

</

# پایتون مقدماتی

موسسه آموزشی آپادانا

/>



Beginners python

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1

# پایتون

Python یک زبان برنامه‌نویسی سطح بالا، تفسیری و عمومی است که برای اولین بار در سال 1991 توسط گuido فان روسوم ( Guido van Rossum) توسعه داده شد. پایتون به دلیل طراحی ساده و خوانا، محبوبیت زیادی در میان توسعه‌دهندگان نرم‌افزار پیدا کرده است. این زبان با هدف افزایش بهره‌وری توسعه‌دهنده و کاهش هزینه‌های نگهداری کد طراحی شده است.



## ویژگی‌های اصلی پایتون:





## کاربردهای پایتون:

علم داده و  
یادگیری ماشین



توسعه وب



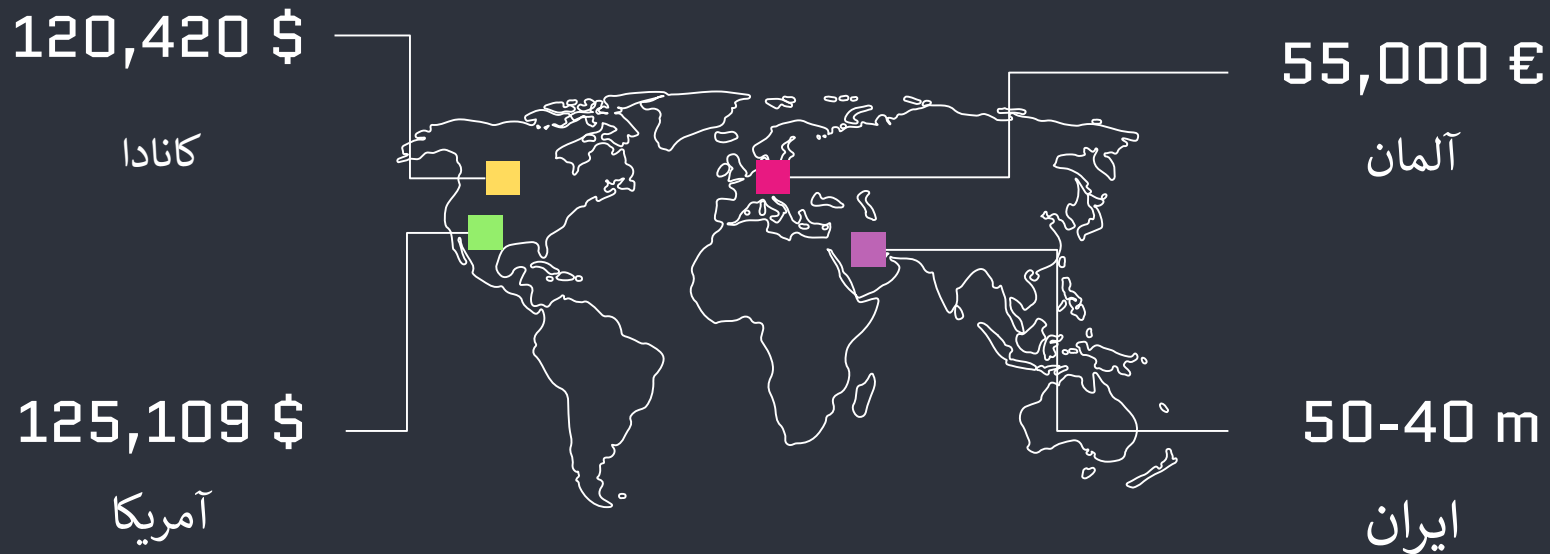
اتوماسیون و  
اسکرپت نویسی



شبکه و امنیت



# میانگین حقوق برنامه نویس پایتون



## نصب python

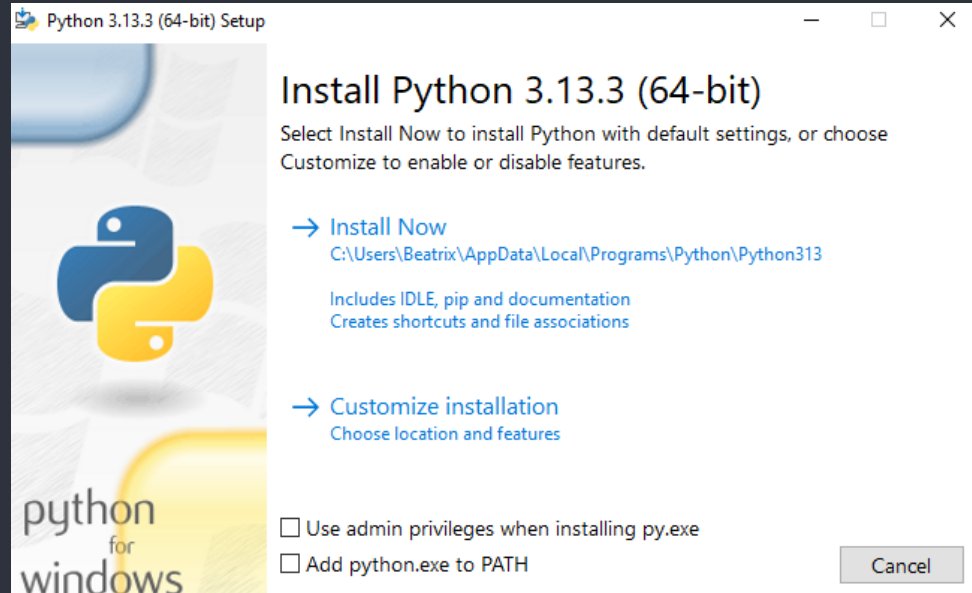
آخرین ورژن پایتون

## نصب vscode

نصب و تنظیم VS Code به عنوان IDE اصلی

## نوشتن اولین کد

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



# دستورات پایه ای CMD

- **نمایش مسیر فعلی**

```
bash
1 cd
```

این دستور مسیر فعلی شما رو نشون می‌ده. اگر بدون پارامتر استفاده بشه، فقط مسیر فعلی رو نمایش می‌ده

