ШИНЖЛЭХ УХААН ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ
Мэдээлэл, Холбооны Технологийн Сургууль

F.CS213 Биоалгоритм

Basic Processing of Biological Sequences

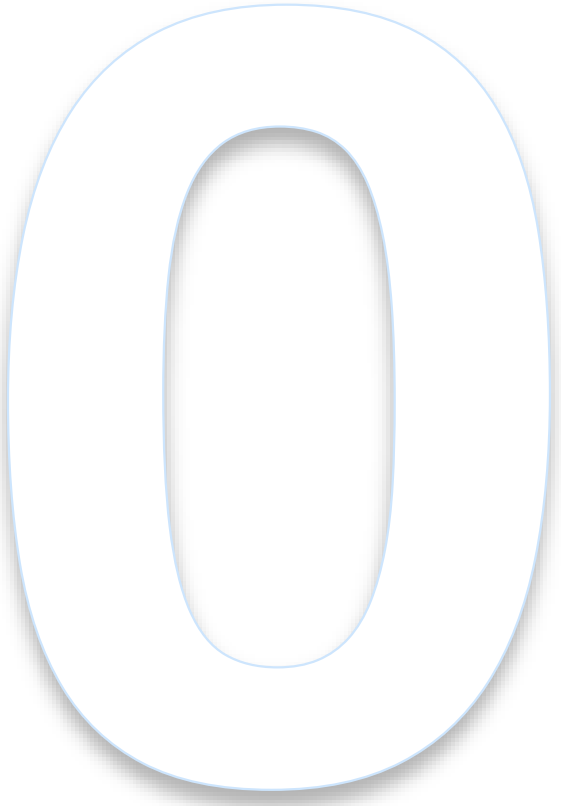
Биологийн дарааллын үндсэн боловсруулалт

Лекц 3

Багш ***Ганбаатарын ГАНБАТ.*** *Өрөө* ***#304,*** *ganbatg@must.edu.mn,* ***99999467***

Лекцийн агуулга

- Биологийн дараалал: Тэмдэг тэмдэглэгээ, үндсэн алгоритмууд
- Хөрвүүлэлт (Transcription) ба урвуу гүйцээлт
- Хувиргалт (Translation)
- Таамаг ген (Putative Genes) илрүүлэх
- Биологийн дарааллын класс
- BioPython ашиглан дараалал боловсруулах



Биологийн дараалал: Тэмдэг, тэмдэглэгээ

Олон янзын практик зорилгоор биоинформатикийн алгоритм, хэрэгслүүдэд генетикийн мэдээллэл бүхий молекулуудыг нуклеотидын *"нэг хэмжээст дараалал"* хэлбэрээр дүрсэлдэг.

Symbol	Name	Nucleotides represented
A	Adenine	A
C	Cytosine	C
G	Guanine	G
T	Thymine	T
U	Uracil	U
K	Keto	G, T
M	Amino	A, C
R	Purine	A, G
S	Strong	C, G
W	Weak	A, T
Y	Pyrimidine	C, T
B	Not A	C, G, T
D	Not C	A, G, T
H	Not G	A, C, T
V	Not T	A, C, G
N	Any base	A, C, G, T

Нуклеотидын IUPAC тэмдэглэгээ

- Нуклейтэдийн үндсэн 5 тэмдэглэгээ
- *International Union of Pure and Applied Chemistry - IUPAC* өргөтгөсөн тэмдэглэгээ
  - Нуклеотид нь тодорхой бус байгаа байрлалыг тэмдэглэхэд ашигтай,
  - Полимеразын гинжин урвалын праймер (primers)-ын загварт тохиромжтой.
- Бусад чухал биологийн дараалал бол уураг
  - Ихэвчлэн амин хүчлүүдийн дараалал
  - 20 амин хүчлийг тэмдэглэдэг
  - Зогсох кодын тэмдэглэгээ: _ эсвэл *
    - Жнь: ДНХ-ийн трансляшн хийх автомат хэрэгслийг ажиллуулахад ашигладаг

Symbol	Name
A	Alanine
C	Cysteine
D	Aspartic acid
E	Glutamic acid
F	Phenylalanine
G	Glycine
H	Histidine
I	Isoleucine
L	Lysine
M	Methionine
N	Asparagine
P	Proline
Q	Glutamine
R	Arginine
S	Serine
T	Threonine
V	Valine
T	Tryptophan
Y	Tyrosine

