

اصول مقدماتی پایتون ۱

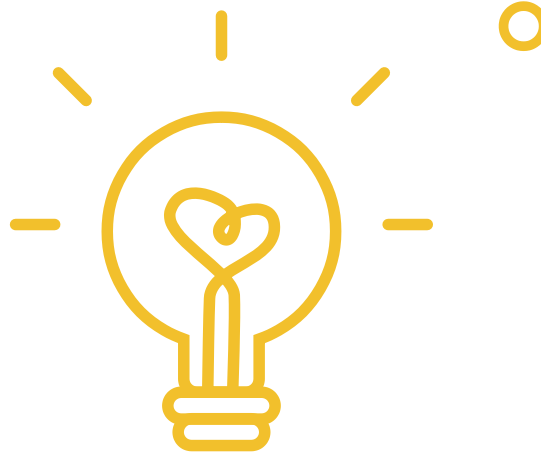
متغیرها

هادی فرهادی
شهریور ۱۴۰۴

h.farhadi.py@gmail.com



متغير (Variable)





ظرفی است برای نگه داری چیزها





مقدار



متغیر





علم کامپیوتر: مکانی در فضای ذخیره سازی (هارد، رم) است برای
ذخیره داده‌ها، که با یک **نام** قابل دسترسی است







تعریف یک متغیر

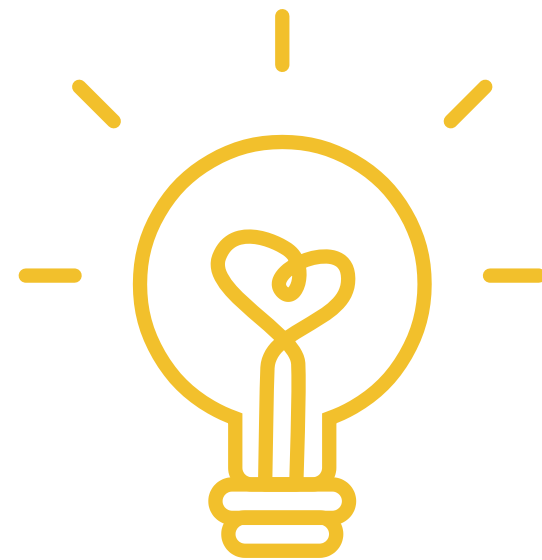




ساخت یک متغیر در پایتون



apples = 5



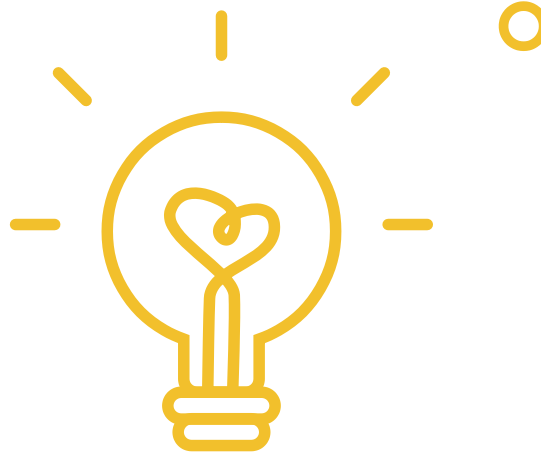


استفاده از هوش مصنوعی تا اطلاع ثانوی ممنوع





تابع print - نمایش خروجی





تابع print - نمایش خروجی



تابع print یکی از پرکاربردترین توابع در پایتون است که برای نمایش خروجی در کنسول استفاده می‌شود.



```
print("Hello World")
```



```
print(42)
```

```
print(3.14)
```

```
name = "Ali"
```



```
age = 25
```

```
print("Name:", name, "Age:", age)
```





تمرین تعریف متغیر و نمایش در خروجی





انواع متغيرها





int (اعداد صحیح)



مقادیر عددی بدون اعشار را ذخیره می کند

```
apple_basket = 5
```



```
user_level = -1
```





float (اعداد اعشاری)



اعداد اعشاری را ذخیره می کند

points = 18.36



PI = 3.14159





(مقادیر منطقی) bool



مقادیر درست یا اشتباه را ذخیره می‌کند تنها دو
مقدار را می‌پذیرد True, False



is_teacher = True



is_admin = False





str (رشته‌ای)



این نوع متغیرها مقادیر متنی را می‌پذیرند
به هر یک از اجزایش کاراکتر گفته می‌شود



```
full_name = 'Hadi Farhadi'
```



```
user_info = "I'm fine thank you so much"
```

اگر متن حاوی چندین خط باشد از "" یا "" استفاده می‌کنیم



```
user_bio = ""  
Hello my dear followers.  
Thanks for your support
```





برشته‌ای str



کاراکترهای خاص (escape characters) در پایتون برای نمایش کاراکترهایی استفاده می‌شوند که نمی‌توان مستقیماً آنها را در رشته‌ها نمایش داد یا کاربرد خاصی دارند.



