Proqramlaşdırmada bir neçə paradiqma var, lakin bunlardan ən fundamental olanları 4-dür:

1. İmperative və ya prosedural
2. Functional
3. Object oriented
4. Logical

Bunların hər birini izah etməzdən əvvəl ilkin olaraq paradiqma haqqında anlayışımız olması lazımdır. Paradiqma elmi olaraq, gerçəkliyin qəbulunu , qavranılmasını təmin edən modeldir. Proqramlaşdırmaya baxsaq, bu fikir dəyişir, yəni proqramlaşdırmada problemlərin həllini necə kod halına sala bilmək sualına cavab tapılması üçün yaradılan stillərdir. Paradiqmalar proqramlaşdırmada hər hansı problemin həllinə necə çıxma yollarını tərif etməklə yanaçı eyni zamanda hansı yollarla getmək olmaz prinsipinə əsaslanır. Bunu belə başa düşmək olar ki, functional paradiqması ilə işləyən bir proqramlaşdırma dilində digər paradiqmalara əsaslanan stildə kod yazmağa dil icazə vermir.

1. İmperativ və ya prosedural paradiqma. Bu məntiqlə işləyən bir dildə əmrlər sətir sətir yazılır və proqram da eyni stildə oxuyur. Məsələn, sürücüsüz bir maşının görəcəyi iş sadəcə maşını önə və geriyə hərəkət etdirməkdir. Bunu etmək üçün bu cür kodlaşdırmaq lazımdır : 1. Motoru işə sal. 2. Sürət qutusunu dəyiş, 3. Qaza bas, 4. əyləcə bas, 5. Yenə sürət qutusunu dəyiş. Bu cür proqramlamaya eyni zamanda prosedural proqramlama da deyilir, çünki kompyuterlərin işləmə prinsipinə ən uyğundur. İmperativ proqramlama kompyuterin həyata keçirəcəyi sıralı əmrlərdir. Hesablamanın “necə?” ediləcəyi önəmlidir. İmperativ proqramlamada verdiyimiz hər bir dəyər yaddaşda müəyyən yer tutur və input və output dəyərlərinin nəticəsini yaddaşda görə bilərik. Yaddaşın necə yönləndirilməsi əvvəəlcədən proqramçı tərəfindən təyin olunur. C, Pascal, Cobol imperativ paradiqması ilə işləyən dillərdir.
2. Functional proqramlama paradiqması. Günümüzdə təqribən bütün proqramlama dillərində istifadə olunan paradiqmadır. Tamamilə riyazi funksiyalara oxşar məntiqlə işləyir. Bu paradiqma sadə dillə desək, kod təkrarını azaltmaq və eyni koddan dəfələrlə istifadə etmək , kod analizini rahatlaşdırmaq və kodun oxunaqlığını artırmaq, eyni zamanda modulyar yəni daşına bilərlilik üçün yaradılmışdır. Haskell, Lisp, Java, C++ kimi dillər functional dillərdir. Funksional dillərdə yaddaşın istiafdəsi tamamilə avtomatikdir, yaddaşdan və prosessordan istifadədə daha az məhsuldardır.
3. Object oriented (obyekt yönümlü) – 60-cı illərdə kodlamanın qarmaqarışıq bir hala gəldiyi vaxtda kodlama keyfiyyətini və yenidən istiafəd edilə bilərliliyin qorumaq məqsədilə qurulub. Bu cür proqramlamada hər şeyə obyekt gözü ilə yanaşılır və görülən işlər əlimizdə var olan obyektlər və onların dəyişənləri üzərində aparılır. Object oriented dillər class məntiqi ilə işləyir və müxtəlif yerlərdə istifadə edəcəyimiz obyektin ümumi halın kodlamaqla istənilən yerdə onun arqumentlərin dəyişməklə ondan təkrar-təkrar istifadə edə bilərik. Bu məntiqin ən önəmli hissəsi classlarda verilən dəyərlərin yaddaşda xüsusi yer tutmaması yalnız çağırılan anda müəyyən həcmə malik olmasıdır. Məsələn: hər hansı bir kompyuter oyunu düşünün və bu oyunu yazmaq üçün prosedural paradiqmasını seçirəm bu zaman həm vaxt baxımından həm də oyunun tutacağı yer baxımından fantastik rəqəmlər ortaya çıxacaq, object oriented paradiqmasında isə classlar yaradaraq eyni obyektdən təkrar-təkrar istənilən yerdə və çox az yaddaş həcmi ilə istiafdə edə bilərik. Bununla yanaşı object oriented paradiqmada iş bölgüsü aspektindən bir qrup proqramçı eyni anda müxtəlif işləri görüb birləşdirə bilər, imperativdə isə hər çey sıralı olduğu üçün adamların belə bir lüksləri olmayacaq.
4. Məntiqi paradiqma sırf riyazi aksiomlar və qanunlar üzərində qurulan dillər aiddir. Burda əsas götürülən hər hansı aksiomun input olunması və hədəf cümlədə daxil olunanların düz olub olmamasını yoxlamaqdır. Məntiqi paradiqma ilə işləyən dillərə Prolog, ECLİPSe, Leda kimi dillər misal ola bilər.