Zhang-Lab 生信小课堂 第三期

和趣求真 Z 秉实生信

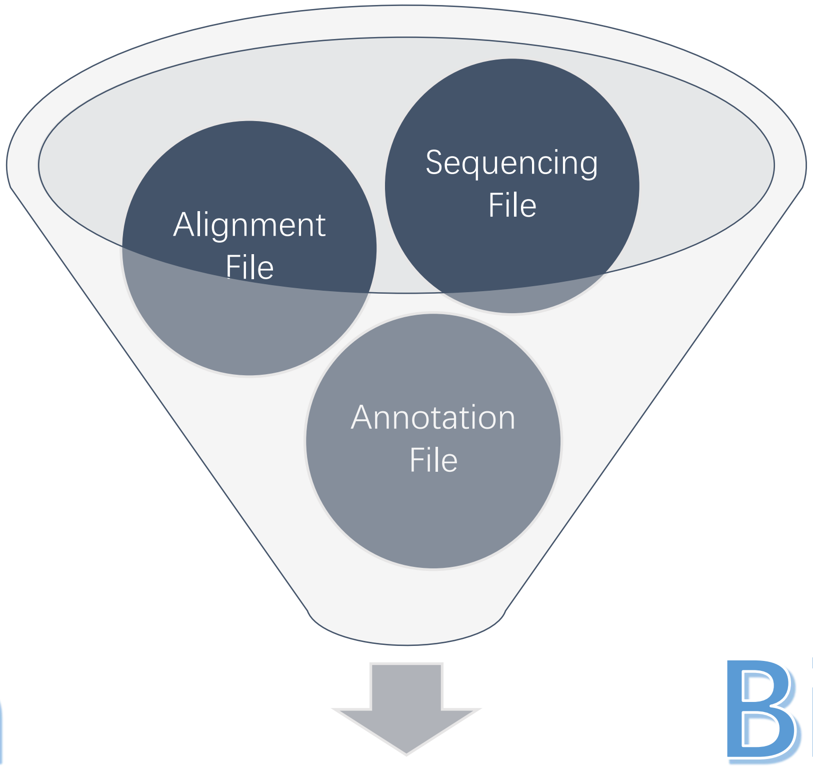
Marvellous Python

Pysam & Biopython for Bioinformatics

黄臆丞

2022/05/06

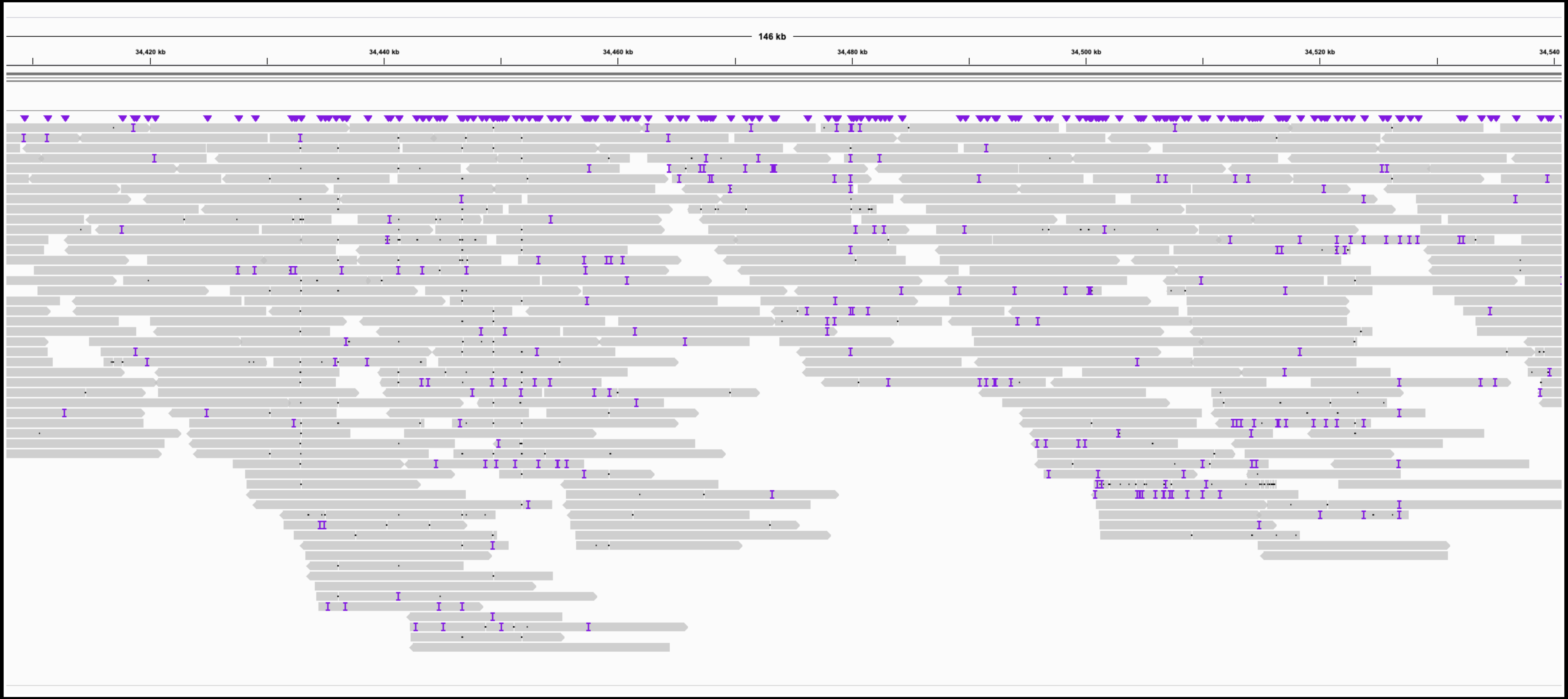
Bioinformatics



python

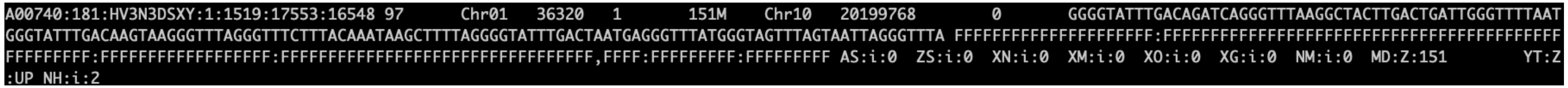
Pysam

Biopython



SAM/BAM files (IGV SAMtools)

SAM/BAM files



```
A00740:181:HV3N3DSXY:1:1519:17553:16548 97      Chr01   36320   1       151M    Chr10   20199768        0       GGGGTATTTGACAGATCAGGGTTTAAGGCTACTTGACTGATTGGGTTTTAAT
GGGTATTTGACAAGTAAGGGTTTAGGGTTTCTTTACAAATAAGCTTTTAGGGGTATTTGACTAATGAGGGTTTATGGGTAGTTTAGTAATTAGGGTTTA FFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFF:FFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF,FFFF:FFFFFFFFF:FFFFFFFFF AS:i:0  ZS:i:0  XN:i:0  XM:i:0  XO:i:0  XG:i:0  NM:i:0  MD:Z:151        YT:Z
:UP NH:i:2
```

QNAME，比对片段的（template）的编号

FLAG，位标识，template mapping情况的数字表示

RNAME，参考序列的编号

POS，比对上的位置，注意是从1开始计数，没有比对上，此处为0；

MAPQ，mapping的质量；

CIGAR，简要比对信息表达式

RNEXT，下一个片段比对上的参考序列的编号

PNEXT，下一个片段比对上的位置，如果不可用，此处为0，

TLEN，Template的长度，最左边得为正，最右边的为负，不可用时为0

SEQ，序列片段的序列信息，如果不存储此类信息，此处为' * '

QUAL，序列的质量信息，格式同FASTQ一样

AS:i 匹配的得分; XS:i 第二好的匹配的得分; YS:i mate 序列匹配的得分; XN:i 在参考序列上模糊碱基的个数; XM:i 错配的个数; XO:i gap open的个数; XG:i gap 延伸的个数; NM:i 经过编辑的序列; YF:i 说明为什么这个序列被过滤的字符串; MD:Z 代表序列和参考序列错配的字符串