Амин хүчлийн IUPAC тэмдэглэгээ

БИО ДАРААЛАЛ Алгоритм

```
def validate_dna (dna_seq):
    """ Checks if DNA sequence is valid. Returns True is sequence is
   valid, or False otherwise. """
    seqm = dna_seq.upper()
    valid = seqm.count("A") + seqm.count("C") + seqm.count("G") +
   seqm.count("T")
    if valid == len(seqm): return True
    else: return False

>>> validate_dna("atagagagatctcg")
True
>>> validate_dna("ATAGAXTAGAT")
False
```

```
def frequency (seq):
    """ Calculates the frequency of each symbol in the sequence.
   Returns a dictionary. """
    dic = {}
    for s in seq.upper():
        if s in dic: dic[s] += 1
        else: dic[s] = 1
    return dic

>>> frequency("atagataactcgcatag")
{'A': 7, 'C': 3, 'G': 3, 'T': 4}
>>> frequency("MVVMKKSHHVLHSQSLIK")
{'H': 3, 'I': 1, 'K': 3, 'L': 2, 'M': 2, 'Q': 1, 'S': 3, 'V': 3}
```

- Функцийн толгой хэсэгт функцийн зориулалт, оролт, гаралтын тухай тайлбарыг бичдэг.

Жишээ

```
seq_aa = input("Protein sequence:")
freq_aa = frequency(seq_aa)
list_f = sorted(freq_aa.items(), key=lambda x: x[1], reverse = True)
for (k,v) in list_f:
    print("Aminoacid:", k, ":", v)
```

- Амин хүчлийн давтамжийг тоолно.
- Давтамжийг эрэмбээр хэвлэнэ.
- lambda тэмдэглэгээг хэрэглэсэн

```
def gc_content (dna_seq):
    """ Returns percentage of G and C nucleotides in a DNA sequence.
    """
    gc_count = 0
    for s in dna_seq:
        if s in "GCgc": gc_count += 1
    return gc_count / len(dna_seq)
```

- Дарааллын урттай харьцуулахад 'G' ба 'C' нуклеотидын хувь

```
def gc_content_subseq (dna_seq, k=100):
    """ Returns GC content of non-overlapping sub-sequences of size k
    . The result is a list. """
    res = []
    for i in range(0, len(dna_seq)-k+1, k):
        subseq = dna_seq[i:i+k]
        gc = gc_content(subseq)
        res.append(gc)
    return res
```

- Ихэнх тохиолдолд, ген эсвэл экзон хайхдаа бид дарааллын өөр өөр хэсэг дэх GC-г тоолох хэрэгтэй.
- *k* урттай хэсгүүд дэд GC-г тоолно.

Хөрвүүлэлт (Transcription) ба урвуу гүйцээлт

ДНХ-ийг РНХ молекул руу хөрвүүлэх: ДНХ-ийн молекулуудад агуулагдах генетик мэдээлэл нь уургийн нийлэгжих бүх процессын үндэс болдог.

def transcription (dna_seq) :

- DNA-ийн молекулууд нь урвуу чиглэлд уншигдах хоёр цуваатай.
- Хөрвүүлэлтийг үүсэх үед уг хоёр цуваа салж, шинэ RNA молекул нь нэг цувааны гүйцээлт болж үүснэ.
- Тиймээс ДНХ-ийн нөгөө цуваатай төстэй (бид үүнийг template цуваа гэдэг), ‘T’-ийн оронд 'U’ байна.

```
def reverse_complement (dna_seq):
    """ Computes the reverse complement of the DNA sequence. """
    assert validate_dna(dna_seq), "Invalid DNA sequence"
    comp = ""
    for c in dna_seq.upper():
        if c == 'A':
            comp = "T" + comp
        elif c == "T":
            comp = "A" + comp
        elif c == "G":
            comp = "C" + comp
        elif c== "C":
            comp = "G" + comp
    return comp
```

def reverse_complement (dna_seq) **:**

- Ерөнхийдаа биоинформатик өгөгдлийн санд зөвхөн нэг цувааг өгдөг нь тул шаардлагатай бол ДНХ нөгөөг нь тооцоолдог.

```
def transcription (dna_seq):
    """ Function that computes the RNA corresponding to the
    transcription of the DNA sequence provided. """
    assert validate_dna(dna_seq), "Invalid DNA sequence"
    return dna_seq.upper().replace("T","U")
```

Хувиргах (Translation)

Messenger RNA молекулуудад агуулагдах мэдээллийн дагуу амин хүчлүүдийн гинжийн дарааллыг үүсгэн уураг нийлэгжүүлдэг.