(New Line) ایجاد خط جدید \n



```
message = "Hello\nWorld"
```

```
print(message)
```

فاصله افقی - (Tab) \t



```
name_message = "Name:\tAli"
```

```
print(name_message)
```





NoneType (نوع تھی)



این نوع، یعنی تھی

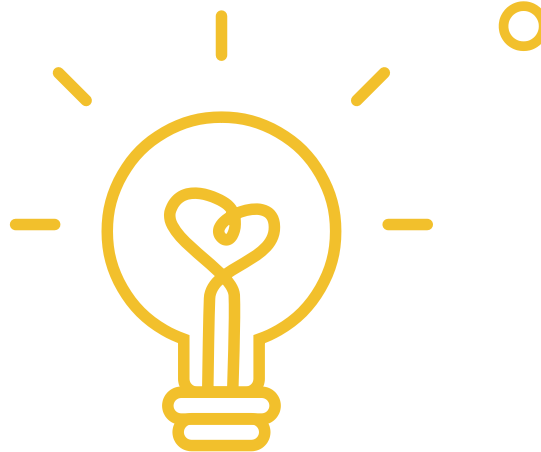


user = None





تمرین تعریف انواع متغیر ها





پایتون یک زبان داینامیک (Dynamic Typing) است یعنی نوع
متغیر در زمان اجرا مشخص می شود





اصول نامگذاری یک متغیر





اصول نامگذاری یک متغیر



۱. نباید از space بین کاراکترها استفاده کنید

apple basket = 5 ❌

apple_basket = 5 ✅



۲. متغیرها با عدد شروع نمی شوند



6apple_company = 6 ❌

apple_6_basket = 6 ✅





اصول نامگذاری یک متغیر



۳. نباید از کاراکترهای خاص استفاده کنید مانند

` ~ / ? . , < > ' " ; : \ | [] { } = + - () * & ^ % \$ # @ !

apple+basket = 5 ✗

apple_basket = 5 ✓

۴. نباید از کلمات کلیدی پایتون استفاده کنید

,False, None, True, and, as, assert, async, await, break

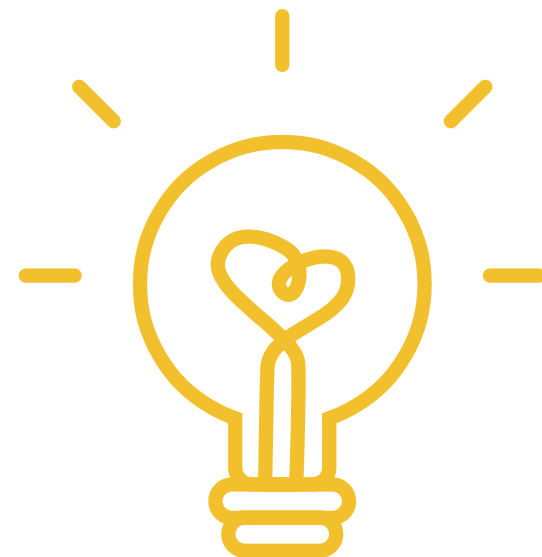
,class, continue, def, del, elif, else, except, finally

,for, from, global, if, import, in, is, lambda, nonlocal

not, or, pass, raise, return, try, while, with, yield

class = 6 ✗

apple_class = 6 ✓





اصول نامگذاری یک متغیر



5. در نامگذاری های چند کلمه ای بهتر است از کاراکتر - بین کلمات استفاده کرد که به آن snake case می گویند



appleBasket = 5 ✗

apple_basket = 5 ✓





اصول نامگذاری یک متغیر



برای انتخاب یک نام مناسب می‌توان از google translate استفاده کرد





Comment - توضیحات





Comment - توضیحات



کامنت‌ها توضیحاتی هستند که در کد قرار می‌گیرند اما توسط مفسر پایتون اجرا نمی‌شوند. هدف اصلی آن‌ها توضیح منطق کد و افزایش خوانایی است.
کامنت تک خطی:

```
print("Name:", name, "Age:", age) # Name: Ali Age: 25
```

