اصول مقدماتی پایتون ۲

control structures

هادی فرهادی
شهریور ۱۴۰۴

h.farhadi.py@gmail.com

یادآوری – نوع bool

یادآوری – اپراتورهای مقایسه ای

شرط‌ها در پایتون: if، else و elif

شرط‌ها در پایتون: if، else و elif

در پایتون، از ساختارهای شرطی برای کنترل جریان
اجرای برنامه بر اساس شرایط مختلف استفاده می‌شود.
مثلاً در یوتوب اگه کاربر بیشتر از ۱۸ سالش بود این محتوا را می تونه
ببینه در غیر اینصورت نمی تونه اون ببینه

```
if condition1:
    # اگر کدی‌که condition1 True باشد اجرا می‌شود
elif condition2:
    # اگر کدی‌که condition2 True باشد اجرا می‌شود
elif condition3:
    # اگر کدی‌که condition3 True باشد اجرا می‌شود
else:
    #اگر هیچ یکی از شرایط بالا True کدی‌که نباشند اجرا می‌شود
```

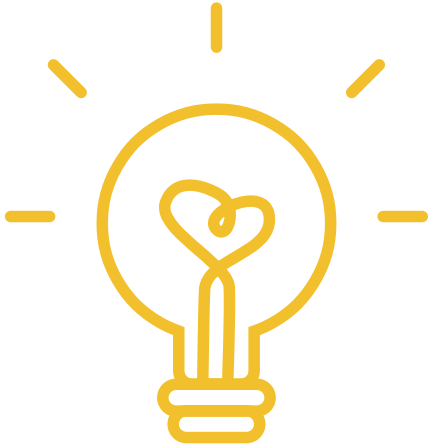
شرط‌ها در پایتون: if، else و elif

تو رفتگی بعد از : الزامی است
برای خوانایی بیشتر است اما اگه رعایت نشه به خطا می‌خورید

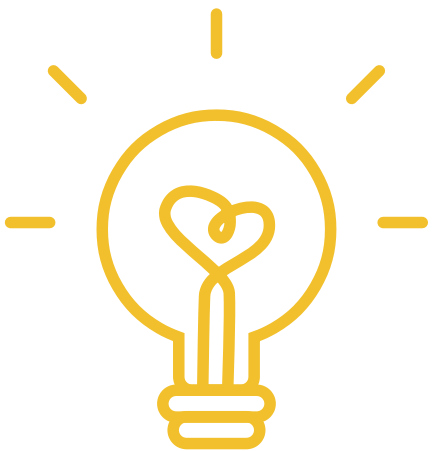
همیشه یکی از شرط ها درست هستند و اجرا می شوند

از تو رفتگی تا انتهای آن تو رفتگی = یک بلاک کد

تمرین : if، else و elif



شرط‌های تو در تو (Nested If) در پایتون



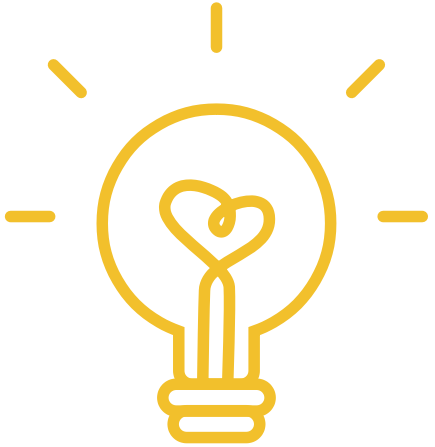
شرط‌های تو در تو (Nested If) در پایتون

شرط‌های تو در تو به معنای استفاده از دستورات شرطی در داخل یکدیگر است.این روش زمانی استفاده می‌شود که نیاز به بررسی شرایط چندلایه داریم.

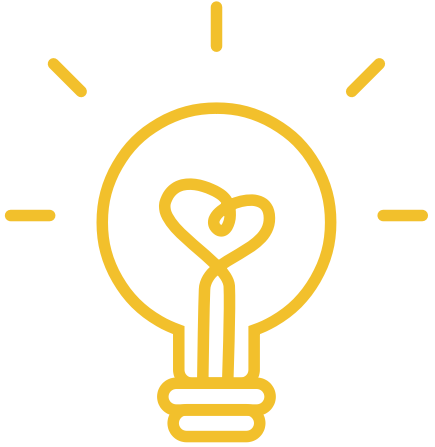
```
if condition1:
    # اگر condition1 True شود لجرا میشود
        if condition2:
                ...
        elif condition3:
                ...
elif condition4:
        ...
```

