



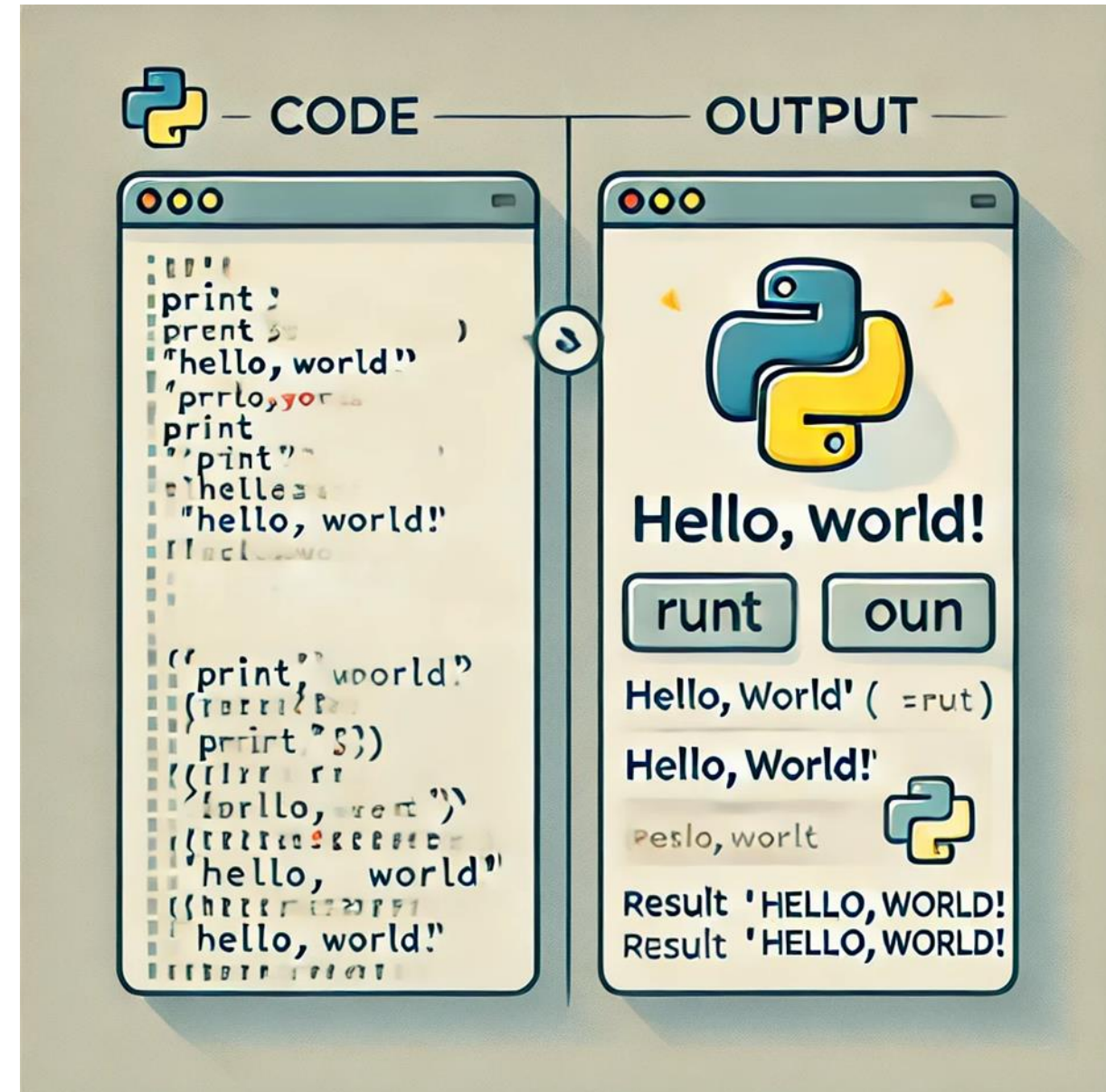
Interactive Shell

```
>>> print("Hello World!")
```

command

```
Hello World!
```

result



```
>>> print("salam")
```

```
salam
```

```
>>> print('saalaam')
```

```
saalaam
```

```
print("an example input")
```

```
print("an example input")
```

```
اسم تابع ( ورودی "an example input")
```

FunctionName(input)

```
print("an example input")
```

```
type("an example input")
```

```
>>> type("salam")  
<class 'str'>  
>>> type('salam')  
<class 'str'>
```

```
>>> type("salam")  
<class 'str'>  
>>> type('salam')  
<class 'str'>
```

str  string  رشته

```
>>> type(3)
```

```
>>> type(3)  
<class 'int'>
```

int → integer → عدد صحيح


```
>>> type("salam")  
<class 'str'>  
? >>> type(salam)
```