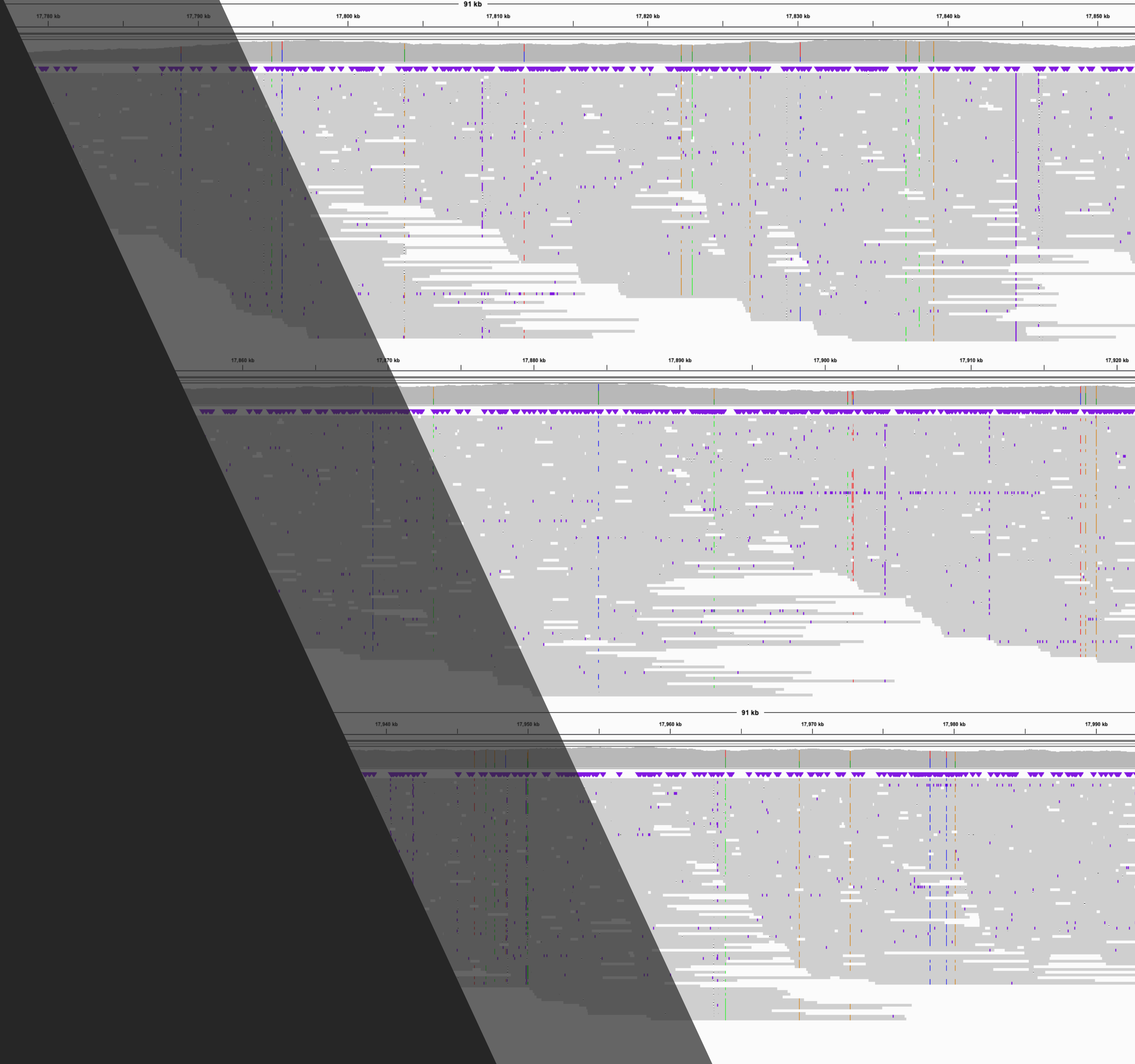
General stuff

- Statistics
- View
- …

Specific operations

- Read cigar string
- count the coverage
- Fetching reads
- …

Specific reads for special location



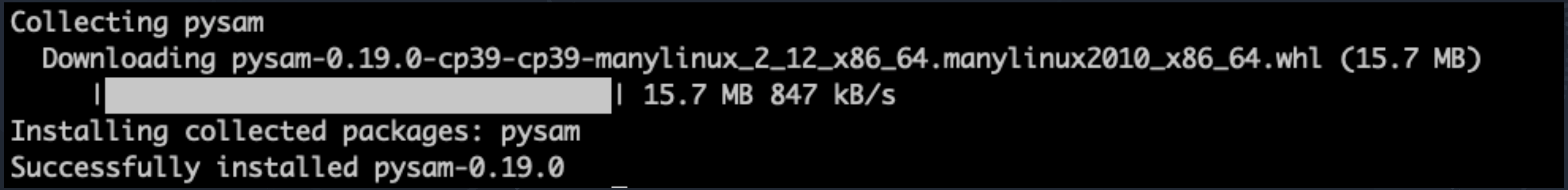
Pysam

- Pysam is a python module that makes it easy to read and manipulate mapped short read sequence data stored in SAM/BAM files.

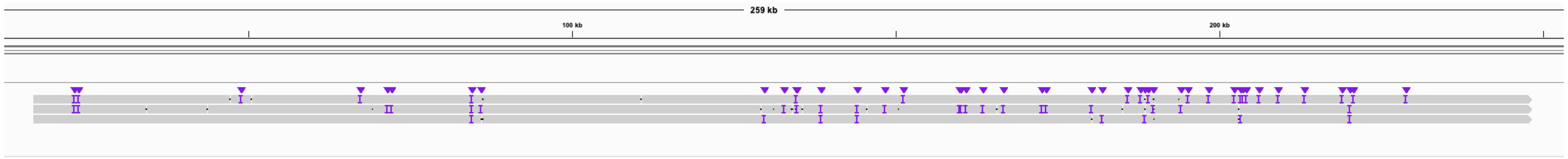
https://pysam.readthedocs.io/en/latest/index.html

Install

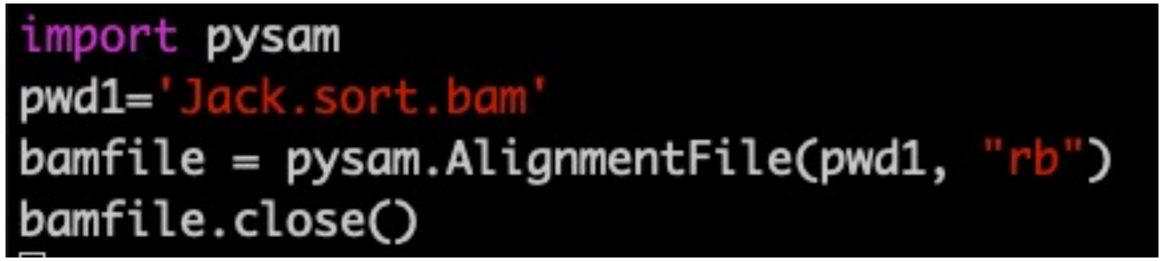
- pip install pysam



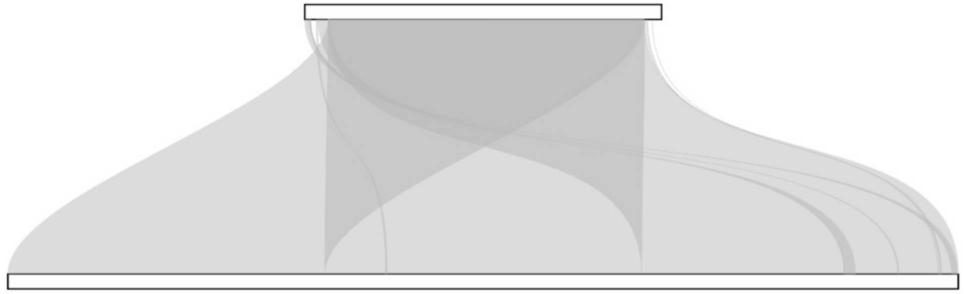
```
Collecting pysam
  Downloading pysam-0.19.0-cp39-cp39-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (15.7 MB)
     |████████████████████████████████| 15.7 MB 847 kB/s
Installing collected packages: pysam
Successfully installed pysam-0.19.0
```

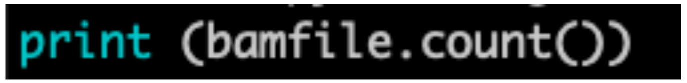


count(*self*, *contig=None*, *start=None*, *stop=None*, *region=None*, *until_eof=False*, *read_callback='nofilter'*, *reference=None*, *end=None*)

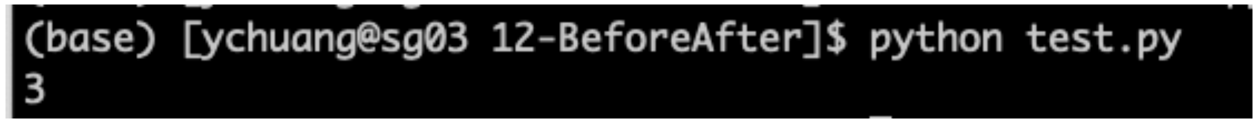


```
import pysam
pwd1='Jack.sort.bam'
bamfile = pysam.AlignmentFile(pwd1, "rb")
bamfile.close()
```