کامنت چند خطی:

```
"""
```

این یک کامنت چند خطی است
که می‌تواند چندین خط را پوشش دهد
و معمولاً برای توضیح توابع و کلاس‌ها استفاده می‌شود

```
"""
```

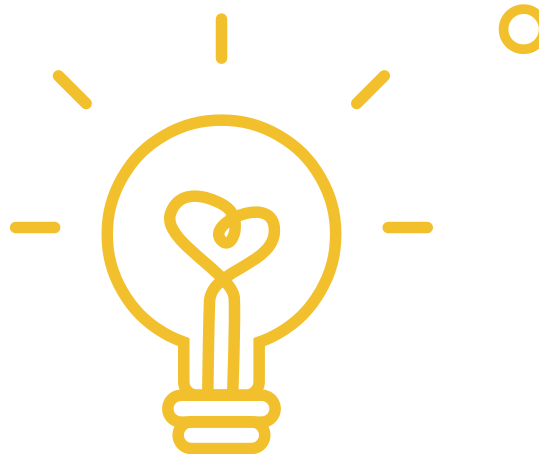




پایتون یک زبان مفسری است، به این معنی: دستورات را خط به خط و به ترتیب از بالا به پایین اجرا می کند

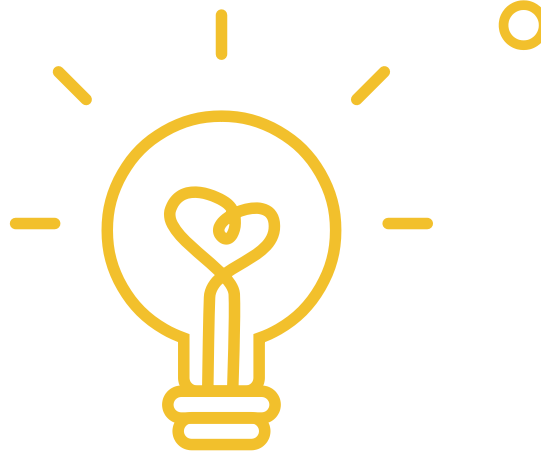


در پایتون، یک دستور Statement، واحد عملیاتی است که مفسر پایتون می‌تواند آن را اجرا کند. هر دستور می‌تواند در یک یا چند خط کد باشد.





Type Annotation





Type Annotation



حاشیه‌نویسی نوع یکی از ویژگی‌های مهم پایتون است که
از نسخه 3.5 به بعد اضافه شده است. این ویژگی به
توسعه‌دهندگان اجازه می‌دهد نوع داده‌های مورد انتظار را
مشخص کنند.

در زمان اجرا اعمال نمی‌شوند و فقط جنبه اطلاع‌رسانی
دارند

```
name: str = "Elnaz"  
age: int = 24  
is_student: bool = True
```

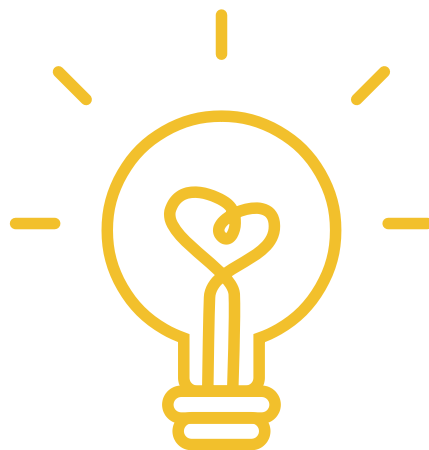


```
# با مقداردهی بعدی  
count: int  
count = 10
```