نمایش سر کلاس در ترمینال



```
>>> type(salam)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'salam' is not defined
```

```
>>> type(3.14)
<class 'float'>
```

float → floating-point number → عدد اعشاری

عدد با نقطه شناور

```
>>> type(-3)
```

```
>>> type(-3)  
<class 'int'>
```

```
>>> type(0)  
<class 'int'>
```

```
>>> type(-2.71)
<class 'float'>
```



```
>>> type(0.0)
```

```
>>> type(0.0)
<class 'float'>
```

```
>>> type(6.0)
<class 'float'>
```

```
type("3.14")
```



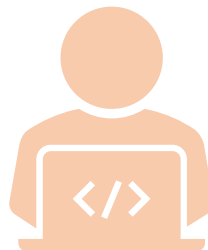
```
>>> type("3.14")  
<class 'str'>
```

str

int

float

چند اسلاید بعدی فقط افرادی که قبلا برنامه نویسی داشته اند



زبان‌های دیگر	نوع داده	توضیح	جایگزین در پایتون
C/C++/Java	short	عدد صحیح کوتاه (معمولاً 2 بایت)	پایتون به‌طور خودکار اندازه‌ی عدد صحیح را مدیریت می‌کند.
C/C++/Java	long	عدد صحیح بزرگ‌تر (معمولاً 4 یا 8 بایت)	مشابه <code>int</code> در پایتون
C/C++/Java	long long	عدد صحیح بسیار بزرگ (معمولاً 8 بایت)	پایتون از <code>int</code> با دقت بی‌نهایت پشتیبانی می‌کند.
C/C++/Java	unsigned	عدد صحیح بدون علامت (فقط مقادیر مثبت)	کتابخانه‌هایی مثل <code>numpy</code> این قابلیت را فراهم می‌کنند.
C++/Java	char	یک کاراکتر (معمولاً 1 بایت)	در پایتون رشته‌ها استفاده می‌شوند.
C/C++/Java	double	عدد اعشاری دقیق‌تر (معمولاً 8 بایت)	<code>float</code> پایتون از دقت دوبرابر (<code>double</code>) استفاده می‌کند.
++C	bool	متغیر منطقی (صحیح/غلط)	پایتون <code>True</code> و <code>False</code> را پشتیبانی می‌کند.
++C	wchar_t	کاراکترهای گسترده برای ذخیره یونیکد	در پایتون رشته‌ها یونیکد هستند.

```
import numpy as np

x = np.int16(100)  # عدد صحیح 16 بیتی
y = np.uint8(255)  # عدد صحیح بدون علامت 8 بیتی
z = np.float32(3.14)  # عدد اعشاری 32 بیتی
```

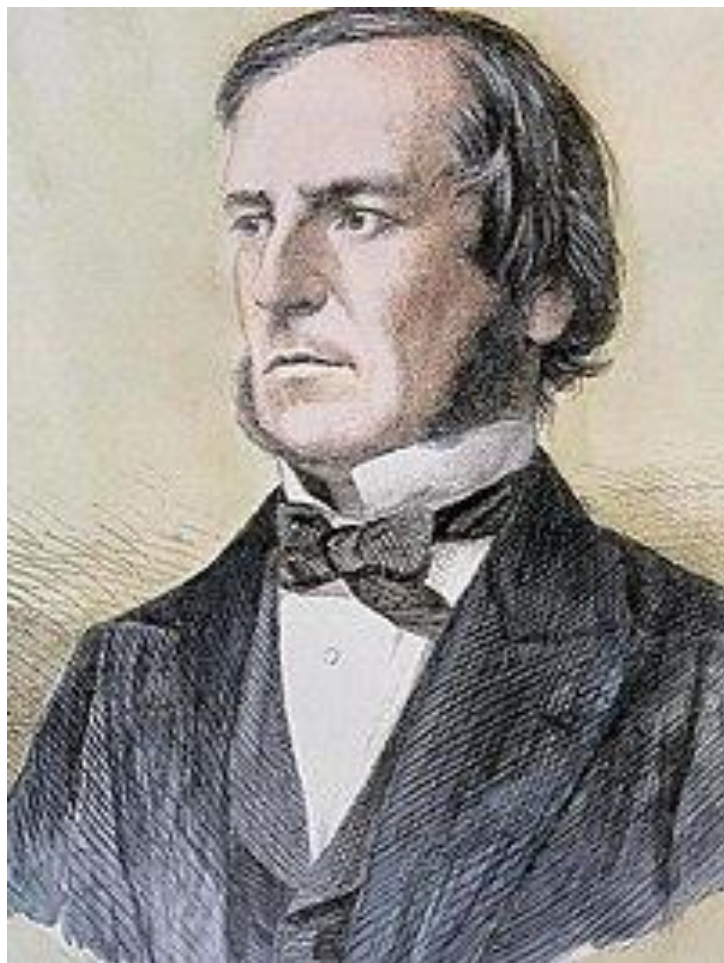
```
from ctypes import c_int, c_long, c_double

x = c_int(10)      # عدد صحیح 4 بایتی
y = c_long(1000)   # عدد صحیح بزرگ‌تر
z = c_double(3.14) # عدد اعشاری دقیق‌تر
```

```
>>> type(True)
<class 'bool'>
>>> type(False)
<class 'bool'>
```

