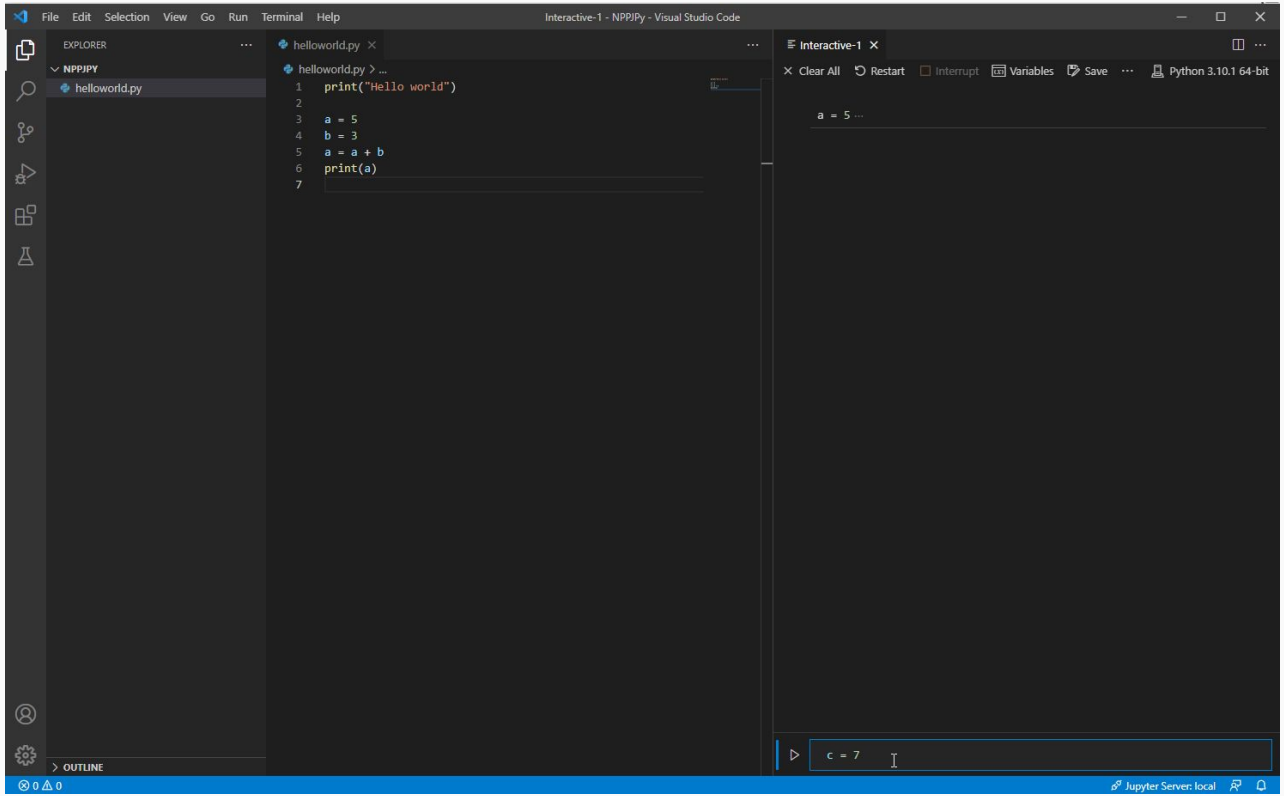

File Edit Selection View Go Run Terminal Help
helloworld.py - NPPJPy - Visual Studio Code
EXPLORER
NPPJPY
helloworld.py
helloworld.py
1 print("Hello world")
2
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS D:\NPPJPy> & D:/Python/Python310/python.exe d:/NPPJPy/helloworld.py
Hello world
PS D:\NPPJPy>
OUTLINE
Python 3.10.1 64-bit
Ln 2, Col 1 Spaces: 4 UTF-8 CRLF Python