سعی کنید از Nested If فقط در صورت نیاز استفاده کنید چون خوانایی کد را پایین می آورد

تمرین : شرط‌های تو در تو (Nested If)



استفاده از چند condition به جای Nested if



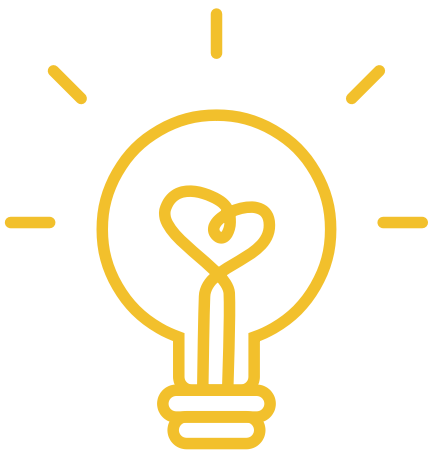
استفاده از چند condition به جای Nested if

جایگزین‌ها: گاهی می‌توان با استفاده از عملگرهای منطقی (and, or) از شرط‌های تو در تو اجتناب کرد:

```
if condition1:
    if condition2:
        if condition3:
            ...

if condition1 and condition2 and condition3:
    ...
```

تمرین: استفاده از چند condition به جای Nested if

... ,pass



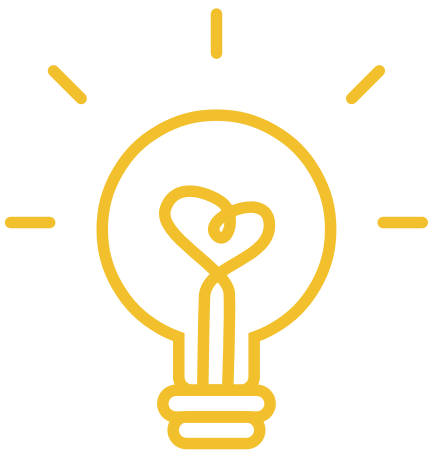
pass, ...

دستور pass در پایتون یک دستور تهی (null statement) است که هیچ کاری انجام نمی‌دهد. از آن به عنوان یک placeholder استفاده می‌شود تا از بروز خطاهای نحوی جلوگیری شود.

یا Ellipsis نیز به عنوان یک placeholder استفاده می‌شود، اما کاربردهای پیشرفته‌تری هم دارد.

```
age: int = 17
is_student: bool = True

if age >= 18:
    # TODO:اضافه کردن منطق رای‌دادن
    pass
elif age < 18 and is_student:
    ...
else:
    print("شما نمی‌توانید رای‌دهید")
```

در پایتون (List) لیست



لیست (List) در پایتون

لیست‌ها (type) یکی از پرکاربردترین ساختارها
که برای ذخیره مجموعه‌ای از مقادیر استفاده می‌شوند.

تغییرپذیر (Mutable): می‌توان محتوای لیست را تغییر داد

```
empty_list: list = []
empty_list: list = list()

numbers: list = [1, 2, 3, 4, 5]
fruits: list = ["apple", "banana", "orange"]
mixed: list = [1, "hello", 3.14, True]
```

تمرین ساخت لیست

دسترسی به عناصر لیست

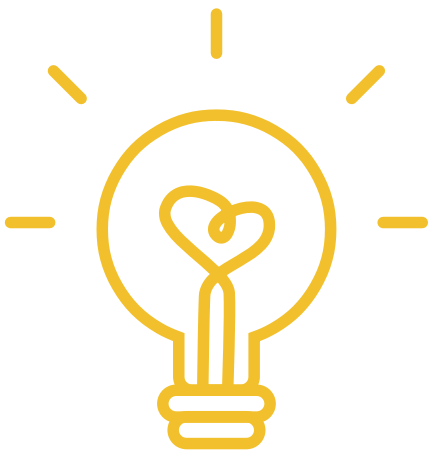
دسترسی به عناصر لیست

دسترسی به آیتم های از طریق اندیس صورت می‌پذیرد
و از صفر شروع می شود

```
fruits: list = ["apple", "banana", "orange", "grape", "mango"]

دسترسی با اندیس(از چپ به راست) #
print(fruits[0])    # apple
print(fruits[2])    # orange
```