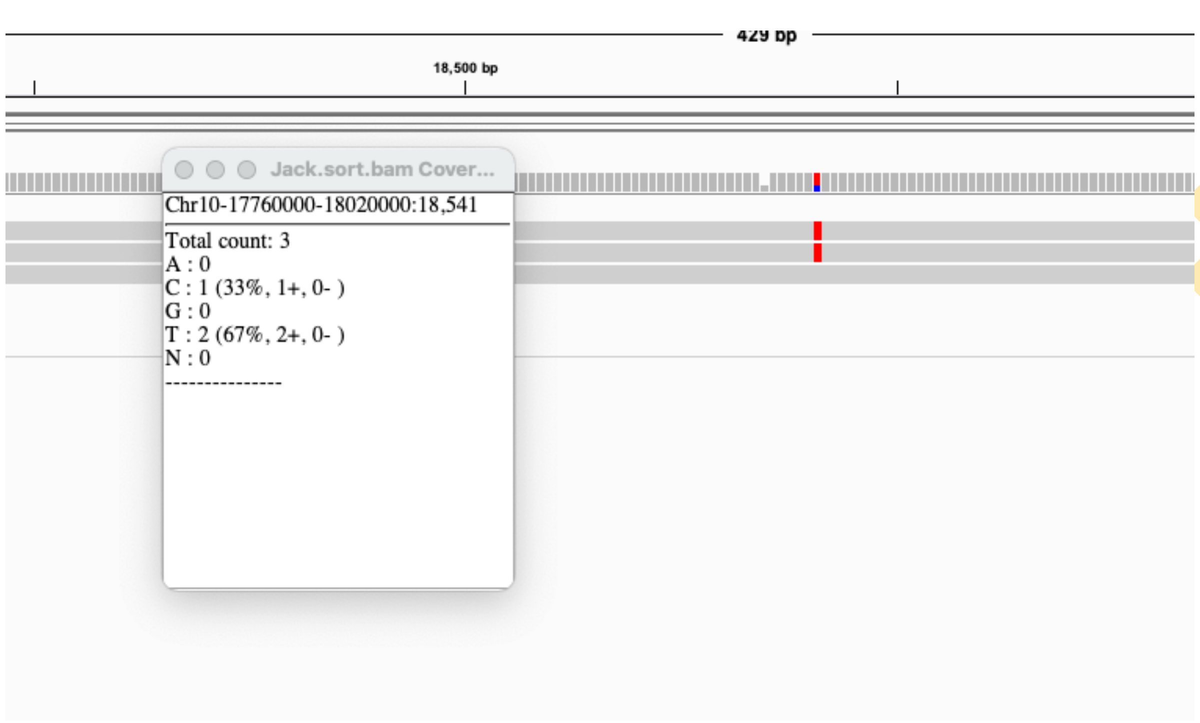


```
print (bamfile.count())
```

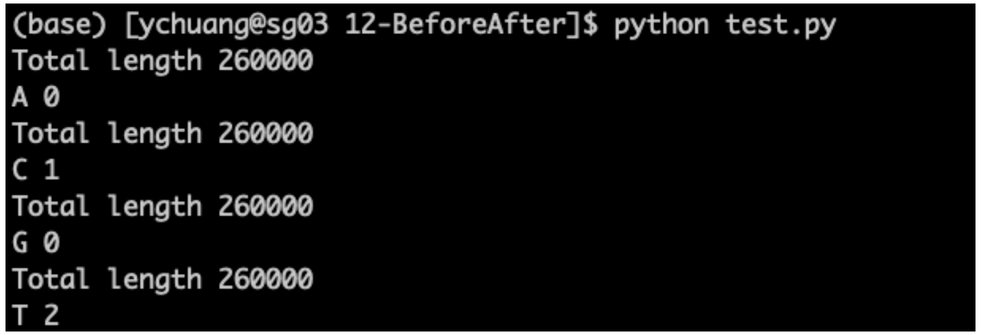


```
(base) [ychuang@sg03 12-BeforeAfter]$ python test.py
3
```

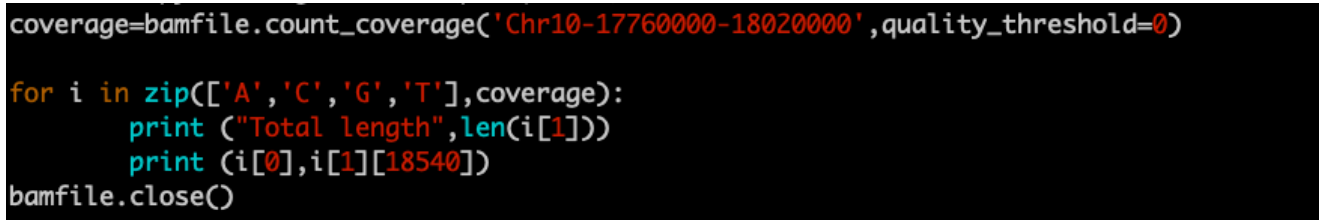
- count_coverage(*self*, *contig*, *start =None*, *stop=None*, *region=None*, *quality_threshold=15*, *read_callback='all'*, *reference=None*, *end=None*)
- **four array.arrays of the same length in order A C G T**



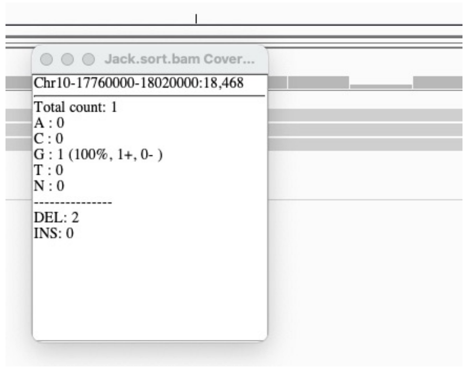
```
Chr10-17760000-18020000:18,541
Total count: 3
A : 0
C : 1 (33%, 1+, 0- )
G : 0
T : 2 (67%, 2+, 0- )
N : 0
---------------
```



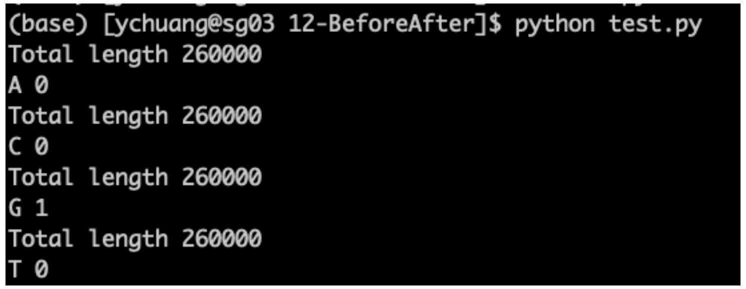
```
(base) [ychuang@sg03 12-BeforeAfter]$ python test.py
Total length 260000
A 0
Total length 260000
C 1
Total length 260000
G 0
Total length 260000
T 2
```



```
coverage=bamfile.count_coverage('Chr10-17760000-18020000',quality_threshold=0)

for i in zip(['A','C','G','T'],coverage):
        print ("Total length",len(i[1]))
        print (i[0],i[1][18540])
bamfile.close()
```

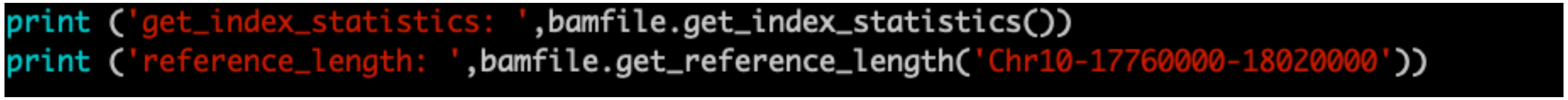


```
Chr10-17760000-18020000:18,468
Total count: 1
A : 0
C : 0
G : 1 (100%, 1+, 0- )
T : 0
N : 0
---------------
DEL: 2
INS: 0
```

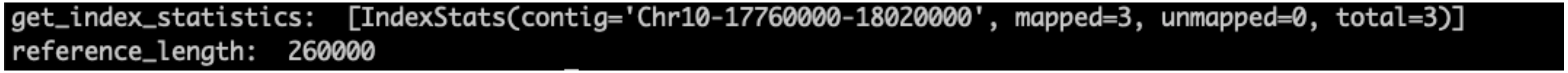


```
(base) [ychuang@sg03 12-BeforeAfter]$ python test.py
Total length 260000
A 0
Total length 260000
C 0
Total length 260000
G 1
Total length 260000
T 0
```

Statistics

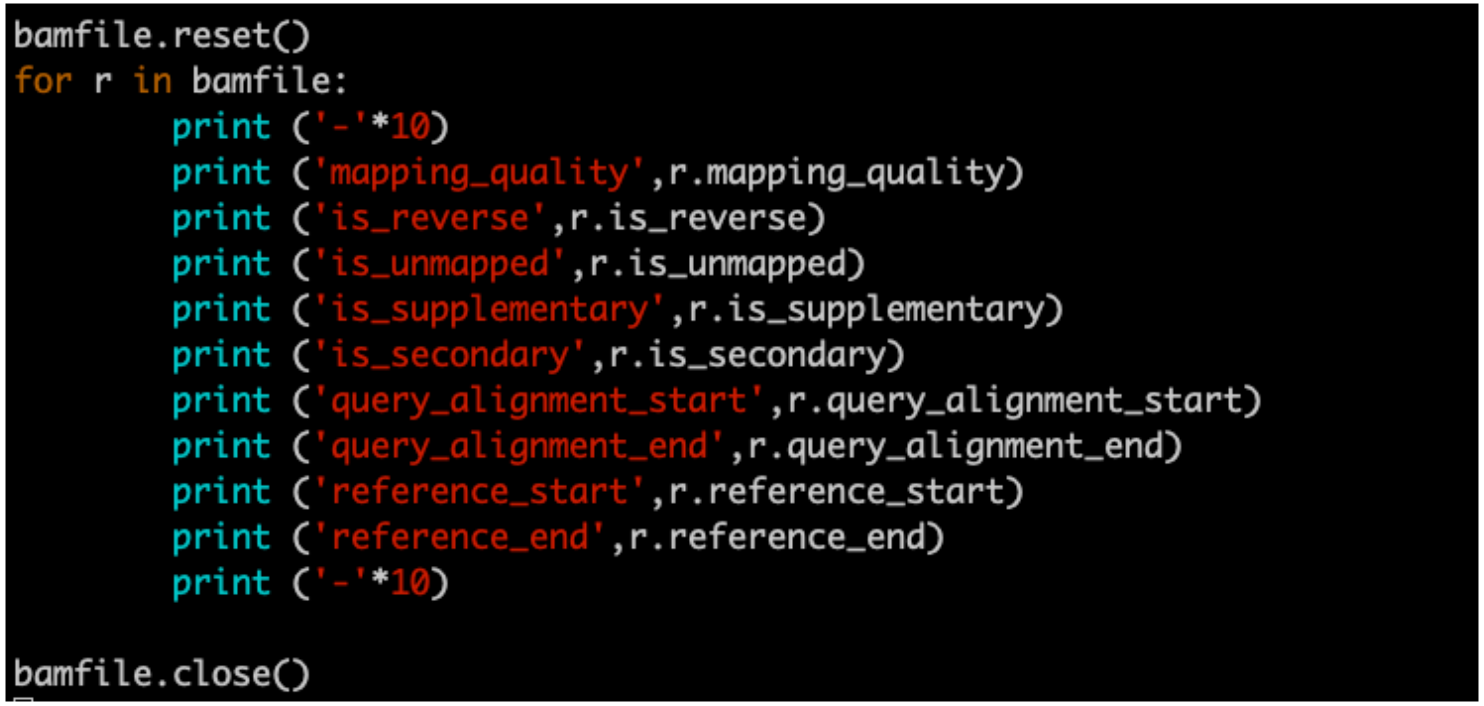


```
print ('get_index_statistics: ',bamfile.get_index_statistics())
print ('reference_length: ',bamfile.get_reference_length('Chr10-17760000-18020000'))
```