bool \longrightarrow boolean \longrightarrow بولین یا منطقی





جرج بول

George Boole

جبر بولی پایه محاسبات کامپیوتری شد

او یکی از بنیان‌گذاران علم کامپیوتر است، گرچه
در زمان او هنوز کامپیوتر اختراع نشده بود.

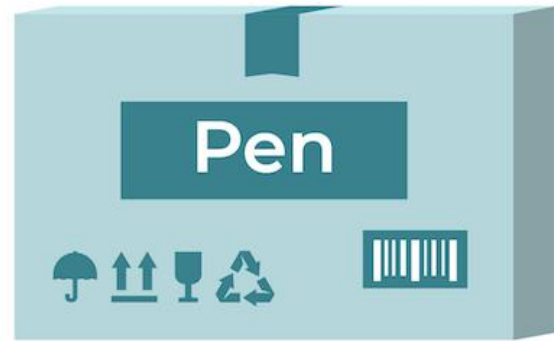
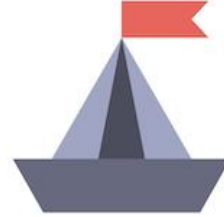
```
>>> type(false)
```

```
>>> type(false)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'false' is not defined. Did you
mean: 'False'?
```

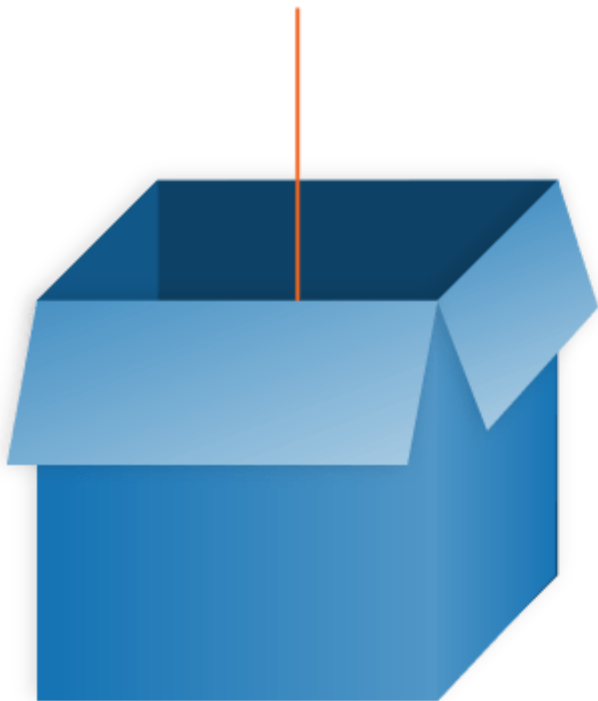
```
>>> type(true)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'true' is not defined. Did you m
ean: 'True'?
```


پایتون یک زبان **case-sensitive** است

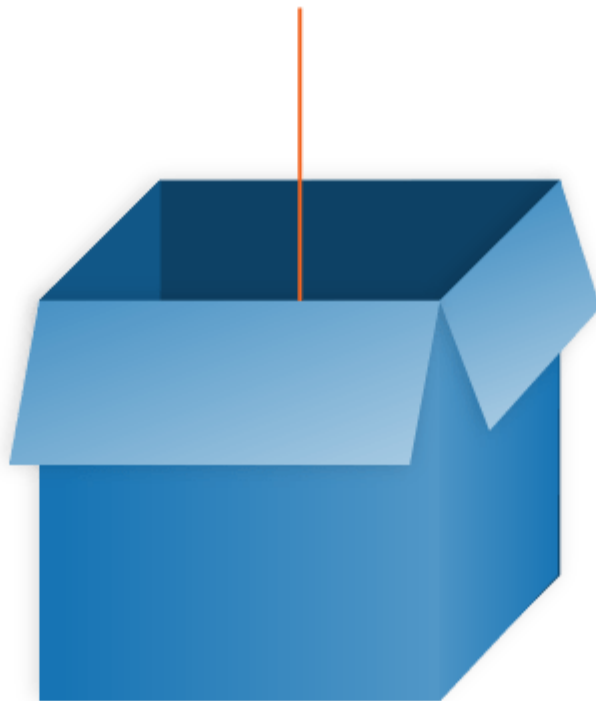
```
>>> print("salam")
salam
>>> Print("salam")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Print' is not defined. Did you
mean: 'print'?
>>> PRINT("salam")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'PRINT' is not defined
```