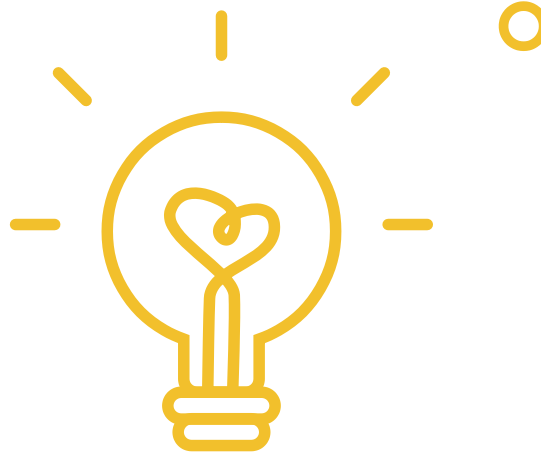


تمرین LinkedIn



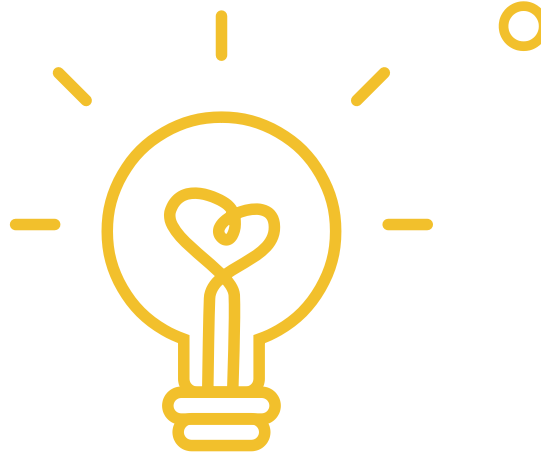


تمرین Youtube





تمرین Music





اپراتور - Operators





عملگرهای حسابی (Arithmetic Operators)





عملگرهای حسابی (Arithmetic Operators)



اپراتور جمع +: برای جمع کردن دو مقدار به کار می رود



```
first_number: int = 5  
second_number: int = 6  
result: int = first_number + second_number  
  
print(first_number, " + ", second_number, " = ", result)
```



```
first_name: str = "Hadi"  
last_name: str = "Farhadi"  
result: str = first_name + last_name  
  
print("Full Name is: ", result)
```





عملگرهای حسابی (Arithmetic Operators)



اپراتور تفریق - : برای تفریق دو مقدار به کار می رود



```
first_number: int = 8
```

```
second_number: int = 2
```

```
Result: int = first_number - second_number
```

```
print(first_number, " - ", second_number, " = ", result)
```





عملگرهای حسابی (Arithmetic Operators)



اپراتور ضرب * : برای ضرب دو مقدار به کار می رود



```
first_number: int = 8  
second_number: int = 9  
Result: int = first_number * second_number
```



```
print(first_number, " * ", second_number, " = ", result)
```

```
Number: int = 8  
Name: str = "Hadi"  
result: str = number * name
```



```
print("result = ", result)
```





عملگرهای حسابی (Arithmetic Operators)



اپراتور تقسیم / : برای تقسیم دو مقدار به کار می رود



```
first_number: int = 12
second_number: int = 9
result: float = first_number / second_number

print(first_number, " / ", second_number, " = ", result)
```





عملگرهای حسابی (Arithmetic Operators)



اپراتور تقسیم صحیح // : برای تقسیم صحیح دو مقدار به کار می رود



```
first_number: int = 12
second_number: int = 9
result: float = first_number // second_number

print(first_number, " // ", second_number, " = ", result)
```





عملگرهای حسابی (Arithmetic Operators)



اپراتور باقیمانده % : برای بدست آوردن باقیمانده تقسیم
دو مقدار به کار می رود



```
first_number: int = 12  
second_number: int = 9  
result: float = first_number % second_number  
  
print(first_number, " % ", second_number, " = ", result)
```





عملگرهای حسابی (Arithmetic Operators)



اپراتور توان **: برای بدست آوردن توان به کار می رود



```
first_number: int = 5
second_number: int = 2
result: int = first_number ** second_number

print(first_number, " ** ", second_number, " = ", result)
```





تمرین اپراتورهای حسابی





عملگرهای مقایسه‌ای (Comparison Operators)





عملگرهای مقایسه‌ای (Comparison Operators)



اپراتور == : برای بررسی تساوی دو مقدار به کار می‌رود.
خروجی عملگرهای مقایسه‌ای همیشه از نوع bool است



```
first_number: int = 5  
second_number: int = 2  
result: bool = first_number == second_number  
  
print(first_number, " == ", second_number, " is ", result)
```





عملگرهای مقایسه‌ای (Comparison Operators)



اپراتور != : برای بررسی عدم تساوی دو مقدار به کار می‌رود.



```
first_number: int = 5
second_number: int = 2
result: bool = first_number != second_number

print(first_number, " != ", second_number, " is ", result)
```





عملگرهای مقایسه‌ای (Comparison Operators)