$$1 \ 0 \ 1 \ 1 \quad 0 \ 1 \ 1 \quad 0 \ 1 \quad 1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \quad 1 \ 0 \quad 1 \ 1 \ 0 \ 1 \ 1 \quad 0 \ 1 \ 1 \quad 0 \ 1 \quad 1 \ 1 \ 0 \ 1 \ 1 \ 0 \quad 1 \ 1 \ 0 \ 1 \ 1 \ 1 \quad 1 \ 1 \ 0 \ 1$$

# دستورات پایه ای CMD

- تغییر مسیر

این دستور به شما کمک می‌کند که به مسیر دیگری بروید. مثلاً اگر می‌خواهید به دسکتاپ بروید، مسیر رو وارد کنید

- دستور :

```
bash
```

```
1 mkdir [نام پوشه]
```

- مثال :

```
bash
```

```
1 mkdir MyFolder
```



# دستورات پایه ای CMD

## • حذف فایل

این دستور فایل مشخص شده رو حذف می کنه. توجه : این عمل غیرقابل بازگشت است

### • دستور :

```
bash
```

```
1 del [نام فایل]
```

### • مثال :

```
bash
```

```
1 del myfile.txt
```

# دستورات پایه ای CMD

## . حذف پوشه

این دستور پوشه مشخص شده رو حذف می کنه. اگر پوشه خالی نباشه، اول باید فایل هاش رو حذف کنید

- دستور :

```
bash
```

```
1 rmdir [نام پوشه]
```

- مثال :

```
bash
```

```
1 rmdir MyFolder
```

# دستورات مرتبط با پایتون

## بررسی نصب پایتون

```
bash
```

```
1 python --version
```

1 0 1 1   0 1 1   0 1   1 0 1 1 0 0 1   1 0   1 1 0 1 1   0 1 1   0 1   1 1 0 1 1 0   1 1 0 1 1 1   1 1 0 1