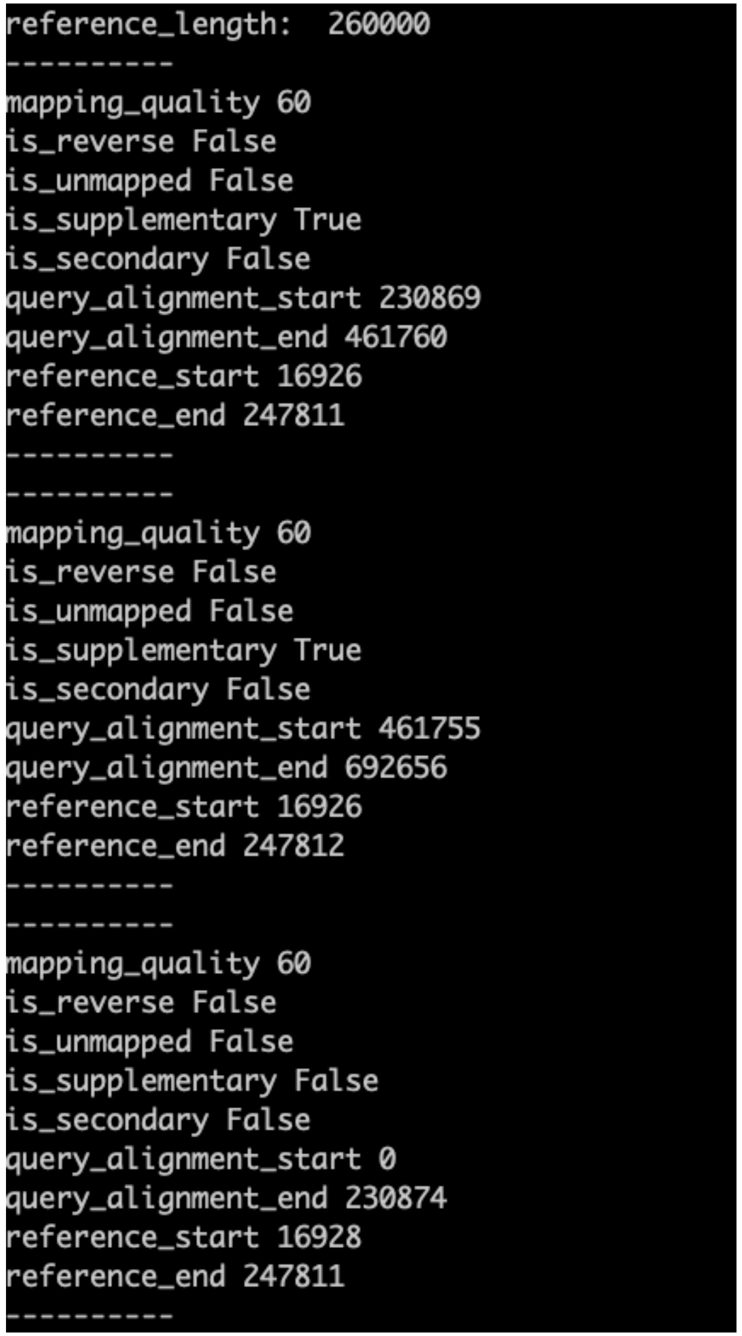


```
get_index_statistics:  [IndexStats(contig='Chr10-17760000-18020000', mapped=3, unmapped=0, total=3)]
reference_length:  260000
```

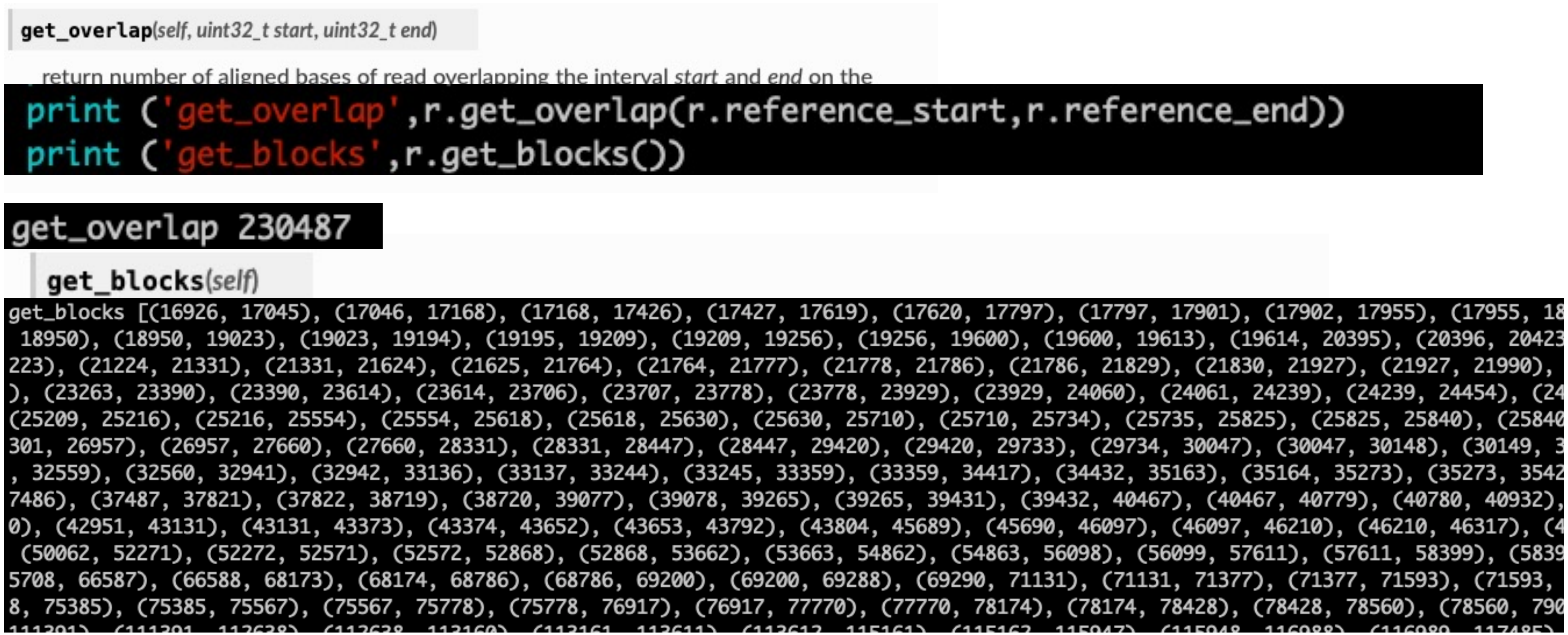
Alignment records



```
bamfile.reset()
for r in bamfile:
        print ('-'*10)
        print ('mapping_quality',r.mapping_quality)
        print ('is_reverse',r.is_reverse)
        print ('is_unmapped',r.is_unmapped)
        print ('is_supplementary',r.is_supplementary)
        print ('is_secondary',r.is_secondary)
        print ('query_alignment_start',r.query_alignment_start)
        print ('query_alignment_end',r.query_alignment_end)
        print ('reference_start',r.reference_start)
        print ('reference_end',r.reference_end)
        print ('-'*10)

bamfile.close()
```



```
reference_length:  260000
----------
mapping_quality 60
is_reverse False
is_unmapped False
is_supplementary True
is_secondary False
query_alignment_start 230869
query_alignment_end 461760
reference_start 16926
reference_end 247811
----------
----------
mapping_quality 60
is_reverse False
is_unmapped False
is_supplementary True
is_secondary False
query_alignment_start 461755
query_alignment_end 692656
reference_start 16926
reference_end 247812
----------
----------
mapping_quality 60
is_reverse False
is_unmapped False
is_supplementary False
is_secondary False
query_alignment_start 0
query_alignment_end 230874
reference_start 16928
reference_end 247811
----------
```



get_overlap(*self, uint32_t start, uint32_t end*)

return number of aligned bases of read overlapping the interval *start* and *end* on the

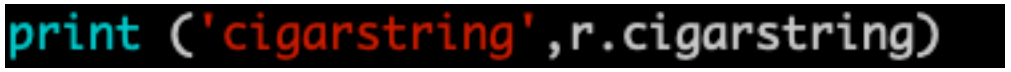
```
print ('get_overlap',r.get_overlap(r.reference_start,r.reference_end))
print ('get_blocks',r.get_blocks())
```

```
get_overlap 230487
```