- Өмнөх алгоритмаас харахад DNA-ээс mRNA үүсгэх нь хялбар тул DNA-ээс шууд уураг үүсгэе.
  I. хувиргах гэж буй DNA дарааллыг гурав урттай (кодон гэж нэрлэдэг) давхцахгүй дэд дараалалд хуваана;
  II. кодон бүрийн хувьд тохирох амин хүчлийг хувиргах хүснэгтээр харгалзуулна;
  III. амин хүчлүүдийг дарааллаар нь авч уургийн дараалал залгана.

```
def translate_codon (cod):
    """Translates a codon into an aminoacid using an internal
   dictionary with the standard genetic code."""
    tc = {"GCT":"A", "GCC":"A", "GCA":"A", "GCG":"A",
      "TGT":"C", "TGC":"C",
      "GAT":"D", "GAC":"D"
                                    ...
      "TAA":"_", "TAG":"_", "TGA":"_"}
    if cod in tc: return tc[cod]
    else: return None
```

- Кодоноос амин хүчлүүд рүү хөрвүүлэх хүснэгтийг Python-ы dictionary-ээр хэрэгжүүлсэн
  - Түлхүүрүүд нь кодонууд (нийт 64), холбогдох утгууд нь амин хүчлүүд.
  - Буруу кодоныг аргумент болгон дамжуулсан тохиолдолд функц нь None-г буцаана.
  - Зогсох кодонд '_' тэмдгийг ашигласан

ХУВИРГАХ

Үргэлжлэл

- Өмнөх функцийг ашигласнаар ДНХ-ийн дарааллыг бүхэлд нь хөрвүүлэх функц бичих боломжтой болсон.
- ДНХ-ийн дараалал болон хувиргах анхны байрлалыг оролт болгон авч, холбогдох амин хүчлийн дарааллыг буцаана.
- Дарааллыг кодон болгон хуваах нь хамгийн сүүлийн аргумент нь алхамын утгыг тодорхойлох боломжийг олгодог **range** функцийг ашиглан тохирох индексүүдийг үүсгэх замаар хийгдэнэ.
- Хуваасан кодонуудын хувиргалт нь өмнө боловсруулсан функцийг буцаах утгаар ирнэ.

```
def translate_seq (dna_seq, ini_pos = 0):
    """ Translates a DNA sequence into an aminoacid sequence. """
    assert validate_dna(dna_seq), "Invalid DNA sequence"
    seqm = dna_seq.upper()
    seq_aa = ""
    for pos in range(ini_pos,len(seqm)-2,3):
        cod = seqm[pos:pos+3]
        seq_aa += translate_codon(cod)
    return seq_aa
```

ХУВИРГАХ

Кодоны хэрэглээ (codon usage)

```
def codon_usage(dna_seq, aa):
    """Provides the frequency of each codon encoding a given
   aminoacid, in a DNA sequence ."""
    assert validate_dna(dna_seq), "Invalid DNA sequence"
    seqm = dna_seq.upper()
    dic = {}
    total = 0
    for i in range(0, len(seqm)-2, 3):
        cod = seqm[i:i+3]
        if translate_codon(cod) == aa:
            if cod in dic:
                dic[cod] += 1
            else: dic[cod] = 1
            total += 1
    if total >0:
        for k in dic:
            dic[k] /= total
    return dic
```

- Генетик код нь олон сонголттой тул давтагдсан утгууд байна,
  - Жнь: нэг амин хүчлийг өөр өөр кодоноор кодлодог.
- Үүнийг *кодоны хэрэглээ (codon usage)* гэдэг бөгөөд янз бүрийн генүүдийн хувьд сонирхолтой статистикийг өгдөг.
- Функцийн үр дүнд dictionary өгөгдөл буцна
  - Түлхүүр нь кодон
  - Утга нь кодон бүрийн хэрэглээний хувь.

Таамаг ген (Putative Genes) илрүүлэх

Ерөнхий тохиолдолд DNA дараалллын (жнь, genome sequencing project) кодлох бүсүүд хаана байгаа нь урьдчилан мэддэгддэггүй.

