**IMPERATIVE PROGRAMMING**

Imperative programming, programming with commands. We need to code what to do individually.

Imperative programming is a paradigm of computer programming in which the program describes a sequence of steps that change the state of the computer. Unlike declarative programming, which describes "what" a program should accomplish, imperative programming explicitly tells the computer "how" to accomplish it. Programs written this way often compile to binary executables that run more efficientlysince all CPU instructions are themselves imperative statements.

Imperative programming languages:

* [Ada](http://www.computerhope.com/jargon/a/ada.htm)
* ALGOL
* [Assembly language](http://www.computerhope.com/jargon/a/al.htm)
* [BASIC](http://www.computerhope.com/jargon/b/basic.htm)
* Blue
* [C](http://www.computerhope.com/jargon/c/c.htm)
* [C#](http://www.computerhope.com/jargon/c/csharp.htm)
* [C++](http://www.computerhope.com/jargon/c/cplus.htm)
* [COBOL](http://www.computerhope.com/jargon/c/cobol.htm)
* [D](http://www.computerhope.com/jargon/d/dlang.htm)
* [FORTRAN](http://www.computerhope.com/jargon/f/fortran.htm)
* [Go](http://www.computerhope.com/jargon/g/go.htm)
* Groovy
* [Java](http://www.computerhope.com/jargon/j/java.htm)
* Julia
* Lua
* [MATLAB](http://www.computerhope.com/jargon/m/matlab.htm)
* Modula
* MUMPS
* [Nim](http://www.computerhope.com/jargon/n/nim.htm)
* Oberon
* OCaml
* [Pascal](http://www.computerhope.com/jargon/p/pascal.htm)
* [Perl](http://www.computerhope.com/jargon/p/perl.htm)
* [PHP](http://www.computerhope.com/jargon/p/php.htm)
* PROSE
* [Python](http://www.computerhope.com/jargon/p/python.htm)
* [Ruby](http://www.computerhope.com/jargon/r/ruby.htm)
* [Rust](http://www.computerhope.com/jargon/r/rust.htm)

#PYTHON

#Kullanıcı ismi ve parola kontrolü

defkullanıcı=”değer”

defparola=”9999”

kullanıcı=input(“kullanıcı adı:”)

parola=input(“parolanız:”)

if ((defkullanıcı==kullanıcı) and (parola==defparola)):

print(“siteye hoşgeldiniz.”)

elif ((defkullanıcı!=kullanıcı) and (parola==defparola)):

print(“kullanıcı adını yanlış girdiniz.”)

elif ((defkullanıcı!=kullanıcı) and (parola!=defparola)):

print(“şifrenizi mi unuttunuz?”)

else:

print(“tekrar deneyiniz.”)

**DECLARATIVE PROGRAMMING**

Declarative programming refers to programming with recipe. The environment itself makes what you want.

Declarative programming is a [computer programming](http://www.computerhope.com/jargon/p/progming.htm) paradigm that the [developer](http://www.computerhope.com/jargon/d/develope.htm) defines what the program should accomplish rather than explicitly defining how it should go about doing so. This approach lends itself naturally to the programmatic definition of formal logic systems, and has the benefit of simplifying the programming of some [parallel processing](http://www.computerhope.com/jargon/p/paraproc.htm) applications.

Declarative programming languages:

* ABSET
* Absys
* Alpha
* Ant
* ASCEND
* Atom
* ATS
* Brooks
* Candle
* Curry
* CLP(R)
* Curl
* CycL
* Datalog
* DASL
* Dependent ML
* ECL
* Embedded SQL
* Erlang
* EAML
* F-Logic
* FXML
* GeneXus
* Glowe
* GOAL
* Gofer
* GtkBuilder
* Harbour
* HiLog
* HPCC
* JavaFX Script
* JModelica
* KM
* Lithe
* LOOM
* Lucid
* Lustre
* MetaFont
* MetaPost
* Miranda
* Modelica
* MXML
* Oz
* Pan
* Prolog
* Prova
* PTQL
* .QL
* QML
* Quark Framework
* QUILL
* RDQL
* SequenceL
* SIGNAL
* SMIL
* SPARQL
* SQL
* Transaction logic
* Web Ontology Language
* XBase
* XProc
* XSLT

% PROLOG

% Aile ağacı

female(dilek).

female(deger).

male(aydin).

male(arif).

parent(dilek,deger).

parent(dilek,arif).

parent(aydin,deger).

parent(aydin,arif).

child(deger,dilek).

child(arif,dilek).

child(deger,aydin).

child(arif,aydin).

sibling(arif,deger).

mother(X,Y):-

female(X),

parent(X,Y).

father(X,Y):-

male(X),

parent(X,Y).

brother(X,Y):-

male(X),

parent(Z,X),

parent(Z,Y),

X \= Y.

sister(X,Y):-

female(X),

parent(Z,X),

parent(Z,Y),

X \= Y.

daughter(X,Y):-

female(X),

parent(Y,X).

son(X,Y):-

male(X),

parent(Y,X).