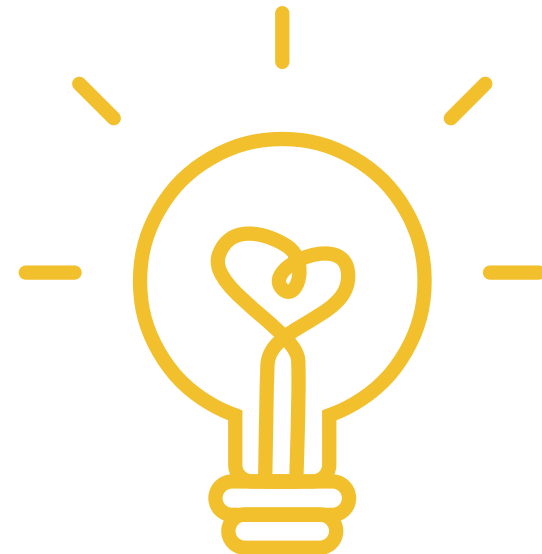


اپراتور > : برای بررسی بزرگتر بودن یک مقدار از دیگری به کار می‌رود.



```
first_number: int = 5  
second_number: int = 2  
result: bool = first_number > second_number
```

```
print(first_number, ">", second_number, " is ", result)
```





عملگرهای مقایسه‌ای (Comparison Operators)



اپراتور $<$: برای بررسی کوچکتر بودن یک مقدار از دیگری به کار می‌رود.



```
first_number: int = 5  
second_number: int = 2  
result: bool = first_number < second_number
```

```
print(first_number, " < ", second_number, " is ", result)
```





عملگرهای مقایسه‌ای (Comparison Operators)



اپراتور \geq : برای بررسی بزرگتر یا مساوی بودن یک مقدار از دیگری به کار می‌رود.



```
first_number: int = 5
second_number: int = 2
result: bool = first_number >= second_number

print(first_number, " >= ", second_number, " is ", result)
```





عملگرهای مقایسه‌ای (Comparison Operators)



اپراتور \geq : برای بررسی کوچکتر یا مساوی بودن یک مقدار از دیگری به کار می‌رود.



```
first_number: int = 5
second_number: int = 2
result: bool = first_number <= second_number

print(first_number, " <= ", second_number, " is ", result)
```



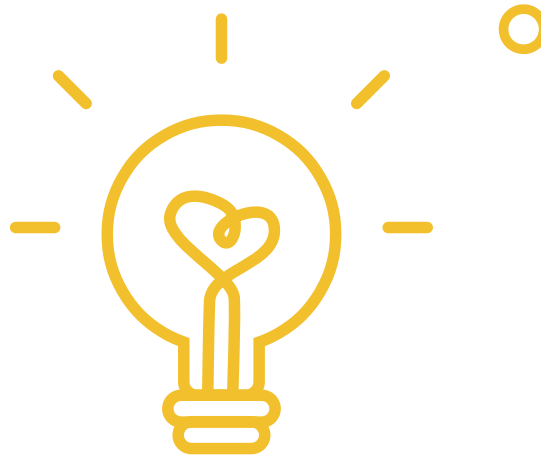


تمرین اپراتورهای مقایسه ای





عملگرهای انتساب (Assignment Operators)





عملگرهای انتساب (Assignment Operators)



اپراتور = : برای انتساب یک مقدار به یک متغیر استفاده می شود.



```
first_number: int = 5
```

اپراتور += : برای جمع مقدار یک متغیر با مقدار متغیر دیگر و ذخیره کردن مقدار نهایی در متغیر اول.



```
first_number: int = 5
```

```
second_number: int = 2
```

```
first_number += second_number
```



```
print("First Number = ", first_number)
```





عملگرهای انتساب (Assignment Operators)



اپراتور -= : برای تفریق مقدار یک متغیر از مقدار متغیر دیگر و ذخیره کردن مقدار نهایی در متغیر اول.



```
first_number: int = 5  
second_number: int = 2  
first_number -= second_number
```



```
print("First Number = ", first_number)
```