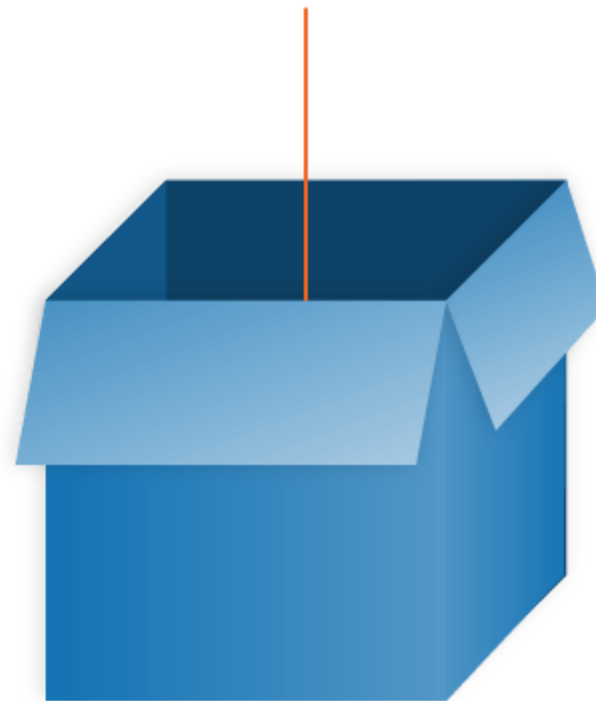
"Bob"