# دستورات مرتبط با پایتون

## • اجرای فایل پایتون

- دستور :

```
bash
```

```
1 python [نام فایل].py
```

- مثال :

```
bash
```

```
1 python script.py
```

# دستورات مرتبط با پایتون

## بررسی نصب پایتون

```
bash
```

```
1 python --version
```

1 0 1 1    0 1 1    0 1    1 0 1 1 0 0 1    1 0    1 1 0 1 1    0 1 1    0 1    1 1 0 1 1 0    1 1 0 1 1 1    1 1 0 1

# دستورات مرتبط با پایتون

• خروج از محیط تعاملی پایتون

• دستور :

python

1 exit()

• یا :

bash

1 Ctrl + Z

# Primitive (Built-in) Data Types

اعداد کامل بدون اعشار	1000 , 12- , 5	int	عدد صحیح
اعدادی که قسمت اعشاری دارند	2.5- , 3.14	float	عدد اعشاری
دنباله‌ای از کاراکترها (رشته)	'Python' , "سلام"	str	متن
داده‌های منطقی، بله یا خیر	True , False	bool	درست/غلط

1 0 1 1   0 1 1   0 1   1 0 1 1 0 0 1   1 0   1 1 0 1 1   0 1 1   0 1   1 1 0 1 1 0   1 1 0 1 1 1   1 1 0 1

# Data Types

عدد بدون اعشار	0 , 3- , 10	int	Integer	عدد صحیح
عدد با اعشار	0.5- , 3.14	float	Floating Point	عدد اعشاری
عدد مختلط (ریاضی)	3j + 2	complex	Complex Number	عدد مختلط
رشته متنی	'Hi' , "سلام"	str	String	متن
درست یا غلط	True , False	bool	Boolean	منطقی
مجموعه‌ی قابل تغییر با ترتیب	["a" , "b"] , [3 , 2 , 1]	list	List	لیست
مجموعه‌ی غیرقابل تغییر با ترتیب	("a" , "b") , (2 , 1)	tuple	Tuple	تاپل
مجموعه‌ی بدون ترتیب و بدون تکرار	{3 , 2 , 1}	set	Set	مجموعه
داده‌های کلید-مقدار	name: "Ali" , "age": " {20	dict	Dictionary	دیکشنری
نبود مقدار یا مقدار تهی	None	NoneType	NoneType	None



```
d = bool("")      # خط 1
e = bool("False") # خط 2
f = int(float(y))  # خط 3
```

```
print(d)  # خروجی؟
print(e)  # خروجی؟
print(f)  # خروجی؟
```

1 0 1 1    0 1 1    0 1    1 0 1 1 0 0 1    1 0    1 1 0 1 1    0 1 1    0 1    1 1 0 1 1 0    1 1 0 1 1 1    1 1 0 1

```
x = "12"
```

```
y = "3.5"
```

```
a = int(x) + 5
```

```
b = float(y) + 1
```

```
c = str(a) + y
```

```
print(a)    # خروجی؟
```

```
print(b)    # خروجی؟
```

```
print(c)    # خروجی؟
```

# Operators

Arithmetic Operators	عملگرهای حسابی
Comparison Operators	عملگرهای مقایسه‌ای
Logical Operators	عملگرهای منطقی
Assignment Operators	عملگرهای انتساب
Membership Operators	عملگرهای عضویت
Identity Operators	عملگرهای هویتی
Bitwise Operators	عملگرهای بیتی (پیشرفته‌تر)

1 0 1 1    0 1 1    0 1    1 0 1 1 0 0 1    1 0    1 1 0 1 1    0 1 1    0 1    1 1 0 1 1 0    1 1 0 1 1 1    1 1 0 1

# Arithmetic Operators

Operator	Name	Example
<code>+</code>	Addition	<code>a + b</code>
<code>-</code>	Subtraction	<code>a - b</code>
<code>*</code>	Multiplication	<code>a * b</code>
<code>/</code>	Division	<code>a / b</code>
<code>//</code>	Floor Division	<code>a // b</code>
<code>%</code>	Modulus (remainder)	<code>a % b</code>
<code>**</code>	Exponentiation	<code>a ** b</code>

1 0 1 1   0 1 1   0 1   1 0 1 1 0 0 1   1 0   1 1 0 1 1   0 1 1   0 1   1 1 0 1 1 0   1 1 0 1 1 1   1 1 0 1