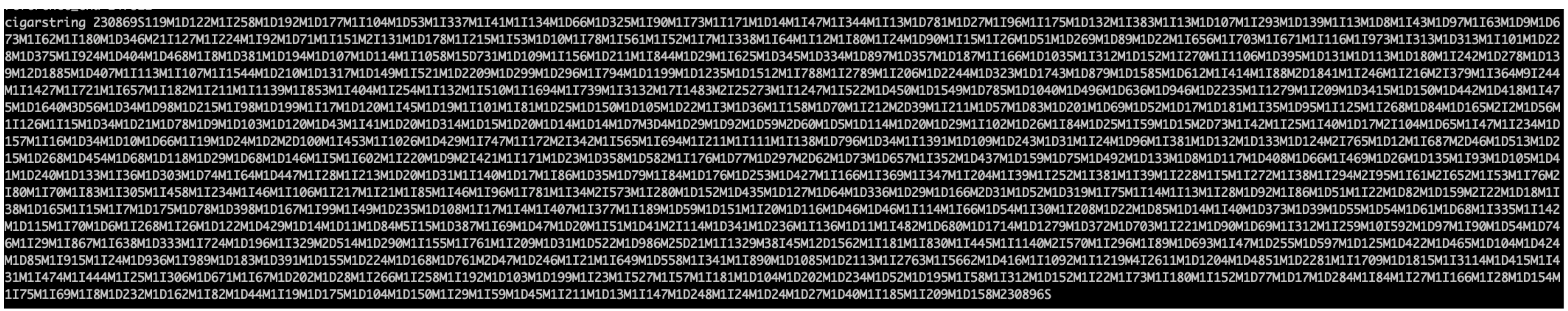
get_blocks(*self*)

```
get_blocks [(16926, 17045), (17046, 17168), (17168, 17426), (17427, 17619), (17620, 17797), (17797, 17901), (17902, 17955), (17955, 18
 18950), (18950, 19023), (19023, 19194), (19195, 19209), (19209, 19256), (19256, 19600), (19600, 19613), (19614, 20395), (20396, 20423
223), (21224, 21331), (21331, 21624), (21625, 21764), (21764, 21777), (21778, 21786), (21786, 21829), (21830, 21927), (21927, 21990),
), (23263, 23390), (23390, 23614), (23614, 23706), (23707, 23778), (23778, 23929), (23929, 24060), (24061, 24239), (24239, 24454), (24
(25209, 25216), (25216, 25554), (25554, 25618), (25618, 25630), (25630, 25710), (25710, 25734), (25735, 25825), (25825, 25840), (25840
301, 26957), (26957, 27660), (27660, 28331), (28331, 28447), (28447, 29420), (29420, 29733), (29734, 30047), (30047, 30148), (30149, 3
, 32559), (32560, 32941), (32942, 33136), (33137, 33244), (33245, 33359), (33359, 34417), (34432, 35163), (35164, 35273), (35273, 3542
7486), (37487, 37821), (37822, 38719), (38720, 39077), (39078, 39265), (39265, 39431), (39432, 40467), (40467, 40779), (40780, 40932),
0), (42951, 43131), (43131, 43373), (43374, 43652), (43653, 43792), (43804, 45689), (45690, 46097), (46097, 46210), (46210, 46317), (4
 (50062, 52271), (52272, 52571), (52572, 52868), (52868, 53662), (53663, 54862), (54863, 56098), (56099, 57611), (57611, 58399), (5839
5708, 66587), (66588, 68173), (68174, 68786), (68786, 69200), (69200, 69288), (69290, 71131), (71131, 71377), (71377, 71593), (71593,
8, 75385), (75385, 75567), (75567, 75778), (75778, 76917), (76917, 77770), (77770, 78174), (78174, 78428), (78428, 78560), (78560, 790
111391), (111391, 112638), (112638, 113160), (113161, 113611), (113612, 115161), (115162, 115947), (115948, 116988), (116989, 117485),
```

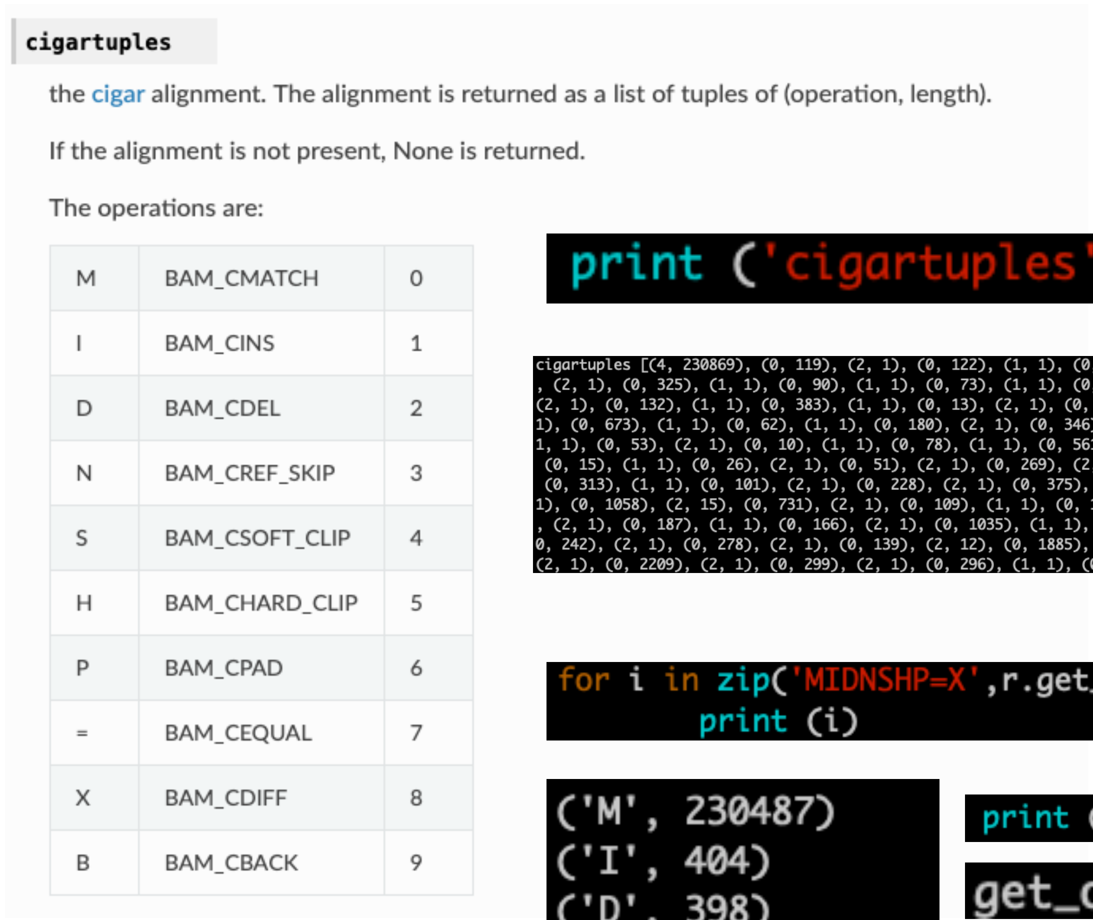
cigarstring



```python
print ('cigarstring',r.cigarstring)
```



```
cigarstring 230869S119M1D122M1I258M1D192M1D177M1I104M1D53M1I337M1I41M1I134M1D66M1D325M1I90M1I73M1I171M1D14M1I47M1I344M1I13M1D781M1D27M1I96M1I175M1D132M1I383M1I13M1D107M1I293M1D139M1I13M1D8M1I43M1D97M1I63M1D9M1D6
73M1I62M1I180M1D346M21I127M1I224M1I92M1D71M1I151M2I131M1D178M1I215M1I53M1D10M1I78M1I561M1I52M1I7M1I338M1I64M1I12M1I80M1I24M1D90M1I15M1I26M1D51M1D269M1D89M1D22M1I656M1I703M1I671M1I116M1I973M1I313M1D313M1I101M1D22
8M1D375M1I924M1D404M1D468M1I8M1D381M1D194M1D107M1D114M1I1058M15D731M1D109M1I156M1D211M1I844M1D29M1I625M1D345M1D334M1D897M1D357M1D187M1I166M1D1035M1I312M1D152M1I270M1I1106M1D395M1D131M1D113M1D180M1I242M1D278M1D13
9M12D1885M1D407M1I113M1I107M1I1544M1D210M1D1317M1D149M1I521M1D2209M1D299M1D296M1I794M1D1199M1D1235M1D1512M1I788M1I2789M1I206M1D2244M1D323M1D1743M1D879M1D1585M1D612M1I414M1I88M2D1841M1I246M1I216M2I379M1I364M9I244
M1I1427M1I721M1I657M1I182M1I211M1I1139M1I853M1I404M1I254M1I132M1I510M1I1694M1I739M1I3132M17I1483M2I25273M1I1247M1I522M1D450M1D1549M1D785M1D1040M1D496M1D636M1D946M1D2235M1I1279M1I209M1D3415M1D150M1D442M1D418M1I47
5M1D1640M3D56M1D34M1D98M1D215M1I98M1D199M1I17M1D120M1I45M1D19M1I101M1I81M1D25M1D150M1D105M1D22M1I3M1D36M1I158M1D70M1I212M2D39M1I211M1D57M1D83M1D201M1D69M1D52M1D17M1D181M1I35M1D95M1I125M1I268M1D84M1D165M2I2M1D56M
1I126M1I15M1D34M1D21M1D78M1D9M1D103M1D120M1D43M1I41M1D20M1D314M1D15M1D20M1D14M1D14M1D7M3D4M1D29M1D92M1D59M2D60M1D5M1D114M1D20M1D29M1I102M1D26M1I84M1D25M1I59M1D15M2D73M1I42M1I25M1I40M1D17M2I104M1D65M1I47M1I234M1D
157M1I16M1D34M1D10M1D66M1I19M1D24M1D2M2D100M1I453M1I1026M1D429M1I747M1I172M2I342M1I565M1I694M1I211M1I111M1I138M1D796M1D34M1I1391M1D109M1D243M1D31M1I24M1D96M1I381M1D132M1D133M1D124M2I765M1D12M1I687M2D46M1D513M1D2
15M1D268M1D454M1D68M1D118M1D29M1D68M1D146M1I5M1I602M1I220M1D9M2I421M1I171M1D23M1D358M1D582M1I176M1D77M1D297M2D62M1D73M1D657M1I352M1D437M1D159M1D75M1D492M1D133M1D8M1D117M1D408M1D66M1I469M1D26M1D135M1I93M1D105M1D4
1M1D240M1D133M1I36M1D303M1D74M1I64M1D447M1I28M1I213M1D20M1D31M1I140M1D17M1I86M1D35M1D79M1I84M1D176M1D253M1D427M1I166M1I369M1I347M1I204M1I39M1I252M1I381M1I39M1I228M1I5M1I272M1I38M1I294M2I95M1I61M2I652M1I53M1I76M2
I80M1I70M1I83M1I305M1I458M1I234M1I46M1I106M1I217M1I21M1I85M1I46M1I96M1I781M1I34M2I573M1I280M1D152M1D435M1D127M1D64M1D336M1D29M1D166M2D31M1D52M1D319M1I75M1I14M1I13M1I28M1D92M1I86M1D51M1I22M1D82M1D159M2I22M1D18M1I
38M1D165M1I15M1I7M1D175M1D78M1D398M1D167M1I99M1I49M1D235M1D108M1I17M1I4M1I407M1I377M1I189M1D59M1D151M1I20M1D116M1D46M1D46M1I114M1I66M1D54M1I30M1I208M1D22M1D85M1D14M1I40M1D373M1D39M1D55M1D54M1D61M1D68M1I335M1I142
M1D115M1I70M1D6M1I268M1I26M1D122M1D429M1D14M1D11M1D84M5I15M1D387M1I69M1D47M1D20M1I51M1D41M2I114M1D341M1D236M1I136M1D11M1I482M1D680M1D1714M1D1279M1D372M1D703M1I221M1D90M1D69M1I312M1I259M10I592M1D97M1I90M1D54M1D74
6M1I29M1I867M1I638M1D333M1I724M1D196M1I329M2D514M1D290M1I155M1I761M1I209M1D31M1D522M1D986M25D21M1I1329M38I45M12D1562M1I181M1I830M1I445M1I1140M2I570M1I296M1I89M1D693M1I47M1D255M1D597M1D125M1D422M1D465M1D104M1D424
M1D85M1I915M1I24M1D936M1I989M1D183M1D391M1D155M1D224M1D168M1D761M2D47M1D246M1I21M1I649M1D558M1I341M1I890M1D1085M1D2113M1I2763M1I5662M1D416M1I1092M1I1219M4I2611M1D1204M1D4851M1D2281M1I1709M1D1815M1I3114M1D415M1I4
31M1I474M1I444M1I25M1I306M1D671M1I67M1D202M1D28M1I266M1I258M1I192M1D103M1D199M1I23M1I527M1I57M1I181M1D104M1D202M1D234M1D52M1D195M1I58M1I312M1D152M1I22M1I73M1I180M1I152M1D77M1D17M1D284M1I84M1I27M1I166M1I28M1D154M
1I75M1I69M1I8M1D232M1D162M1I82M1D44M1I19M1D175M1D104M1D150M1I29M1I59M1D45M1I211M1D13M1I147M1D248M1I24M1D24M1D27M1D40M1I185M1I209M1D158M230896S
```