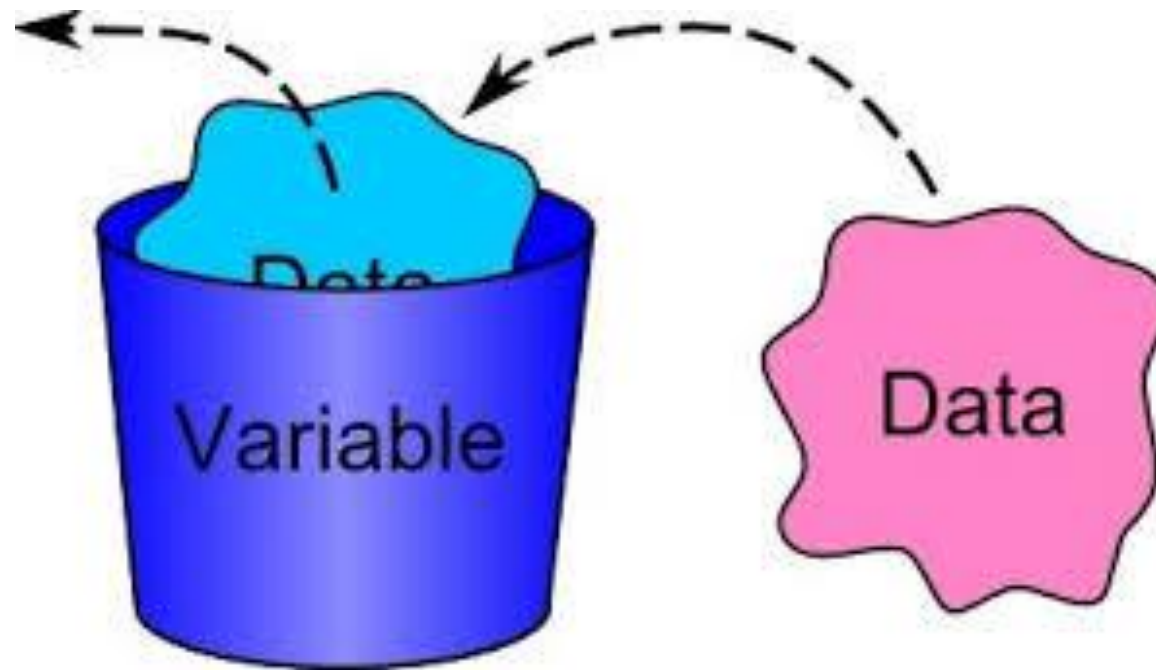


true



35





variable → متغیر

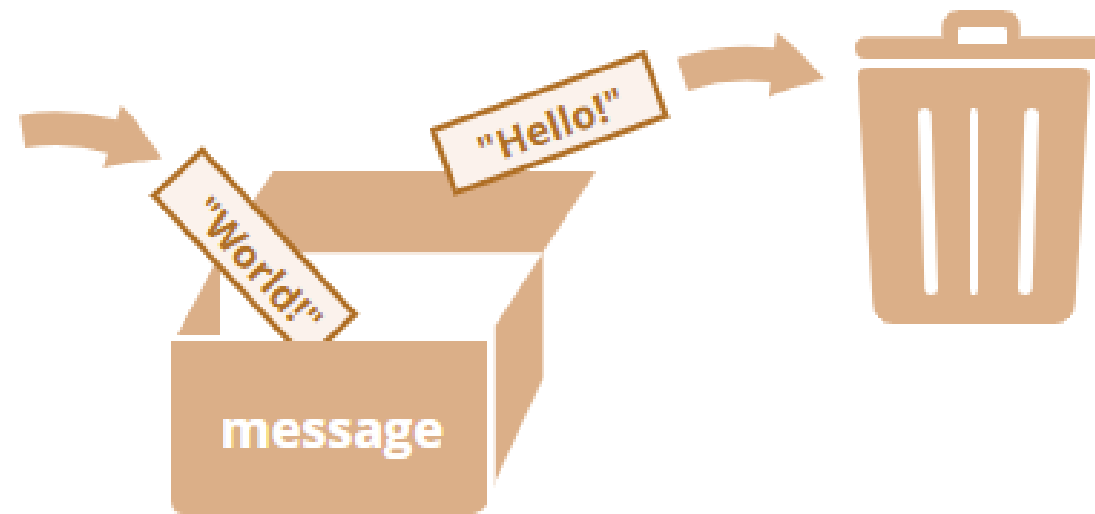
Variable Value

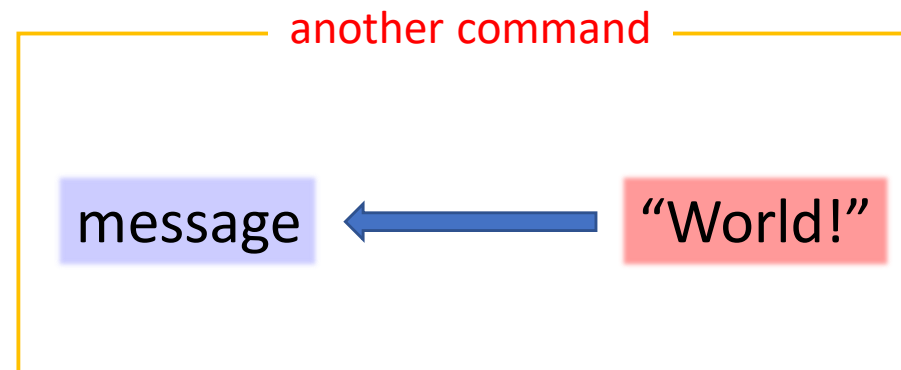
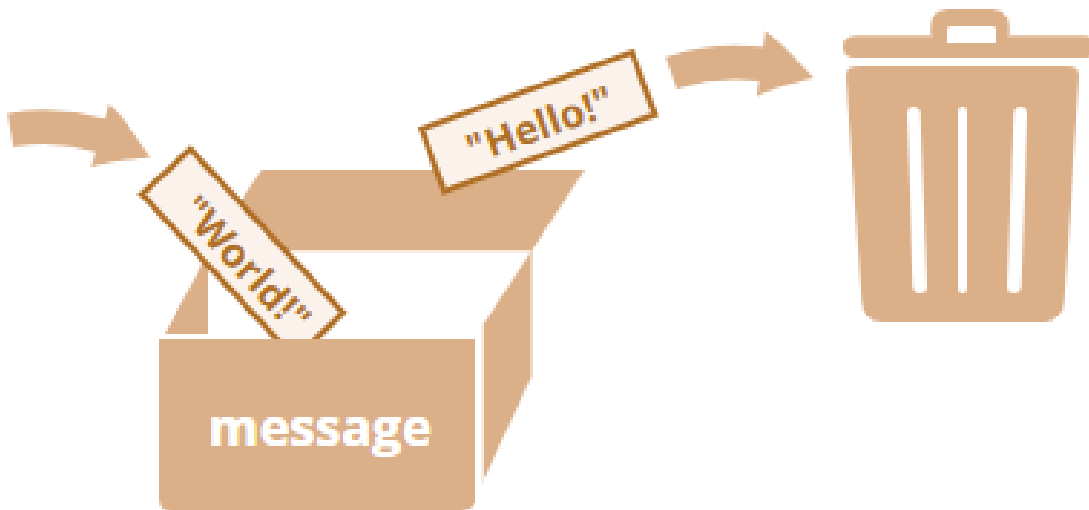
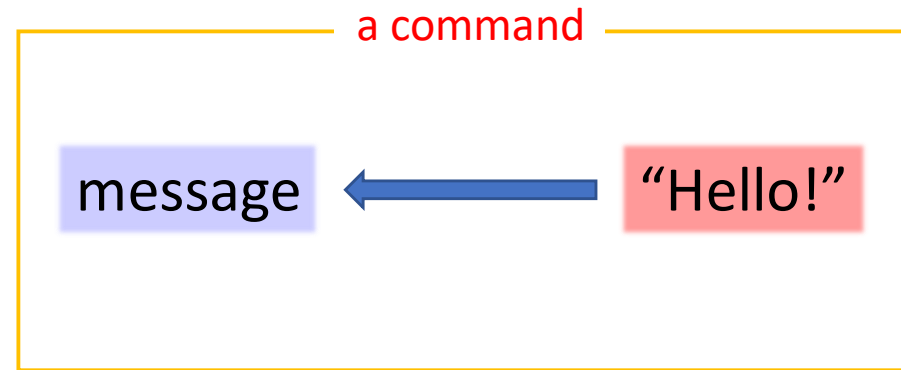
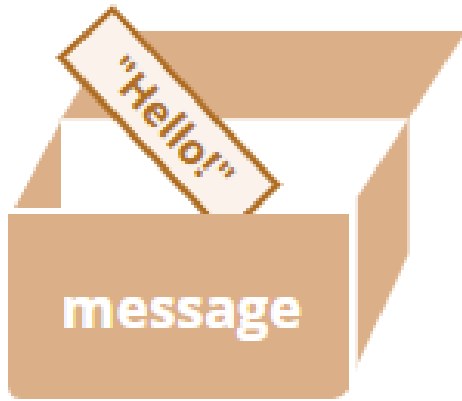
10

