Coding functions	
Function signature	**Description**
alternant_code (h, a, r) alternant_code (h, a, r, K) AC = Alternant_Code = alternant_code	If h and a are vectors of the same length, say n, with entries in a finite field F and r is a positive integer such that r ≤ n, alternant_code(h,a,r) constructs the alternant code A_K(h,a,r) associated to h, a and r over a ground field K that is assumed to be known from the context (usually K = base(F), the field out of which F has been constructed). The call alternant_code(h,a,r,K), where K is a subfield of F, is defined to be C = alternant_code(h, a,r) together with the setting C.K_ = K.
BCH (a, d, l) BCH (a, d, l, K) BCH (a, d) BCH (a, d, K) BCH_Code = BCH	If a is a non-zero element of a finite field F, d > 0 an integer such that d < n = period(a), BCH(a,d,l) is defined as alternant(h,a,d-1), where h = geometric_series(a^l,n) and a = geometric_series(a,n). This code is the BCH code of length n = period(a), design distance d and offset l. On the other hand, BCH(a,d) = BCH(a,d,1), which is called the strict BCH code of length n = period(a) and design distance d. Finally BCH(a,d,l,K) is equivalent to BCH(a,d,l).K_= K.
RS (a, k) Reed_Salomon_Code = RS PRS (F, r)	Given a finite field K, a vector a ∈ K^n, RS(a,k) is C = alternant_code(h,a,n-k) extended with the assignements C.G_ = vandermonde(a,k) and C.K_ = field(a), where h is the vector in K^n such that h_i = 1 / Πj≠i (a_i - a_j). Note that its generating matrix is included in the data of the code, and also the base field K (since it can be inferred from the vector a, it is not necessary to include it as parameter of the function RS). If F is a finite field, q its cardinal, ω a primitive element of F, n = q−1 = period(ω), and a non-negative integer ≤ n, then PRS(F,r) = BCH(ω, r+1, F), which is the RS code of dimention n − r on the vector of non-zero elements of F.
GRS(h,a,k)	Let h and a are vectors of the same length n over a field a finite field F and assume that the components of a are pairwise distinct and those of h are all non-zero. Then the code constructed with GRS(h,a,k) is equivalent to alternant_code(h,a,n-k,F). Note that if all components of h are one, then RS(a,k).
goppa(g,a) Goppa = Goppa_Classic_Code = goppa	The parameters of this funtion are a list or vector a of elements of a finite field F = K[x] that is a simple extension of K and a polynomial g ε F[T] such that g(t) ≠ 0 for any component t of a. Then the definition of goppa(g,a) is equivalent to alternant_code(h,a,r,K), where h is the vector [1/g(t) for t in a] and r is the degree of g.
hadamard_code(X)	It returns a hadamard matrix with -1 changed to 0 to representa the matrix of a hadamard code.
paley_code(F)	It returns the paley code matrix of the field F.
alternant_decoder (y, C) BMS(y,C) AD(y,C)	If C is an alternant code and y is the received vector, alternant_decoder (y,C) delivers the C-decoding of y according to the Berlekamp-Massey-Sugiyama algorithm. The names BMS and AD are synonyms of alternant_decoder.
PGZ(y,C) PGZm(y,C)	If C is an alternant code and y is the received vector, then both PGZ(y,C) and PGZm(y,C) deliver the C-decoding of y according to the Peterson-Gorenstein-Zierler algorithm. They share the same error-location scheme, but differ in the error evaluation: PGZ uses a variation of Forney's formula and PGZm uses linear algebra.

