اتولیسپ : درس هشتم

نویسنده : عبدالحکیم قدس

در این بخش می پردازیم به چند تابع کاربردی در اتولیسپ. برای شروع تابع **distance** فاصله بین دو نقطه را در سه بعد محاسبه کرده و باز می گرداند.

```
(setq point1 (list 0 0))                    (0 0)
(setq point2 (list 3 4))                    (3 4)
(distance point1 point2)                      5.0
```

اگر قائده وترها و اضلاع مثلث را بخاطر بیاورید جذر مجذور دو ضلع برابر طول وتر و در حقیقت فاصله دو نقطه خواهد بود.

دقت کنید از این پس متغیر سازی به یکی از ملزومات کار شما تبدیل می شود پس در انتخاب نام متغیر خود به چند مورد توجه کنید. اول اینکه از عبارات مخفف شده استفاده نکنید چراکه به زودی از خاطر شما خواهند رفت و به محض اینکه چند روز از نوشتن برنامه بگذرد حتی شما هم قادر به فهم آن نخواهید بود. دوم اینکه اگر نام متغیر شما دو قسمتی است مثل **Center Point** بهتر است به شکل **CenterPoint** نامگذاری کنید. گرچه اتولیسپ بین حروف بزرگ و کوچک فرقی نمی گذارد ولی این روش به شما کمک می کند تا عبارت را راحت تر بخوانید و نیز به برنامه زیبایی و نظم می بخشد. از قرار دادن زیر خط بین دو نام **Center** و **Point** به شکل **Center_Point** خودداری کنید. چراکه سرعت تایپ شما را کاهش می دهد و در ضمت معمولا در نام توابع یا فایلها از این روش استفاده می شود. در کل بهتر است یک رویه خاص را انتخاب کرده و همیشه از آن پیروی کنید و در این مورد بهتر است از رویه معمول همه ی برنامه نویسان این رشته بهره جویید. نکته ی دیگری که باید بخاطر داشته باشید همنام نبودن نام تابع شما با یکی از توابع موجود در اتوکد می باشد. این عمل اشکالات جدی در اجرای کدهای اتوکد بوجود می آورد.

روش دیگری برای مختصات دهی به یک نقطه وجود دارد.

```
(setq point1 '(2 3 7))
(setq point2 '(9 0 1))
```

توجه کنید در مثال بالا مختصات نقاط در سه بعد داده شده است. کاراکتر ' در واقع کار List را انجام می دهد از این پس بجای List می توانید از این روش استفاده کنید. دو مثال زیر معادلند.

```
(setq mylist (list "Hakim" 21))                    ("hakim" 21)
(setq myList1 '("hakim" 21))                       ("hakim" 21)
```

به جای فرمان Princ هم می توانید از عملگر ! استفاده کنید. به نمونه های توجه کنید.

```
(princ pi)                                           3.14159
!pi                                                  3.14159

(setq myCenterPoint '(3 15 27))
!myCenterPoint                                    (3 15 27)
```

بعد از تابع distance که فاصله بین دو نقطه را باز می گرداند تابع angle زاویه بین *خط حاصل از دو نقطه مصرفی شده* و *خط پیشفرض افق* را بازمیگرداند.

```
(setq p1 '(0 0))
(setq p2 '(0 7))

(angle p1 p2)                                        1.5708
```

عبارت بالا زاویه را برحسب رادیان نمایش می دهد . برای تبدیل آن به درجه ی ده دهی از روش زیر استفاده می کنیم.

```
(angtos (angle p1 p2) 0 1)                           "90"
```

برای نمایش متن به کاربر علاوه بر princ می توان از prompt استفاده کرد. در واقع چندان تفاوتی بین این دو فرمان وجود ندارد.

```
(prompt "Your Name :")                          Your Name :nil
```

برای پیشگیری از نمایش nil در انتهای مقدار برگشتی یک عبارت princ بعد از فرمان اظافه کنید.

```
(prompt "Your Name :")(princ)                    Your Name :
```

در ادامه ی این بخش می پردازیم به تابع GetPoint که از کاربر می خواهد که یک نقطه را انتخاب کند.

```
(getpoint "Pick a point")          نشانگر موس به حالت انتخاب نقطه می رود
```

برای بکارگیری نتیجه حاصل از نقطه انتخاب شده باید آن را به یک متغیر نسبت دهیم.

```
(setq MyPoint (getPoint "pick a Point : "))
```