Num

Variable Name









زبان APL

message ← "Hello!"

Python & others

message = "Hello!"

"Bob"



name = "Bob"

True

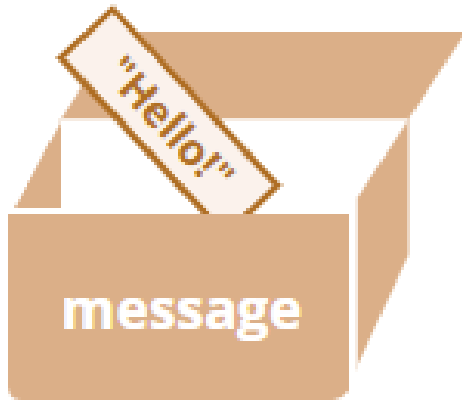


winner = True

35



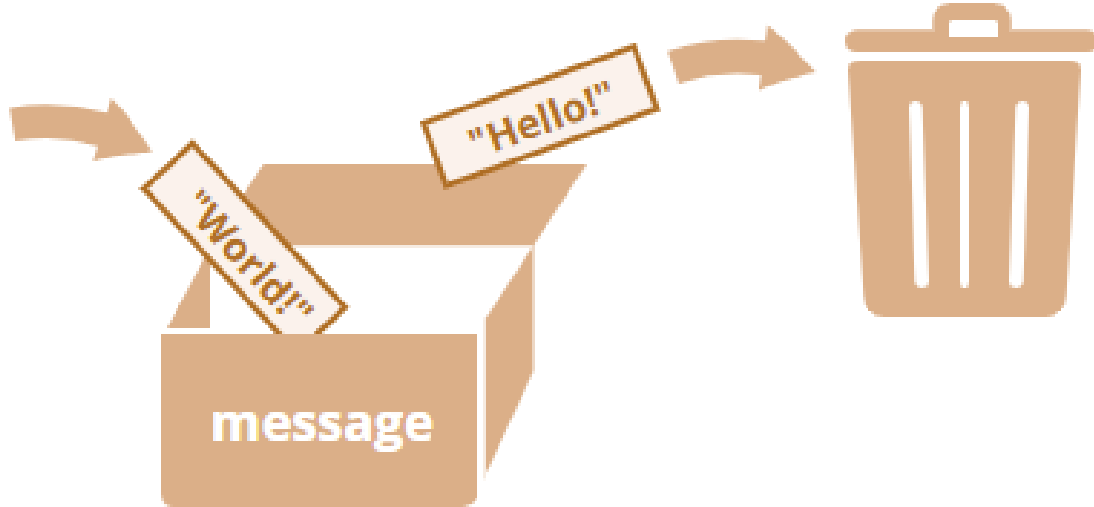
score = 35



message = "Hello!"



a python command



message = "World!"



a python command

score = 35



“score” = 35

```
>>> "score" = 35
      File "<stdin>", line 1
        "score" = 35
        ^^^^^^^
```

SyntaxError: cannot assign to literal here.

35 = score

```
>>> 35 = score
      File "<stdin>", line 1
        35 = score
          ^^
```

SyntaxError: cannot assign to literal here.

$$a = 2$$

$$a=2$$

```
>>> a = 2 ✓  
>>> a=2 ✓
```

PEP

Python Enhancement Proposal

پیشنهاد بهبود پایتون

این اسناد، ایده‌ها و پیشنهادهای رسمی هستند که به پیشرفت زبان پایتون کمک می‌کنند.

PEP ها می توانند شامل:

- معرفی ویژگی های جدید.
- استانداردهای سبک کدنویسی.
- تغییرات در فرآیندها یا قوانین توسعه زبان.