Coding functions	
Function signature	**Description**
RSID (a, k, y)	For RS codes, there is an interesting decoder based on polynomial interpolation. Since it only needs the vector α and the dimension k, it can be implemented as a function RSID(a,k,y), where a stands for the vector α and y for the received vector. The PyECC implementation is based on the following scheme. Let n = len(a) and t = (n − k) // 2. Then there are polynomials P = p_0 + p_1 T + ⋯ + p_{n-t-1} T^(n-t-1) and Q = q_0 + q_1 T + ⋯ + q_{n-t-k} T^(n-t-k), not both zero, such that P(a_j) + y_jQ(a_j) = 0 for j = 0, ..., n-1 (these conditions amount to a system of n linear homogeneous equations in the n-t + n-k-t+1 = n - k -2t +n+1 > n unknowns p_0, ..., p_{n-t-1}, q_0, ..., q_{n-t-k}). Let x and e be the sent and error vectors, respectively, so that y = x + e. By definition of the RS codes, there is a well-defined polynomial f(T) ∈ F[T]k, where F is the finite field in which the aj have been chosen and F[T]k is the space of polynomials with coefficients in F of degre < k, such that x_j = f(a_j). Therefore we have the relations P(a_j) + (f(a_j) + e_j) Q(a_j) = 0 for j = 0, ..., n-1. This implies that the polynomial g(T) = P (T) + f(T) Q(T) vanishes for all a_j such that e_j = 0. Since g(T) has degree < n − t, we conclude that g(T) = 0 if the number n − |e| ≥ n − t , that is, if |e| ≤ t. Under this assumption we conclude that we can recover f(T), and hence x, as f(T) = - P(T) / Q(T).
hadamard_decoder (y, H)	If H is a Hadamard matrix of order n and y is a binary vector of length n, the function decodes y with respect to the Hadamard code associated with H.
meggitt (y, g, E)	Given the Meggitt table E for the cyclic code of length n over a finite field K generated by the monic polynomial g, and a vector y ∈ Kn, this function decodes y according to Meggitt algorithm.
decoder_trial (C, s, K) decoder_trial(C,s) AD_checker(C,s,K) AD_checker(C,s)	If C is an alternant code (let n be its length) defined over the finite field K and s ≤ n a positive integer, decoder_trial(C,s,K) first generates a random vector x of C and a random error pattern e in K^n of weight s, then finds the value x* = alternant_decoder(x + e, C) and presents [x, e, x+e] if x* = x (correct decoding), [x, e, x+e, x*] if x* is a vector ≠ x (undetectable error), and [x, e] if x* is not a vector (decoder error). If the field K is included in the data of C, then we can use decoder_trial(C, s) instead, as this call is defined to be decoder_trial(C, s, K_ (C)). AD_checker is an alies of decoder_trial.
SHDT(C,s,K) SHDT(C,s)	If C is an alternant code (let n be its length) defined over the finite field K and s ≤ n a positive integer, SHDT creats a random word x of the code C and generates a random error e of weight s. Then, it calls the function alternant_decoder(x+e,C) to obtain a vector y. The function finally tests if y is equal to x to write if the decoding process was succesfull.
isbn(x)	It returns the International Standard Book Number checksum character of the value x.
weight_enumerator_mds(n,k,q) wieght_enumerator_mds(n,k)	It delivers the list of weights of and MDS code of length n and dimension k over q-ary alphabet. By default, q = 2.
min_weight(X)	given a list of code words, it returns the minimum weight in them.
min_weights(X)	Given a list of code words, it returns the ones that have minimum weight.

Coding functions	
Function signature	**Description**
hamming_weight_enumerator(r,q) hamming_weight_enumerator(r) HWE = hamming_weight_enumerator	It delivers the weight polynomial of a codimension r Hamming code over F_q. By default, q = 2.
macwilliams(n,k,A,q) macwilliams(n,k,A)	It returns the weight polynomial of the dual of an [n,k]_q code with weight enumerator polynomial A. By default q = 2.
ub_griesmer(n,d,q) ub_griesmer(n,d)	It computes the upper bound on B_q(n,d). By default q = 2.
ub_plotkin(n,d,q) ub_plotkin(n,d)	I computes the plotkin upper bound. By default q = 2.
ub_elias(n,d,q) ub_elias(n,d)	It computes the elias upper bound (Bassalygo-Elias bound). By default q = 2.
ub_johnson(n,d,w) ub_johnson(n,d)	It computes the Johnson upper bound of the function A(n, d, w) that is defined as the greatest cardinal M of a binary code (n, M) with the condition that its minimum distance is > d and all its vectors have weight w
singleton(x)	It computes the asymptotic Singleton upper bound.
plotkin(x)	It computes the asymptotic Plotkin upper bound.
gilbert(x)	It computes the asymptotic Gilbert-Varshamov lower bound.
hamming(x)	It computes the asymptotic Hamming upper bound.
elias(x)	It computes the asymptotic Elias upper bound.
valint(x)	It computes the asymptotic Van Lint upper bound.
mceliece(x)	It computes the asymptotic McEliece-Rodemich-Rumsey-Welch upper bound.
lookup(k,T)	It search the values associated to a key k in a table T given as a list of pairs.
shorthand(X,E)	It searches the values in the vector X or matrix X in the table E.
meggitt_decoder(y,g)	Decoder for cyclic codes. It presupposes having computed a meggit table E.
PRS_G(w,k)	It returns a generating matrix of a PRS code with parameters w and k.
mattson_solomon_matrix(w)	It returns the Mattson-Solomon matrix of the element w. If w is a domain, it returns the Mattson-Solomon matrix of a primitive root of the domain.
h_	It returns the h vector of the code C
a_	It returns the alpha vector of the code C

Coding functions	
Function signature	**Description**
r_	It returns the dimension of the code.
b_	It returns the beta vector of the code C
G_	It returns the generating matrix of a code C
H_(C)	Yields the control matrix of a code C.
set_G_(C,G)	Sets the generating matrix of a code C to G.