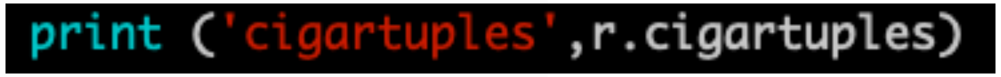
`cigartuples`

the cigar alignment. The alignment is returned as a list of tuples of (operation, length).

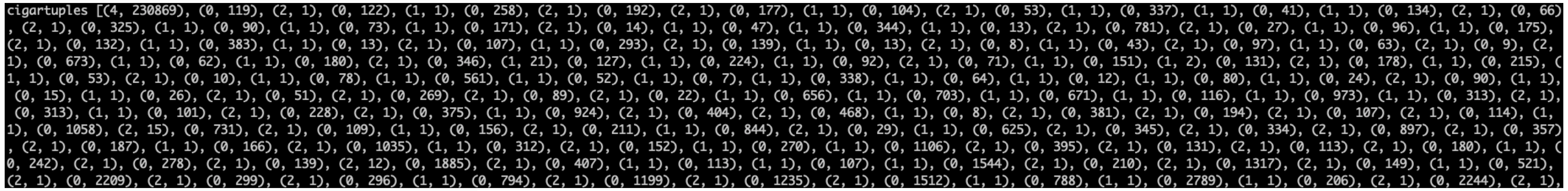
If the alignment is not present, None is returned.

The operations are:

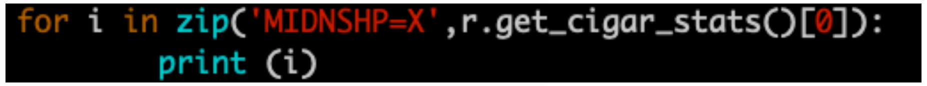
M	BAM_CMATCH	0
I	BAM_CINS	1
D	BAM_CDEL	2
N	BAM_CREF_SKIP	3
S	BAM_CSOFT_CLIP	4
H	BAM_CHARD_CLIP	5
P	BAM_CPAD	6
=	BAM_CEQUAL	7
X	BAM_CDIFF	8
B	BAM_CBACK	9



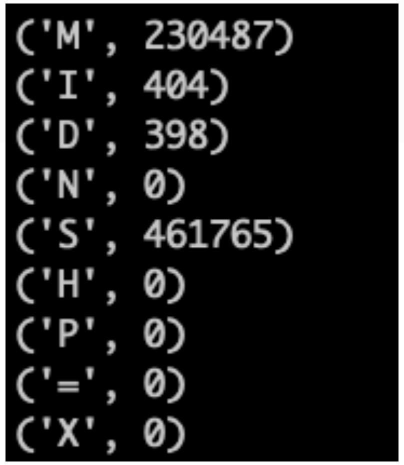
```
print ('cigartuples',r.cigartuples)
```



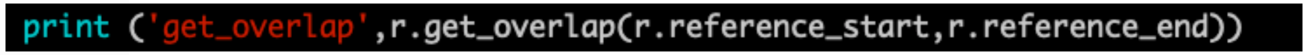
```
cigartuples [(4, 230869), (0, 119), (2, 1), (0, 122), (1, 1), (0, 258), (2, 1), (0, 192), (2, 1), (0, 177), (1, 1), (0, 104), (2, 1), (0, 53), (1, 1), (0, 337), (1, 1), (0, 41), (1, 1), (0, 134), (2, 1), (0, 66)
, (2, 1), (0, 325), (1, 1), (0, 90), (1, 1), (0, 73), (1, 1), (0, 171), (2, 1), (0, 14), (1, 1), (0, 47), (1, 1), (0, 344), (1, 1), (0, 13), (2, 1), (0, 781), (2, 1), (0, 27), (1, 1), (0, 96), (1, 1), (0, 175),
(2, 1), (0, 132), (1, 1), (0, 383), (1, 1), (0, 13), (2, 1), (0, 107), (1, 1), (0, 293), (2, 1), (0, 139), (1, 1), (0, 13), (2, 1), (0, 8), (1, 1), (0, 43), (2, 1), (0, 97), (1, 1), (0, 63), (2, 1), (0, 9), (2,
1), (0, 673), (1, 1), (0, 62), (1, 1), (0, 180), (2, 1), (0, 346), (1, 21), (0, 127), (1, 1), (0, 224), (1, 1), (0, 92), (2, 1), (0, 71), (1, 1), (0, 151), (1, 2), (0, 131), (2, 1), (0, 178), (1, 1), (0, 215), (
1, 1), (0, 53), (2, 1), (0, 10), (1, 1), (0, 78), (1, 1), (0, 561), (1, 1), (0, 52), (1, 1), (0, 7), (1, 1), (0, 338), (1, 1), (0, 64), (1, 1), (0, 12), (1, 1), (0, 80), (1, 1), (0, 24), (2, 1), (0, 90), (1, 1),
 (0, 15), (1, 1), (0, 26), (2, 1), (0, 51), (2, 1), (0, 269), (2, 1), (0, 89), (2, 1), (0, 22), (1, 1), (0, 656), (1, 1), (0, 703), (1, 1), (0, 671), (1, 1), (0, 116), (1, 1), (0, 973), (1, 1), (0, 313), (2, 1),
 (0, 313), (1, 1), (0, 101), (2, 1), (0, 228), (2, 1), (0, 375), (1, 1), (0, 924), (2, 1), (0, 404), (2, 1), (0, 468), (1, 1), (0, 8), (2, 1), (0, 381), (2, 1), (0, 194), (2, 1), (0, 107), (2, 1), (0, 114), (1,
1), (0, 1058), (2, 15), (0, 731), (2, 1), (0, 109), (1, 1), (0, 156), (2, 1), (0, 211), (1, 1), (0, 844), (2, 1), (0, 29), (1, 1), (0, 625), (2, 1), (0, 345), (2, 1), (0, 334), (2, 1), (0, 897), (2, 1), (0, 357)
, (2, 1), (0, 187), (1, 1), (0, 166), (2, 1), (0, 1035), (1, 1), (0, 312), (2, 1), (0, 152), (1, 1), (0, 270), (1, 1), (0, 1106), (2, 1), (0, 395), (2, 1), (0, 131), (2, 1), (0, 113), (2, 1), (0, 180), (1, 1), (
0, 242), (2, 1), (0, 278), (2, 1), (0, 139), (2, 12), (0, 1885), (2, 1), (0, 407), (1, 1), (0, 113), (1, 1), (0, 107), (1, 1), (0, 1544), (2, 1), (0, 210), (2, 1), (0, 1317), (2, 1), (0, 149), (1, 1), (0, 521),
(2, 1), (0, 2209), (2, 1), (0, 299), (2, 1), (0, 296), (1, 1), (0, 794), (2, 1), (0, 1199), (2, 1), (0, 1235), (2, 1), (0, 1512), (1, 1), (0, 788), (1, 1), (0, 2789), (1, 1), (0, 206), (2, 1), (0, 2244), (2, 1),
```