تمرین دسترسی به عناصر لیست

تغییر عناصر لیست



تغییر عناصر لیست

```
fruits: list = ["apple", "banana", "orange"]

fruits[2] = "kiwi"
print(fruits)  # ['apple', 'banana', 'kiwi']
```

تمرین تغییر عناصر لیست

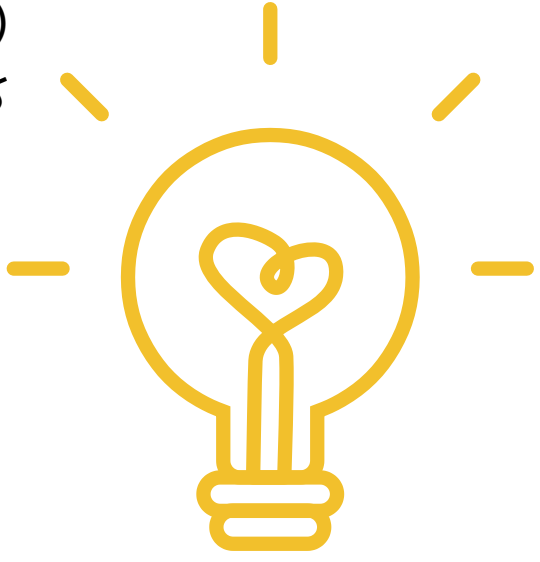
اضافه و حذف کردن عناصر لیست

اضافه و حذف کردن عناصر لیست

یکسری عملیات هست که از طریق کدهایی (متدها یا فانکشن هایی)
که برای آن نوع طراحی شده انجام می‌پذیرد.
به عنوان مثال:
تابع یا فانکشن print() که توسط خود پایتون طراحی شده است
و به ما در نمایش خروجی کمک می‌کند.
انواع یا Type ها (ذکر شده در درس قبل) هم شامل یکسری تابع یا فانکشن هستند
(که به آن‌ها متد گفته می‌شود) که توسط خود پایتون طراحی شده است
که از طریق نام متغییر و کاراکتر . قابل دستیابی هستند

```
fruits: list = ["apple", "banana"]
fruits.append("orange")
fruits.insert(1, "kiwi")
fruits.extend(["grape", "mango"])
fruits.remove("kiwi")
fruits.index(“banana”)
```

```
del fruits[1]  #حذف با اندیس
removed_item: str = fruits.pop()
fruits.clear()
fruits.count(“apple”)
```



تمرین اضافه و حذف کردن عناصر لیست

built in functions (list)

built in functions (list)

```
numbers: list = [3, 1, 4, 1, 5, 9, 2, 6]

print(len(numbers))

print(min(numbers))

print(max(numbers))

print(sum(numbers))

print(sorted(numbers))
```

لیست اصلی تغییر نمی‌کند: توابع max ،min()، sorted() و sum()
لیست اصلی را تغییر نمی‌دهند
کارایی: این توابع بهینه‌سازی شده‌اند و برای لیست‌های بزرگ مناسب هستند

تمرین built in functions (list)

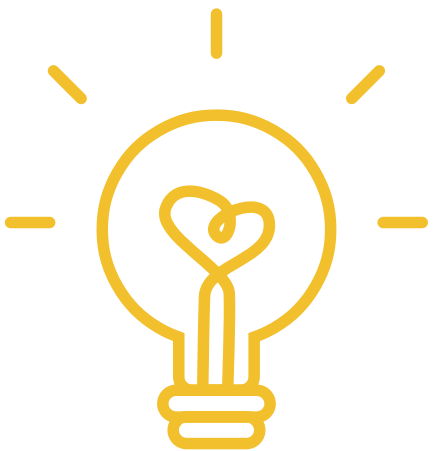
Membership operators (in)

Membership operators (in)

عملگرهای in و not in در پایتون برای بررسی وجود یا عدم وجود یک مقدار در یک دنباله (مانند لیست، رشته و ...) استفاده می‌شوند و مقدار True یا False برمی‌گرداند

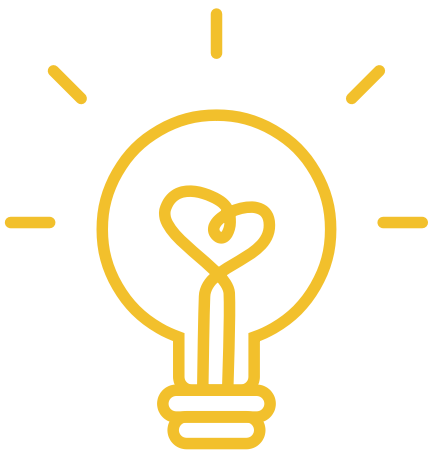
```
fruits = ["apple", "banana", "orange", "grape"]


print("apple" in fruits)   # True
print("kiwi" in fruits)    # False

if "banana" in fruits:
    print("موز در لیستمیوه‌ها وجود دارد")


if "mango" not in fruits:
    print("انبه در لیستمیوه‌ها وجود ندارد")
```