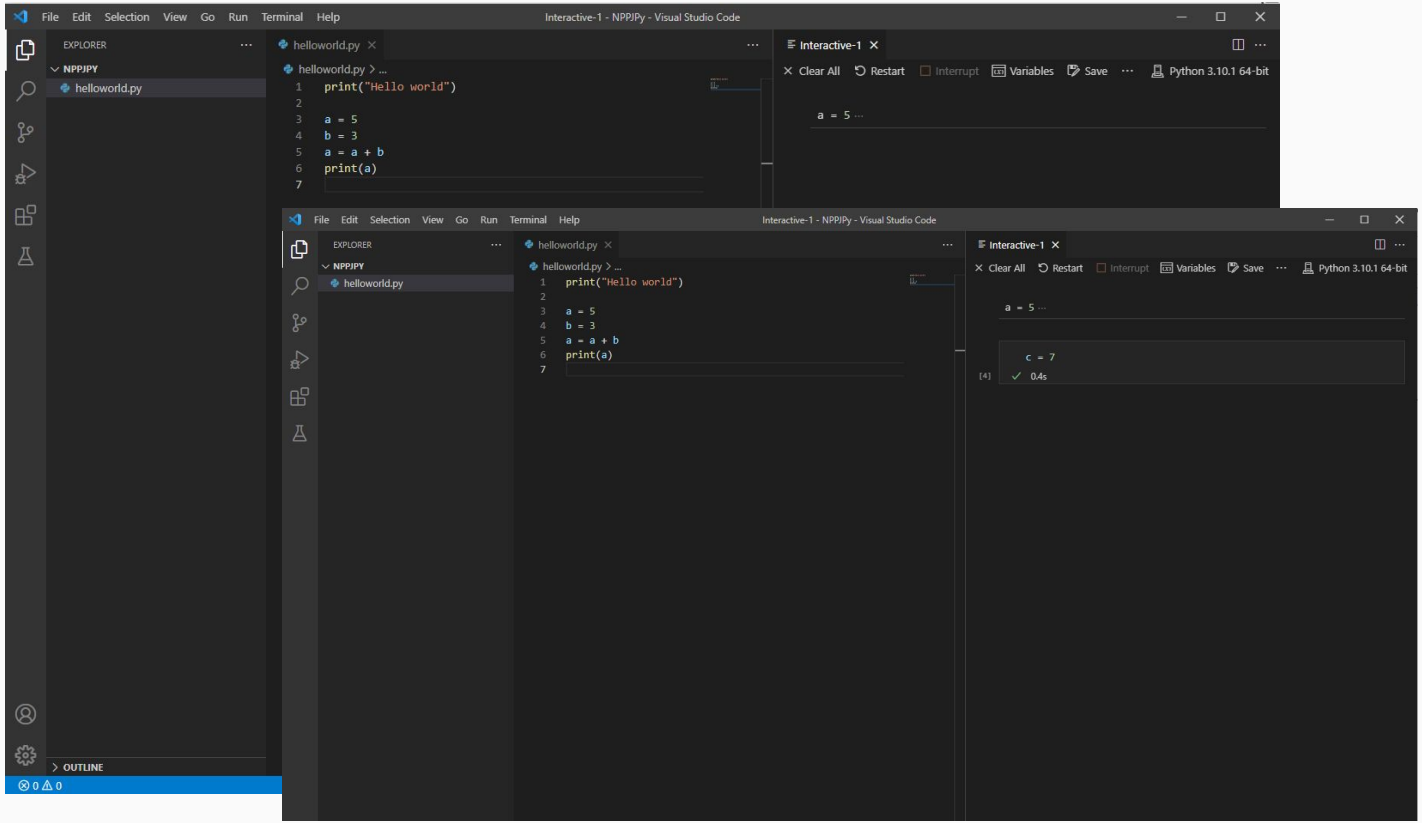

File Edit Selection View Go Run Terminal Help
helloworld.py - NPPJPy - Visual Studio Code
EXPLORER
NPPJPY
helloworld.py
print("Hello world")
Go to Definition F12
Go to Declaration
Go to Type Definition
Go to References Shift+F12
Peek
Find All References Shift+Alt+F12
Show Call Hierarchy Shift+Alt+H
Rename Symbol F2
Change All Occurrences Ctrl+F2
Format Document Shift+Alt+F
Format Document With...
Refactor... Ctrl+Shift+R
Source Action...
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
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
Open to the Side Ctrl+Enter
Open With...
Reveal in File Explorer Shift+Alt+R
Open in Integrated Terminal
Select for Compare
Cut Ctrl+X
Copy Ctrl+C
Copy Path Shift+Alt+C
Copy Relative Path Ctrl+K Ctrl+Shift+C
Rename F2
Delete Delete
Run Current File in Interactive Window
Run Python File in Terminal
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
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>
OUTLINE
Python 3.10.1 64-bit
Ln 2, Col 1 Spaces: 4 UTF-8 CRLF Python