- Уургийн хувиргалт нь үргэлж *Methionine ("M")* амин хүчлийн кодоноор (эхлэх кодон - "ATG") эхэлдэг.
- Энэ амин хүчил нь уургийн эхлэлээс гадна бусад байрлалд ч тохиолдож болно.
- Мөн зогсох кодон олдвол хувиргалты процесс дуусдаг байх ёстой.
- Уг сэжигтэй хэсгүүдийг хайхдаа эхлээд DNA (эсвэл RNA) дараалал дээрх *унших хүрээг (reading frames)* тооцоолох юм.
- Унших хүрээ нь ДНХ-ийн дарааллыг дараалсан давхцаагүй гурвалсан (боломжтой кодон) болгон хуваах арга юм.

```
def reading_frames (dna_seq):
    """Computes the six reading frames of a DNA sequence (including
   the reverse complement."""
    assert validate_dna(dna_seq), "Invalid DNA sequence"
    res = []
    res.append(translate_seq(dna_seq,0))
    res.append(translate_seq(dna_seq,1))
    res.append(translate_seq(dna_seq,2))
    rc = reverse_complement(dna_seq)
    res.append(translate_seq(rc,0))
    res.append(translate_seq(rc,1))
    res.append(translate_seq(rc,2))
    return res
```

- Өгөгдсөн дараалал нь эхний, хоёр, гурав дахь байрлалаас эхлэн унших боломжтой гурван тохиолдолтой.
- Энэ гурваас гадна урвуу гүйцээлтийнх нь гурван хүрээг тооцоолох хэрэгтэй.

- Ингэснээр геномын хэсэг дээрх уургийн байж болох бүх кодыг хайх боломжийг олгодог.

Уншилтын нээлттэй хүрээ хайх

Open Reading Frames-ORF хайх: Унших хүрээнүүдийг тооцоолсны дараагийн алхам бол эдгээр хүрээгээр кодлох боломжтой уураг олох явдал юм.

```
def all_proteins_rf (aa_seq):
    """Computes all possible proteins in an aminoacid sequence.
   Returns list of possible proteins. """
    aa_seq = aa_seq.upper()
    current_prot = []
    proteins = []
    for aa in aa_seq:
        if aa == "_":
            if current_prot:
                for p in current_prot:
                    proteins.append(p)
                current_prot = []
        else:
            if aa == "M":
                current_prot.append("")
            for i in range(len(current_prot)):
                current_prot[i] += aa
    return proteins
```

- Амин хүчлийн дарааллаас бүх боломжит уургийг гаргаж авна.
- Энэ функц нь дарааллаар явагддаг бөгөөд
- 'M' илэрсэн үед таамагласан уургийг *current_prot* жагсаалтад үүсгэж эхэлнэ
- Зогсох тэмдэг олдвол энэ жагсаалтын бүх уураг гаралтын хувьсагчид нэмнэ.

Бүх боломжит уургийг тооцоолох

```
def all_orfs (dna_seq):
    """Computes all possible proteins for all open reading frames."""
    assert validate_dna(dna_seq), "Invalid DNA sequence"
    rfs = reading_frames (dna_seq)
    res = []
    for rf in rfs:
        prots = all_proteins_rf(rf)
        for p in prots: res.append(p)
    return res
```

- Эхлээд унших хүрээнүүдийн хөрвүүлэлтийг тооцоолно
- Дараа нь зургаан амин хүчлийн дараалал дээр давталт хийнэ
- Өмнөх функцийг ашиглан бүх боломжит уургийг олж, үр дүнгийн жагсаалтад цуглуулна.

- Уг функцийн буцаах жагсаалт нь бодит тохиолдолд олон тооны уураг агуулж болзошгүй
- Жагсаалтыг таамагласан уургийн хэмжээгээр эрэмбэлж, хамгийн бага хэмжээтэйгээс нь эхлэх хэрэгтэй.
- Жижиг таамаг уургууд нь санамсаргүй байдлаар олдож болох ч хэдэн арван амин хүчил агуулсан уураг ингэж тохиолдох магадлал багатай юм.
- Зогсоох кодон үүсэх магадлал 3/64 буюу 5% орчим байдаг тул ойролцоогоор 20 амин хүчил тутамд зогсолт кодон үүсэх төлөвтэй байдаг.
- Энэ нөхцлийг оруулж хөгжүүлсэн хувилбарыг сурах бичиг дээрээс үзнэ үү. 4-р бүлэг, 89-р хуудас, def all_orfs_ord (dna_seq, minsize = 0)