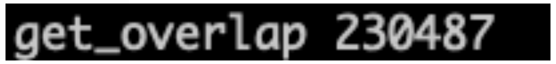
```
for i in zip('MIDNSHP=X',r.get_cigar_stats()[0]):
        print (i)
```



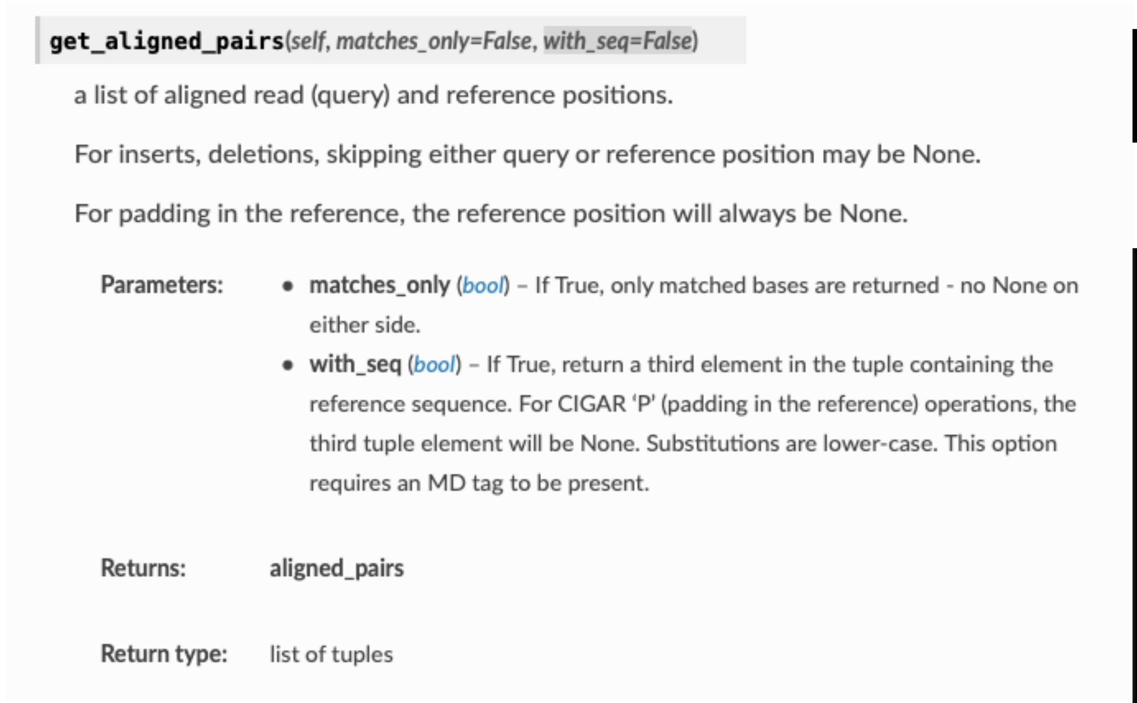
```
('M', 230487)
('I', 404)
('D', 398)
('N', 0)
('S', 461765)
('H', 0)
('P', 0)
('=', 0)
('X', 0)
```



```
print ('get_overlap',r.get_overlap(r.reference_start,r.reference_end))
```



```
get_overlap 230487
```



get_aligned_pairs(*self*, *matches_only=False*, *with_seq=False*)

a list of aligned read (query) and reference positions.

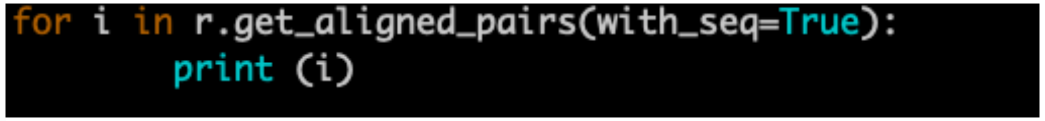
For inserts, deletions, skipping either query or reference position may be None.

For padding in the reference, the reference position will always be None.

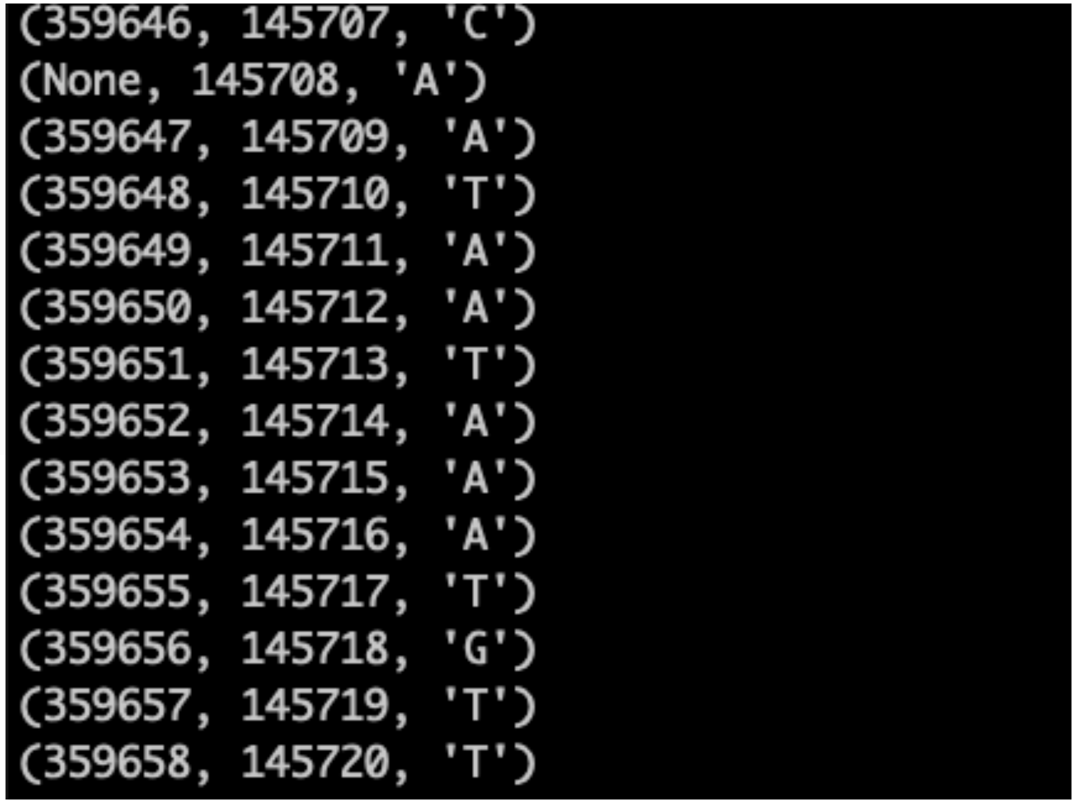
Parameters:
- **matches_only** (*bool*) – If True, only matched bases are returned - no None on either side.
- **with_seq** (*bool*) – If True, return a third element in the tuple containing the reference sequence. For CIGAR 'P' (padding in the reference) operations, the third tuple element will be None. Substitutions are lower-case. This option requires an MD tag to be present.