اهمیت PEP

- یکپارچگی و نظم: PEP ها باعث می شوند توسعه پایتون با یک ساختار منظم و شفاف پیش برود.
- مستندسازی تصمیمات: هر تغییری که در پایتون اتفاق می افتد، با جزئیات در یک PEP توضیح داده می شود. این کار تاریخچه ای ارزشمند از تصمیمات زبان ایجاد می کند.
- همراهی جامعه: PEP ها فرصتی برای جامعه پایتون فراهم می کنند تا ایده ها را بررسی کرده و به بهبود زبان کمک کنند.

PEP شبیه به قانون اساسی یا دستورالعمل‌های رسمی است.

همان‌طور که قوانین به یک جامعه نظم می‌دهند، PEP ها به پایتون نظم و ساختار می‌دهند.

PEP 8

راهنمای سبک کدنویسی

PEP 20

اصول طراحی پایتون

Zen of Python

```
import this
```

P E P - 8

Your guide to beautiful Python Code



`a=2` ❌ Not recommended

`a = 2` ✅ Recommended



python-formatter

```
>>> x = 10
>>> x = "hello"
>>> x
'hello'
```

✗ Not recommended

```
>>> a = 2
```

```
>>> a
```

```
2
```

✓ Recommended

```
>>> print(a)
```

```
2
```

main.py



1 a = 2

2 a

3 |

Python Tutor

main.py



```
1 print(x)
2 x = 2
3 |
```

Ln: 3, Col: 1



Run



Share



Command Line Arguments



```
Traceback (most recent call last):
  File "main.py", line 1, in <module>
    print(x)
NameError: name 'x' is not defined
```

```
>>> 1variable = 10
```

```
>>> 1variable = 10
      File "<stdin>", line 1
        1variable = 10
        ^
SyntaxError: invalid decimal literal
```

نام متغیر باید با حرف یا زیرخط (_) شروع شود.

def = 5

```
>>> def = 5
      File "<stdin>", line 1
        def = 5
          ^
SyntaxError: invalid syntax
```

از کلمات کلیدی پایتون استفاده نکنید.





```
myvar = "John"
```

```
my_var = "John"
```

```
_my_var = "John"
```

```
myVar = "John"
```

```
MYVAR = "John"
```

```
myvar2 = "John"
```



```
2myvar = "John"
```

```
my-var = "John"
```

```
my var = "John"
```


نام‌ها توصیفی و معنادار باشند.

age = 25

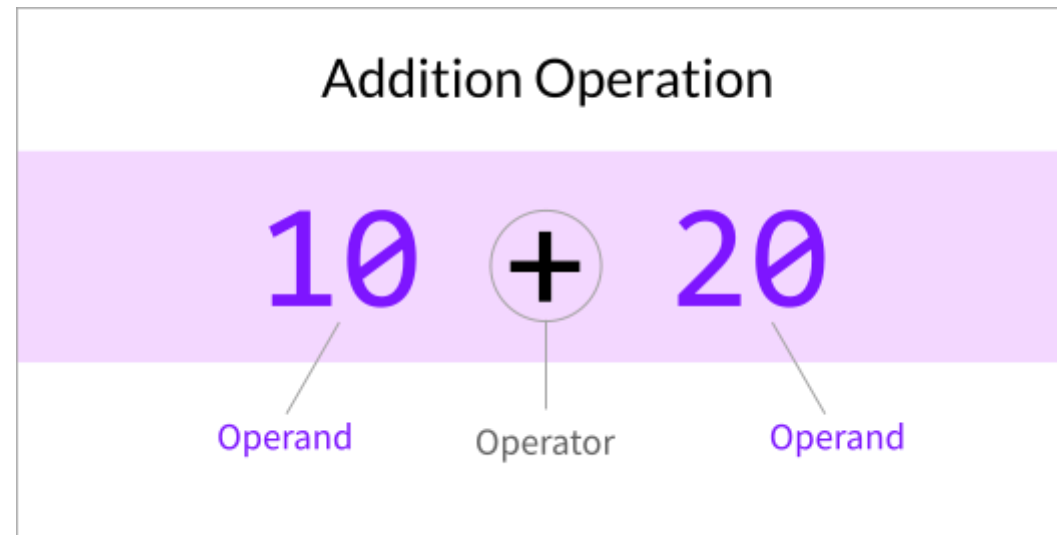
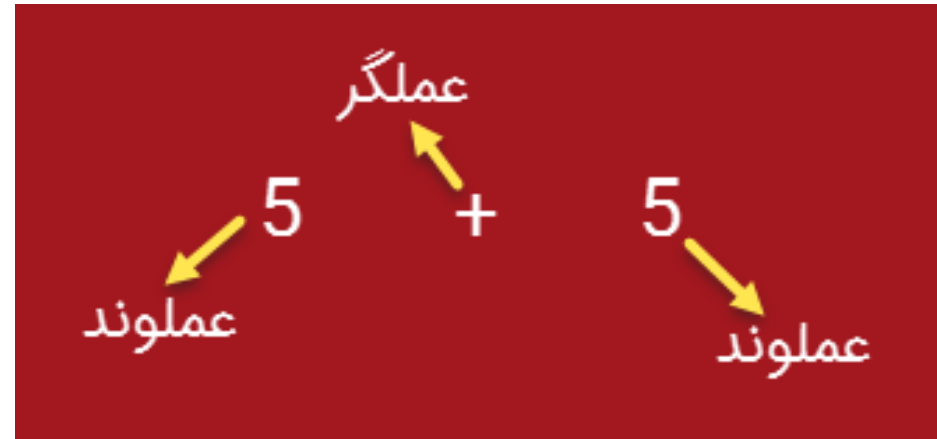
total_price = 100.5



```
x = 10  
y = 20  
z = x + y
```



```
item_price = 10  
tax = 20  
total_price = item_price + tax
```



```
>>> 1 + 3
```

```
4
```

```
>>> 8 - 1
```

```
7
```

```
>>> 120-23
```

```
97
```