# Comparison Operators

Operator	Name	Example
<code>==</code>	Equal to	<code>a == b</code>
<code>!=</code>	Not equal to	<code>a != b</code>
<code>&gt;</code>	Greater than	<code>a &gt; b</code>
<code>&lt;</code>	Less than	<code>a &lt; b</code>
<code>&gt;=</code>	Greater than or equal to	<code>a &gt;= b</code>
<code>&lt;=</code>	Less than or equal to	<code>a &lt;= b</code>

1 0 1 1   0 1 1   0 1   1 0 1 1 0 0 1   1 0   1 1 0 1 1   0 1 1   0 1   1 1 0 1 1 0   1 1 0 1 1 1   1 1 0 1

# Logical Operators

Operator	Name	Description	Example ( a = True , b = False )	Result
and	Logical AND	Returns True if <b>both</b> are true	a and b	False
or	Logical OR	Returns True if <b>at least one</b> is true	a or b	True
not	Logical NOT	Reverses the Boolean value	not a	False

1 0 1 1    0 1 1    0 1    1 0 1 1 0 0 1    1 0    1 1 0 1 1    0 1 1    0 1    1 1 0 1 1 0    1 1 0 1 1 1    1 1 0 1

# Assignment Operators

Operator	Description	Example
<code>=</code>	Assign value	<code>x = 5</code>
<code>+=</code>	Add and assign	<code>x += 3</code>
<code>-=</code>	Subtract and assign	<code>x -= 2</code>
<code>*=</code>	Multiply and assign	<code>x *= 4</code>
<code>/=</code>	Divide and assign	<code>x /= 2</code>
<code>//=</code>	Floor divide and assign	<code>x //= 3</code>
<code>%=</code>	Modulus and assign	<code>x %= 2</code>
<code>**=</code>	Exponentiate and assign	<code>x **= 2</code>

1 0 1 1   0 1 1   0 1   1 0 1 1 0 0 1   1 0   1 1 0 1 1   0 1 1   0 1   1 1 0 1 1 0   1 1 0 1 1 1   1 1 0 1

# Membership Operators

Operator	Description	Example
<code>in</code>	Checks if a value exists in the sequence	<code>a in lst</code>
<code>not in</code>	Checks if a value does not exist in the sequence	<code>a not in lst</code>

1 0 1 1   0 1 1   0 1   1 0 1 1 0 0 1   1 0   1 1 0 1 1   0 1 1   0 1   1 1 0 1 1 0   1 1 0 1 1 1   1 1 0 1



# Identity Operators

Operator	Description	Example
<code>is</code>	Checks if two variables refer to the same object in memory	<code>a is b</code>
<code>is not</code>	Checks if two variables do not refer to the same object in memory	<code>a is not b</code>

# String operations

Operation	Syntax	Description	Example
Indexing	<code>s[i]</code>	Character at index <code>i</code>	<code>'hello'[1] → 'e'</code>
Slicing	<code>s[start:end]</code>	Substring from <code>start</code> to <code>end-1</code>	<code>'hello'[1:4] → 'ell'</code>
Slicing with Step	<code>s[start:end:step]</code>	Substring every <code>step</code> chars	<code>'hello'[::-2] → 'hlo'</code>
Length	<code>len(s)</code>	Number of characters	<code>len('hello') → 5</code>
Concatenation	<code>s1 + s2</code>	Joins strings	<code>'hi' + '!' → 'hi!'</code>
Repetition	<code>s * n</code>	Repeats the string <code>n</code> times	<code>'ha' * 3 → 'hahaha'</code>
In Operator	<code>'sub' in s</code>	Checks for substring	<code>'he' in 'hello' → True</code>
Format (f-string)	<code>f"{var}"</code>	Inserts variable in string	<code>f"Hi {name}" → 'Hi Ali'</code>

1 0 1 1   0 1 1   0 1   1 0 1 1 0 0 1   1 0   1 1 0 1 1   0 1 1   0 1   1 1 0 1 1 0   1 1 0 1 1 1   1 1 0 1