Био дарааллын класс

Илүү зохион байгуулалттай, цаашид хэрэглэх боломжтой КЛАСС бүтэц

Объект хандалгат программчлал

```
class MySeq:
    """ Class for biological sequences. """

    def __init__ (self, seq, seq_type = "DNA"):
        self.seq = seq.upper()
        self.seq_type = seq_type

    def __len__(self):
        return len(self.seq)

    def __getitem__(self, n):
        return self.seq[n]

    def __getslice__(self, i, j):
        return self.seq[i:j]

    def __str__(self):
        return self.seq

    def get_seq_biotype (self):
        return self.seq_type

    def show_info_seq (self):
        print ("Sequence: " + self.seq + " biotype: " + self.seq_type
    )
```

Гишүүн функцүүд

```
    def alphabet (self):
        if (self.seq_type=="DNA"): return "ACGT"
        elif (self.seq_type=="RNA"): return "ACGU"
        elif (self.seq_type=="PROTEIN"): return "ACDEFGHIKLMNPQRSTVWY
    "
        else: return None

    def validate (self):
        alp = self.alphabet()
        res = True
        i = 0
        while i < len(self.se
            if self.seq[i] no
            else: i += 1
        return res
```

```
    def transcription (self):
        if (self.seq_type == "DNA"):
            return MySeq(self.seq.replace("T","U"), "RNA")
        else:
            return None

    def reverse_comp (self):
        if (self.seq_type != "DNA"): return None
        comp = ""
        for c in self.seq:
            if (c == 'A'): comp = "T" + comp
            eli
            eli
            eli
        return
```

```
    def translate (self, iniPos= 0):
        if (self.seq_type != "DNA"): return None
        seq_aa = ""
        for pos in range(iniPos,len(self.seq)-2,3):
            cod = self.seq[pos:pos+3]
            seq_aa += translate_codon(cod)
        return MySeq(seq_aa, "PROTEIN")
```

Классыг хэрэглэх

```
s1 = MySeq("ATGTGATAAGAATAGAATGCTGAATAAATAGAATGACAT")
s2 = MySeq("MKVVLSVQERSVVSLL", "PROTEIN")
print(s1.validate(), s2.validate())
print(s1)
s3 = s1.transcription()
s3.show_info_seq()
s4 = s1.reverse_comp().translate()
s4.show_info_seq()
```

http://www.biopython.org/

CoLab: !pip3 install biopython

```
>>> import Bio
>>> print(Bio.__version__)
...
```

Дараалалтай ажиллах

- Биоинформатикийн гол объект нь дараалал

```
>>> from Bio.Seq import Seq
>>> my_seq = Seq("AGTACACTGGT")
>>> my_seq
Seq('AGTACACTGGT')
>>> print(my_seq)
AGTACACTGGT
```

- Seq объектийг хэрэглэх

```
>>> my_seq
Seq('AGTACACTGGT')
>>> my_seq.complement()
Seq('TCATGTGACCA')
>>> my_seq.reverse_complement()
Seq('ACCAGTGTACT')
```

Биологийн өгөгдлийн сантай холбогдох

- Bio.Entrez.esearch() – NCBI Entrez (and PubMed)
- Bio.ExPASy.get_sprot_raw()
- Bio.SCOP.search()

Дарааллын файл форматыг задлах

- Blas, Clustalw, FASTA, GenBank, PubMed and Medline, ExPASy files, SCOP, UniGene, SwissProt

```
>gi|2765658|emb|Z78533.1|CIZ78533 C.irapeanum 5.8S rRNA gene and ITS1 and ITS2 DNA
CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGGAATAAACGATCGAGTG
AATCCGGAGGACCGGTGTACTCAGCTCACCGGGGGCATTGCTCCCGTGGTGACCCTGATTTGTTGTTGGG
...
```

- Жишээ

```
>>> from Bio import SeqIO
>>> for seq_record in SeqIO.parse("ls_orchid.fasta", "fasta"):
...     print(seq_record.id)
...     print(repr(seq_record.seq))
...     print(len(seq_record))
...
```

```
gi|2765658|emb|Z78533.1|CIZ78533
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGG...CGC')
740
...
gi|2765564|emb|Z78439.1|PBZ78439
Seq('CATTGTTGAGATCACATAATAATTGATCGAGTTAATCTGGAGGATCTGTTTACT...GCC')
592
```

```
>>> from Bio import SeqIO
>>> for seq_record in SeqIO.parse("ls_orchid.gbk", "genbank"):
...     print(seq_record.id)
...     print(repr(seq_record.seq))
...     print(len(seq_record))
...
```

```
Z78533.1
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGG...CGC')
740
...
Z78439.1
Seq('CATTGTTGAGATCACATAATAATTGATCGAGTTAATCTGGAGGATCTGTTTACT...GCC')
592
```