File Edit Selection View Go Run Terminal Help
helloworld.py - NPPJPy - Visual Studio Code
EXPLORER
NPPJPY
helloworld.py
helloworld.py
helloworld.py > ...
1 print("Hello world")
2
3 a = 5
4 b = 3
5 a = a + b
6 print(a)
7
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python
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>
OUTLINE
Python 3.10.1 64-bit
Ln 7, Col 1 Spaces: 4 UTF-8 CRLF Python


File Edit Selection View Go Run Terminal Help
helloworld.py - NPPJPy - Visual Studio Code
EXPLORER
NPPJPY
helloworld.py
helloworld.py
print("Hello world")
Go to Definition F12
Go to Declaration
Go to Type Definition
Go to References Shift+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...
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
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
PROBLEMS
Python
OUTLINE
Python 3.10.1 64-bit
Ln 3, Col 1 (5 selected) Spaces: 4 UTF-8 CRLF Python


File Edit Selection View Go Run Terminal Help
helloworld.py - NPPJPy - Visual Studio Code
EXPLORER
NPPJPY
helloworld.py
helloworld.py
helloworld.py > a
1 print("Hello world")
2
3 a = 5
4 b = 3
5 a = a + b
6 print(a)
7
Interactive-1
Clear All
Restart
Interrupt
Variables
Save
Python 3.10.1 64-bit
a = 5 ...
Type code here and press Shift+Enter to run
OUTLINE
Python 3.10.1 64-bit
Ln 3, Col 1
Spaces: 4
UTF-8
CRLF
Python


File Edit Selection View Go Run Terminal Help
Interactive-1 - NPPJPy - Visual Studio Code
EXPLORER
NPPJPY
helloworld.py
helloworld.py
print("Hello world")
a = 5
b = 3
a = a + b
print(a)
Interactive-1
Clear All
Restart
Interrupt
Variables
Save
Python 3.10.1 64-bit
a = 5 ...
c = 7
OUTLINE
Jupyter Server: local



Jupyter notebooks

```cpp
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

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

```java
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

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
`<`	strictly less than
`<=`	less than or equal
`>`	strictly greater than
`>=`	greater than or equal
`==`	equal
`!=`	not equal
`is`	object identity
`is not`	negated object identity

Operation	Result	Notes	
`x + y`	sum of *x* and *y*		
`x - y`	difference of *x* and *y*		
`x * y`	product of *x* and *y*		
`x / y`	quotient of *x* and *y*		
`x // y`	floored quotient of *x* and *y*	(1)	
`x % y`	remainder of `x / y`	(2)	
`-x`	*x* negated		
`+x`	*x* unchanged		
`abs(x)`	absolute value or magnitude of *x*		`abs()`
`int(x)`	*x* converted to integer	(3)(6)	`int()`
`float(x)`	*x* converted to floating point	(4)(6)	`float()`
`complex(re, im)`	a complex number with real part *re*, imaginary part *im*. *im* defaults to zero.	(6)	`complex()`
`c.conjugate()`	conjugate of the complex number *c*		
`divmod(x, y)`	the pair `(x // y, x % y)`	(2)	`divmod()`
`pow(x, y)`	*x* to the power *y*	(5)	`pow()`
`x ** y`	*x* to the power *y*	(5)	

Operation	Result	Notes
`x | y`	bitwise *or* of *x* and *y*	(4)
`x ^ y`	bitwise *exclusive or* of *x* and *y*	(4)
`x & y`	bitwise *and* of *x* and *y*	(4)
`x << n`	*x* shifted left by *n* bits	(1)(2)
`x >> n`	*x* shifted right by *n* bits	(1)(3)
`~x`	the	

Operator	Description
`(expressions...)`, `[expressions...]`, `{key: value...}`, `{expressions...}`	Binding or parenthesized expression, list display, dictionary display, set display
`x[index]`, `x[index:index]`, `x(arguments...)`, `x.attribute`	Subscription, slicing, call, attribute reference
`await x`	Await expression
`**`	Exponentiation [5]
`+x`, `-x`, `~x`	Positive, negative, bitwise NOT
`*`, `@`, `/`, `//`, `%`	Multiplication, matrix multiplication, division, floor division, remainder [6]
`+`, `-`	Addition and subtraction
`<<`, `>>`	Shifts
`&`	Bitwise AND
`^`	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