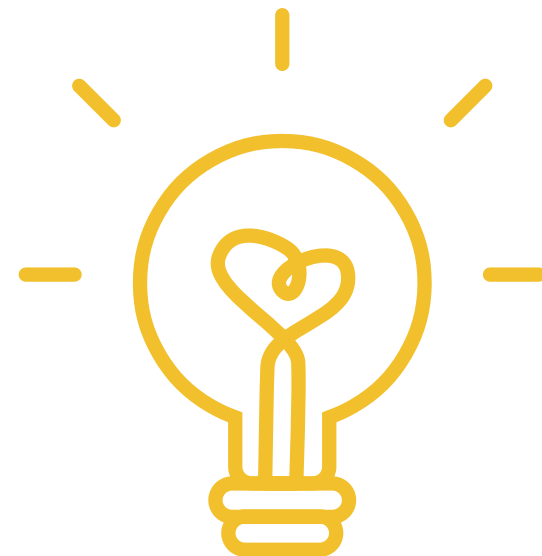
اپراتور *= : برای ضرب مقدار یک متغیر در مقدار متغیر دیگر و ذخیره کردن مقدار نهایی در متغیر اول.



```
first_number: int = 5  
second_number: int = 2  
first_number *= second_number
```



```
print("First Number = ", first_number)
```





عملگرهای انتساب (Assignment Operators)



اپراتور `/=` : برای تقسیم مقدار یک متغیر بر مقدار متغیر دیگر و ذخیره کردن مقدار نهایی در متغیر اول.



```
first_number: int = 5  
second_number: int = 2  
first_number /= second_number
```



اپراتور `//=` : برای تقسیم صحیح مقدار یک متغیر بر مقدار متغیر دیگر و ذخیره کردن مقدار نهایی در متغیر اول.



```
first_number: int = 5  
second_number: int = 2  
first_number //= second_number
```



```
print("First Number = ", first_number)
```





عملگرهای انتساب (Assignment Operators)



اپراتور $\% =$: برای بدست باقیمانده یک تقسیم مقدار یک متغیر بر مقدار متغیر دیگر و ذخیره کردن مقدار نهایی در متغیر اول.

```
first_number: int = 5
```

```
second_number: int = 2
```

```
first_number %= second_number
```

```
print("First Number = ", first_number)
```

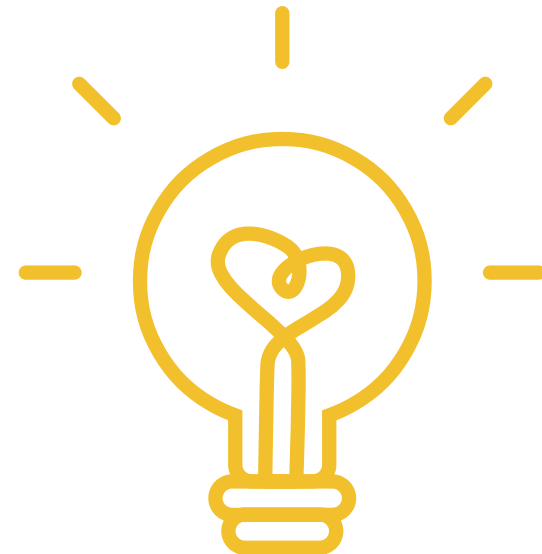
اپراتور $** =$: برای به توان رساندن مقدار یک متغیر توسط مقدار متغیر دیگر و ذخیره کردن مقدار نهایی در متغیر اول.

```
first_number: int = 5
```

```
second_number: int = 2
```

```
first_number **= second_number
```

```
print("First Number = ", first_number)
```





تمرین اپراتورهای انتساب





عملگرهای منطقی (Logical Operators)





عملگرهای منطقی (Logical Operators)



اپراتور and : بررسی درست بودن هر دو مقدار دو متغیر



```
first_number: int = 5  
second_number: int = 2  
result: any = first_number and second_number
```



```
print(first_number, " and ", second_number, " is ", result)
```

اپراتور or : مقدار یکی از دو متغیر درست باشد یا این یا آن



```
first_number: int = 5  
second_number: int = 2  
result: int = first_number or second_number
```



```
print(first_number, " or ", second_number, " is ", result)
```





عملگرهای منطقی (Logical Operators)



اپراتور not: خلاف مقدار این متغیر



```
first_number: int = 5
```

```
result: bool = not first_number
```



```
print("not", first_number, " is ", result)
```