Membership operators (in) تمرین



Loop structures

while



while

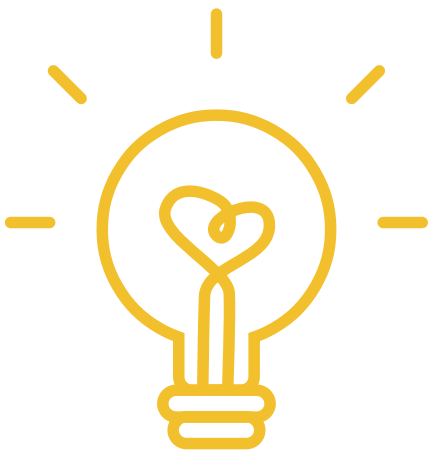
حلقه while در پایتون برای تکرار اجرای یک بلوک کد
تا زمانی که یک شرط خاص برقرار باشد استفاده می‌شود.

```
while condition:
    # شرط که تا زمانی که True می‌شود اجرا باشد
    # ...
```

```
count: int = 5
while count > 0:
    print(count)
    count -= 1

print(“Throw it”)
```

تمرین while

break

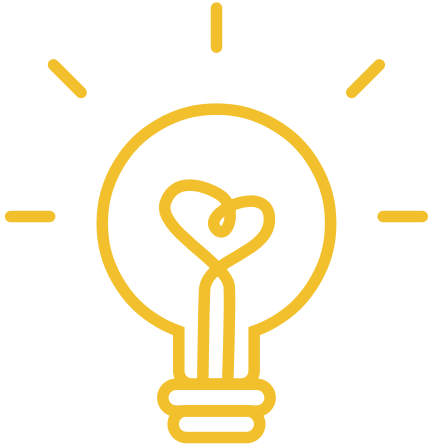


break

دستور break برای خروج از loop است

```
count: int = 5

while count > 0:
    print(count)
    count -= 1

    If count == 3:
        break

print(“Throw it”)
```

تمرین break



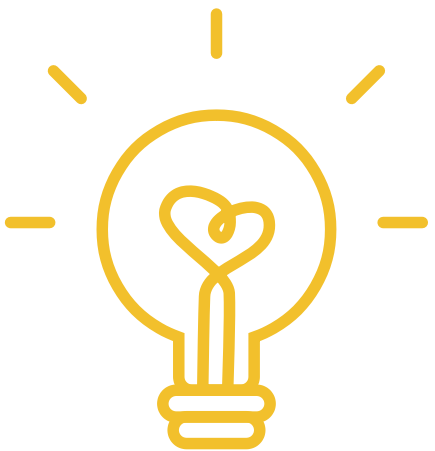
continue

continue

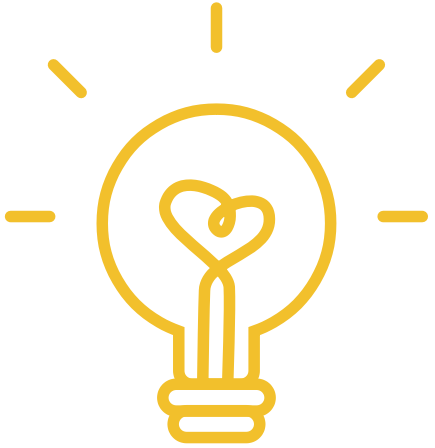
برای رد شدن از تکرار فعلی است

```
count: int = 5

while count > 0:

    If count == 3:
        continue

    print(count)
    count -= 1

print(“Throw it”)
```

تمرین continue



حلقه while با else



حلقه while با else

در پایتون، حلقه while می‌تواند یک بلوک else داشته باشد که بعد از اتمام طبیعی حلقه (بدون break) اجرا می‌شود.

```
count: int = 0

while count < 5:
    print(count)
    count += 1
else:
    print("loop ended")

print("program ended")
```