main.py

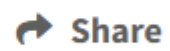


```
1 print(1 + 3)
2 print(8 - 1)
3 print(120 - 23)
```

Ln: 3, Col: 15



Run



Share



4



7



97

main.py



```
1 x = 10
2 y = 5
3 print(x + y)
```

Ln: 5, Col: 1



Run



Share



15

main.py



```
1 x = 10
2 y = 5
3 result = x + y
4 print(result)
5 result = 5 + 10
6 print(result)
```

Ln: 8, Col: 1



Run



Share



15



15

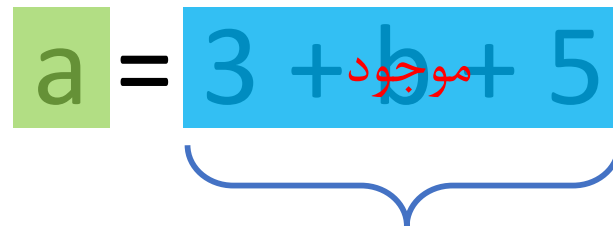


عملگرهای حسابی

Arithmetic Operators

Operators	Meaning	Example	Result
+	Addition	$4 + 2$	6
-	Subtraction	$4 - 2$	2
*	Multiplication	$4 * 2$	8
/	Division	$4 / 2$	2
%	Modulus operator to get remainder in integer division	$5 \% 2$	1
**	Exponent	$5 ** 2 = 5^2$	25
//	Integer Division/ Floor Division	$5 // 2$ $-5 // 2$	2 -3

$$a = 3 + b + 5$$


$$a = 3 + b + 5$$

$$a = 10$$


```
result = 2 + 3 * 4  
print(result)
```

$$2 + (3 * 4) = 2 + 12 = 14$$

```
result = (2 + 3) * 4  
print(result)
```

$$10 + (6 / 2) = 10 + 3.0 = 13.0$$

```
result = (2 + 3) * 4  
print(result)
```

$$(2 + 3) * 4 = 5 * 4 = 20$$

```
result = 10 % 3 * 2  
print(result)
```

$$(10 \% 3) * 2 = 1 * 2 = 2$$


```
result = 2 * 3 ** 2  
print(result)
```

$$2 * (3 ** 2) = 2 * 9 = 18$$

```
result = 10 - 5 // 2  
print(result)
```

8

```
result = 5 + 2 * 3 ** 2 // 4 - 1  
print(result)
```

تمرین



```
result = -3 ** 2  
print(result)
```

$$-(3 \times 2) = -9$$

```
result = -10 // 3  
print(result)
```

-4


```
result = 10 % 4 ** 2  
print(result)
```

$$10 \% (4 ** 2) = 10 \% 16 = 10$$

Precedence	Operator	Description	Example	Result
1	<code>**</code>	Exponentiation (Power)	<code>2 ** 3</code>	<code>8</code>
2	<code>+x</code> , <code>-x</code>	Unary plus and minus	<code>-3</code> , <code>+5</code>	<code>-3</code> , <code>5</code>
3	<code>*</code> , <code>/</code> , <code>//</code> , <code>%</code>	Multiplication, Division, Floor Division, Modulus	<code>10 / 3</code> , <code>10 // 3</code> , <code>10 % 3</code>	<code>3.333</code> , <code>3</code> , <code>1</code>
4	<code>+</code> , <code>-</code>	Addition and Subtraction	<code>10 + 5</code> , <code>10 - 3</code>	<code>15</code> , <code>7</code>

```
result = 10 - 5 - 2  
print(result)
```