تمرین اپراتورهای منطقی



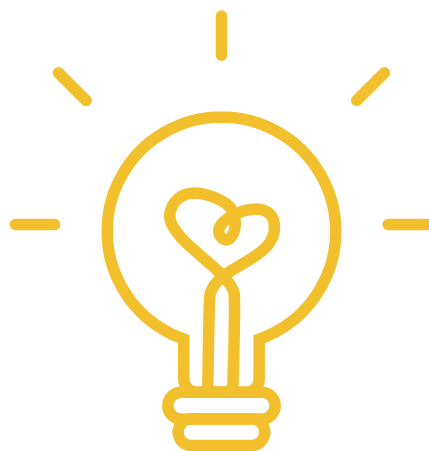


Indentation and Readabilty





اولویت اپراتورها (Operator Precedence)





اولویت اپراتورها (Operator Precedence)



اولویت عملگرها تعیین می‌کند که کدام عملگرها زودتر در یک عبارت ارزیابی شوند.



ارزیابی از چپ به راست برای عملگرهای با اولویت یکسان

- 1) ()
- 2) **
- 3) +x, -x
- 4) *, /, //, %
- 5) +, -
- 6) ==, !=, >, >=, <, <=
- 7) not
- 8) and
- 9) or



نکته: شرکت‌ها پرداخت و مالیات





اولویت اپراتورها (Operator Precedence)



مثال:



```
result_one: int = 10 + 5 * 2 # 10 + 10 = 20
```

```
result_two: int = (10 + 5) * 2 # 15 * 2 = 30
```

```
print(result_one, result_two) # 20 30
```



```
result: int = 10 + 5 * 2 ** 3 / 4 - 1
```

```
print(result)
```





اولویت اپراتورها (Operator Precedence)



استفاده از پرانتز: برای خوانایی و اطمینان از اولویت



صحیح

`result = (a + b) * (c - d)`



`result = a + b * c - d`

`x = 5 # 1`

`y = 10`

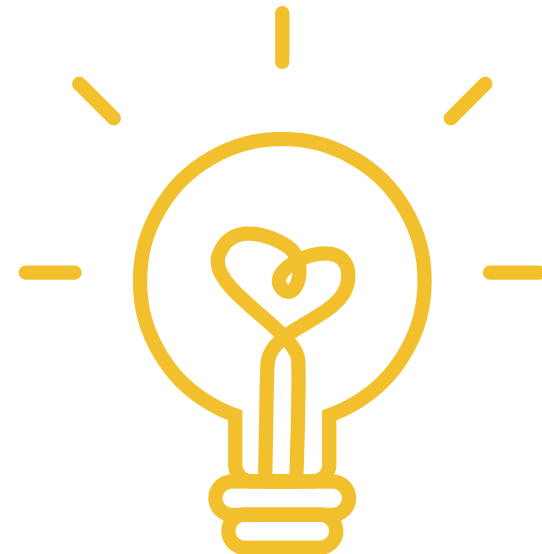
`z = 15`



`result = x < y and y < z or z == 15 # 2`



مشکل کدهای بالا چیست؟





Interactive mode





f-strings





f-strings



فرمت کردن رشته ها



```
message_first: str = "Hello"  
message_second: str = "Sara"
```

```
print(message_first + " " + message_second)
```



یکی از بهترین روش‌های فرمت سازی رشته‌ها در پایتون ۳.۶ به بعد است. این روش خوانایی و کارایی بالایی دارد.

f-strings

```
message_first: str = "Hello"  
message_second: str = "Ali"
```

```
print(f"{message_first} {message_second}")
```






تابع input - گرفتن ورودی از کاربر





تابع input - گرفتن ورودی از کاربر



تابع input() برای دریافت ورودی از کاربر از طریق خط فرمان (console) استفاده می‌شود. این تابع همیشه یک رشته (string) برمی‌گرداند.  سینتکس پایه:

```
variable = input(prompt)
```

مثال:

```
name: str = input("لطفاً نام خود را وارد کنید: ")  
print(f"سلام {name}!")
```





Type Casting (تبدیل نوع)





Type Casting (تبدیل نوع)



فرآیند تبدیل یک نوع داده به نوع دیگر است. پایتون
توابع داخلی برای انجام این تبدیل‌ها ارائه می‌دهد.



```
float_number: float = 5.23
```

```
str_number: str = "645"
```

```
result: float = float_number + str_number
```



تابع‌های تبدیل نوع

Integer (int()) تبدیل به

Float (float()) تبدیل به

String (str()) تبدیل به

Boolean (bool()) تبدیل به





مینی پروژه ماشین حساب





THANK YOU

h.farhadi.py@gmail.com