چرا این ویژگی منطقی به نظر نمی رسد؟

تمرین حلقه while با else

for loop

for loop

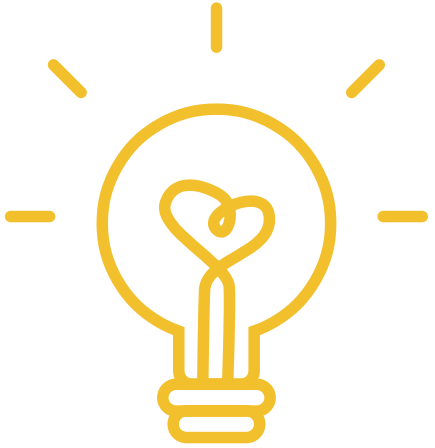
حلقه for در پایتون برای تکرار روی عناصر یک دنباله
(لیست، رشته، ....) یا دیگر شیءهای قابل تکرار استفاده می‌شود.

```
for variable in iterable:
    # code
```

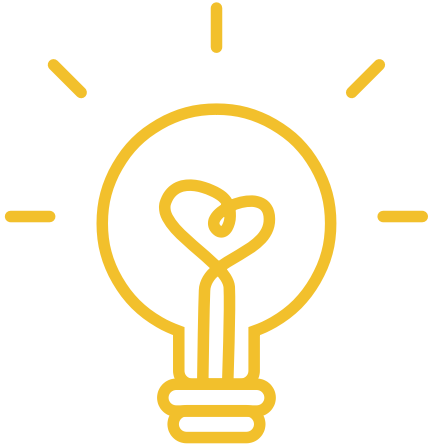
```
fruits: list = ["apple", "banana", "orange", "grape"]
```

```
for fruit in fruits:
    print(fruit)
```

تمرین for loop



range function



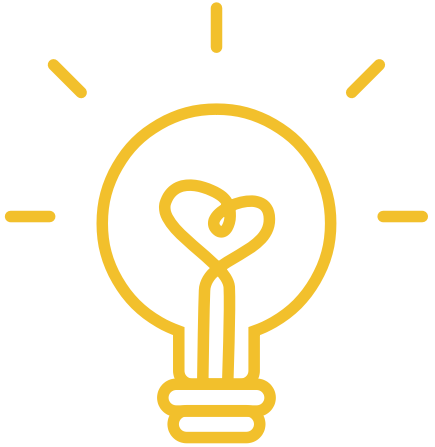
range function

برای تولید دنباله‌ای از اعداد استفاده می‌شود.

```
for i in range(5):
    print(i)
```

```
numbers: any = range(5)

print(f”type of numbers is {type(numbers)}”)
```

تمرین range



کجا از کدام استفاده شود

کجا از کدام استفاده شود

برای تکرار تا زمانی که شرط برقرار است while

برای تکرار روی دنباله‌ها for

تعداد تکرارها ممکن است نامشخص باشد while

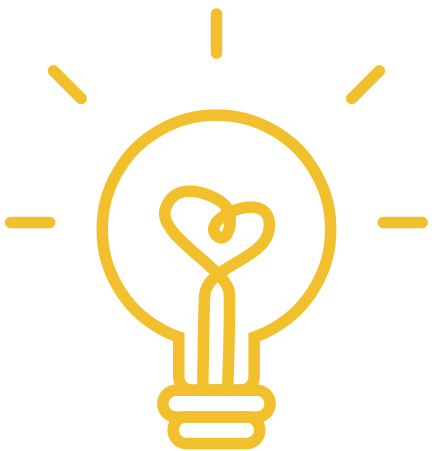
تعداد تکرارها معمولاً مشخص است for

نیاز به تعریف و به‌روزرسانی متغیر شمارنده while

از پیش تعریف شده بر اساس دنباله while

بیشتر بدانید: در سیستم عامل از while برای event loop استفاده می شود

Nested for



Nested for

حلقه‌های for تو در تو به معنی قرار دادن یک حلقه for داخل حلقه for دیگر است. این تکنیک برای پردازش ساختارهای داده دو بعدی، تولید ترکیبات و انجام محاسبات ماتریسی بسیار مفید است.

```
for i in range(1, 6):
    for j in range(1, 6):
        print(f"{i} × {j} = {i*j}", end="\t")
    print()
```

پیچیدگی زمانی: حلقه‌های تو در تو می‌توانند پیچیدگی زمانی $O(n^2)$ یا بیشتر داشته باشند، بنابراین برای داده‌های بزرگ باید با احتیاط استفاده شوند.

تمرین nested for

مینی پروژه

THANK YOU

h.farhadi.py@gmail.com