BIOPYTHON Дараалал объект: **Seq**

```
>>> from Bio.Seq import Seq
>>> coding_dna = Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG")
>>> coding_dna
Seq('ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG')
>>> messenger_rna = coding_dna.transcribe()
>>> messenger_rna
Seq('AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG')
>>> messenger_rna.translate()
Seq('MAIVMGR*KGAR*')
```

```
>>> from Bio.Data import CodonTable
>>> standard_table = CodonTable.unambiguous_dna_by_name["Standard"]
>>> mito_table = CodonTable.unambiguous_dna_by_name["Vertebrate Mitochondrial"]
>>> print(mito_table)
Table 2 Vertebrate Mitochondria

  |  T      |  C      |  A      |  G      |
--+---------+---------+---------+---------+--
T | TTT F   | TCT S   | TAT Y   | TGT C   | T
T | TTC F   | TCC S   | TAC Y   | TGC C   | C
T | TTA L   | TCA S   | TAA Sto | TGA Trp | A
T | TTG L   | TCG S   | TAG Sto | TGG W   | G
--+---------+---------+---------+---------+--
C | CTT L   | CCT P   | CAT H   | CGT R   | T
```

```
>>> print(standard_table)
Table 1 Standard, SGC0

  |  T      |  C      |  A      |  G      |
--+---------+---------+---------+---------+--
T | TTT F   | TCT S   | TAT Y   | TGT C   | T
T | TTC F   | TCC S   | TAC Y   | TGC C   | C
T | TTA L   | TCA S   | TAA Stop| TGA Stop| A
T | TTG L(s)| TCG S   | TAG Stop| TGG W   | G
--+---------+---------+---------+---------+--
C | CTT L   | CCT P   | CAT H   | CGT R   | T
```

- Дараалал ерөнхийдөө тэмдэгт мөр
- Дарааллын хэсгийг хуулах
- Тэмдэгт мөрөнд буцаах
- Дарааллуудыг залгах болон дараалал нэмэх
- Дарааллын утгыг томруулах, жижгэрүүлэх
- (Урвуу) Гүйцээлт
- Хөрвүүлэлт, Хувиргалт
- Хувиргалтын хүснэгтүүд
- Seq объектуудыг харьцулах
- Утга нь үл мэдэгдэж байх үед ашиглах дараалал
- MutableSeq объект – Дарааллын утгыг засварлах
- Тэмдэгт мөрийг дарааллын оронд шууд ашиглах

UnknownSeq objects - Deprecated

BIOPYTHON

Дараалын аннотоц объектууд

Bio.SeqRecord **нь дараалал ба тэдгээрийн аннотацын үндсэн контейнер класс.**

SeqRecord

- *.seq* – дараалал өөрөө буюу Seq классын объект
- *.id* – дарааллын дугаар
- *.name* – дарааллын нэр
- *.description* – дарааллын тайлбар
- *.annotations* – бүхэл дарааллын глобал тайлбарууд, dictionary төрөлтэй,
  - *Түлхүүр:* аннотацын төрлүүд болон бүтэцлэгдээгүй талбарууд
  - *Утга:* тухайн дарааллын талбаруудын утгууд
- *.features* – бүтэцлэгдсэн үзүүлэлтүүд, дараалалд эсвэл хэсгүүдэд нь хамаарах SeqFeature объектын жагсаалт;
- *.letter_annotations* – дараалал дахь үсэг (байрлал) бүрийн боломжит аннотацууд;
- *.dbxrefs* – өгөгдлийн сангийн лавлах мэдээлэл

SeqFeature

- *.location* – энэ аннотацад хамаарах дарааллын бүс, FeatureLocation классын объект;
- *.type* – тухайн үзүүлэлтийн төрлийн төлөв, тэмдэгт мөр;
- *.qualifiers* – тухайн үзүүлэлтийн талаар нэмэлт мэдээлэл, талбар-утга dictionary төрөлтэй.

АНХААРАЛ ТАВЬСАНД БАЯРЛАЛАА