# If and else statements

```
If a > b :  
    print("a is bigger than b")  
Elif b > a:  
    print("b is bigger")  
Else:  
    print("a is equal to b")
```

1 0 1 1   0 1 1   0 1   1 0 1 1 0 0 1   1 0   1 1 0 1 1   0 1 1   0 1   1 1 0 1 1 0   1 1 0 1 1 1   1 1 0 1

# Flow Chart

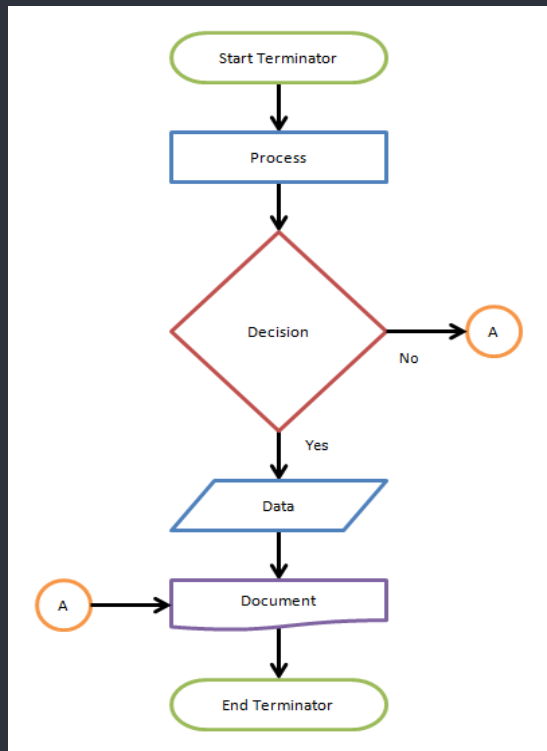
هر فلوچارت از چند نماد تشکیل شده که هر کدام  
معنی خاصی دارند:

**Start/End** بیضی : شروع و پایان برنامه.

**Process** مستطیل : پردازشها/Input/Output

متوازی الاضلاع : ورودی و خروجی.

**Decision** لوزی : تصمیم گیری (مثل شرطها).



# </Debugging

## 1. اشتباهات نحوی : (Syntax Errors)

• "وقتی قوانین زبان پایتون رو رعایت نکنید. مثلاً فراموش کردن : در انتهای دستور "if."

## 2. اشتباهات منطقی : (Logical Errors)

• "کد اجرا می‌شه ولی خروجی اشتباهه. مثلاً جمع دو عدد رو به جای ضرب حساب کنید."

## 3. اشتباهات زمان اجرا : (Runtime Errors)

• "کد اجرا می‌شه ولی توی اجرا خطایی پیش میاد. مثلاً تقسیم یه عدد به صفر."

# Git & Github

## Git چیست؟

یک نرم افزار برای کنترل نسخه (Version Control) است که به شما اجازه می دهد تغییرات فایل های پروژه را ذخیره، بررسی و بازگردانی کنید.

## GitHub چیست؟

یک سایت است که به کمک Git، پروژه ها را آنلاین و اشتراکی ذخیره و مدیریت می کند. مثل Google Drive برای پروژه های برنامه نویسی.

# Git & Github

ذخیره نسخه‌های مختلف پروژه ✓


کار تیمی روی یک پروژه ✓


ارسال تکالیف و مشاهده تغییرات ✓


گرفتن بک‌آپ از کدها در فضای ابری ✓


همکاری با دیگران از راه دور ✓

# Git & Github

1. دریافت پروژه (Clone) 

2. ویرایش فایل‌ها 

3. ثبت تغییرات (Add + Commit) 

4. ارسال به گیت‌هاب (Push) 

5. دریافت به‌روزرسانی (Pull) ← در کار تیمی 

 نکته مهم

Git = ابزار

GitHub = فضای ذخیره‌سازی و همکاری آنلاین



# Git and Github

```
git clone https://github.com/yourusername/assignment-1.git  
cd assignment-1
```

```
git add .
```

```
git commit -m "Completed assignment 1"
```

```
git push origin main
```