Returns: aligned_pairs

Return type: list of tuples

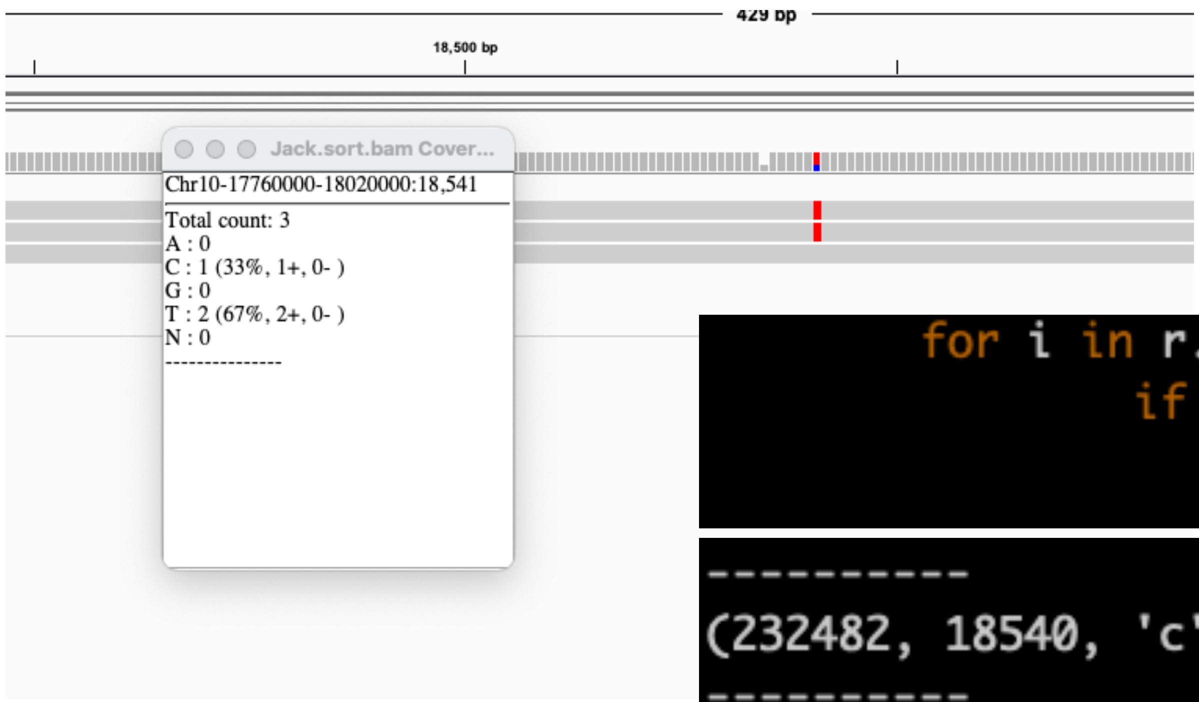


```
for i in r.get_aligned_pairs(with_seq=True):
        print (i)
```

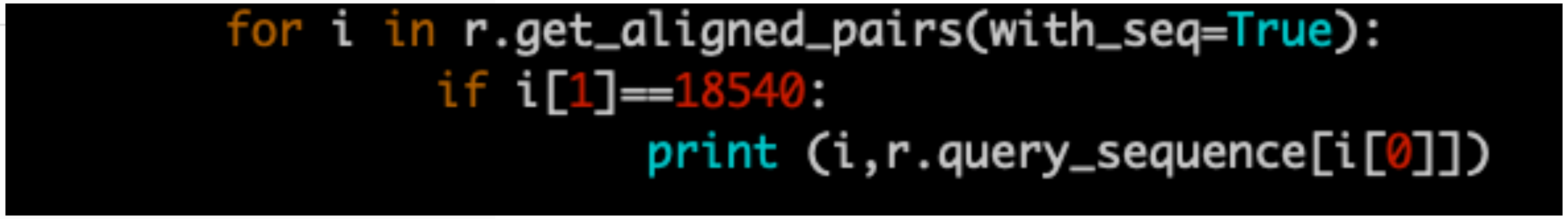


```
(359646, 145707, 'C')
(None, 145708, 'A')
(359647, 145709, 'A')
(359648, 145710, 'T')
(359649, 145711, 'A')
(359650, 145712, 'A')
(359651, 145713, 'T')
(359652, 145714, 'A')
(359653, 145715, 'A')
(359654, 145716, 'A')
(359655, 145717, 'T')
(359656, 145718, 'G')
(359657, 145719, 'T')
(359658, 145720, 'T')
```

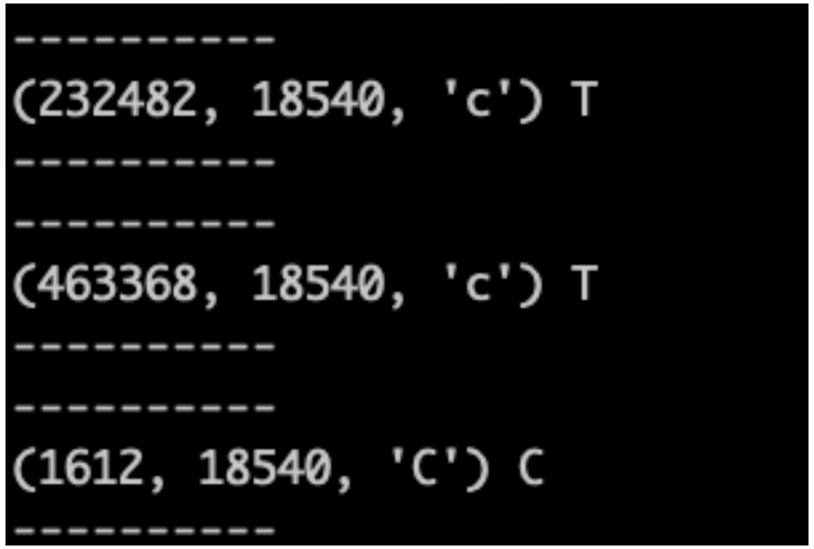
Reads, Reference, Reference base



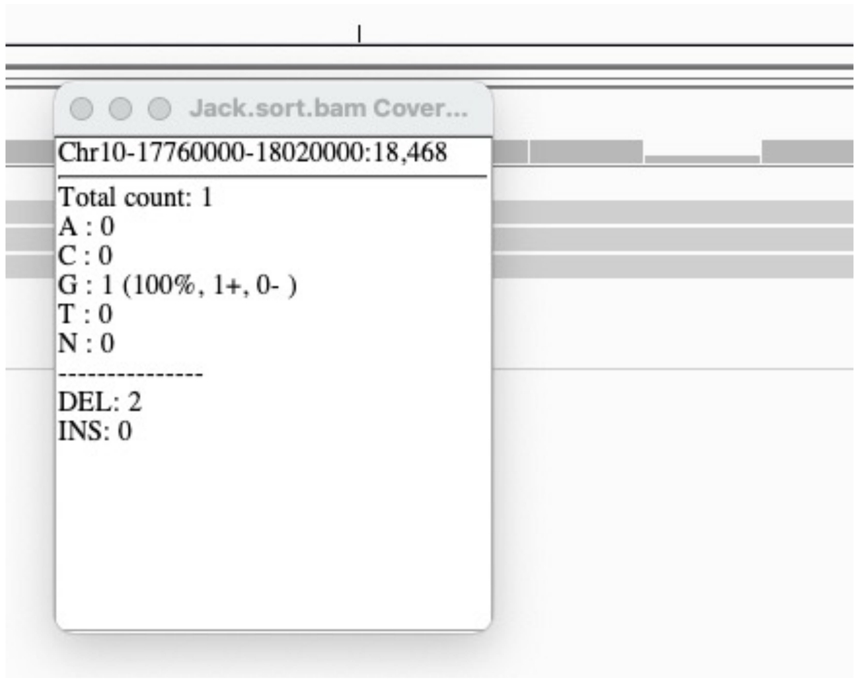
```
Chr10-17760000-18020000:18,541
Total count: 3
A : 0
C : 1 (33%, 1+, 0- )
G : 0
T : 2 (67%, 2+, 0- )
N : 0
---------------
```



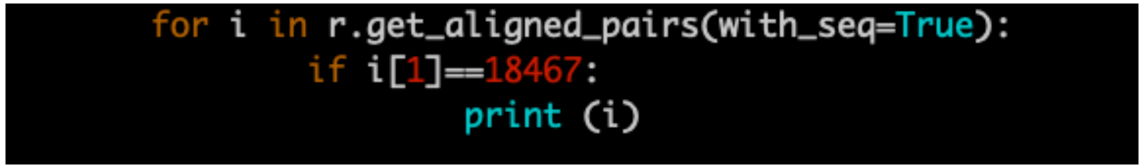
```
for i in r.get_aligned_pairs(with_seq=True):
    if i[1]==18540:
        print (i,r.query_sequence[i[0]])
```



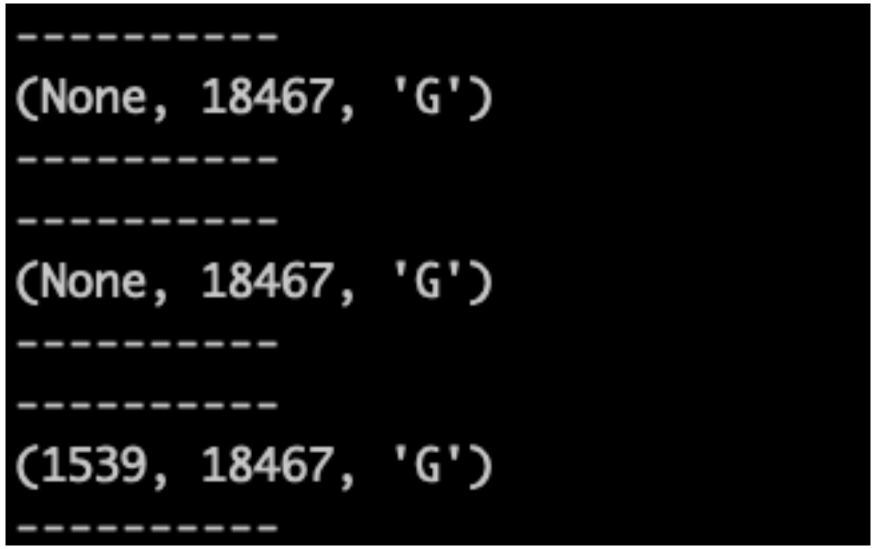
```
----------
(232482, 18540, 'c') T
----------
----------
(463368, 18540, 'c') T
----------
----------
(1612, 18540, 'C') C
----------
```



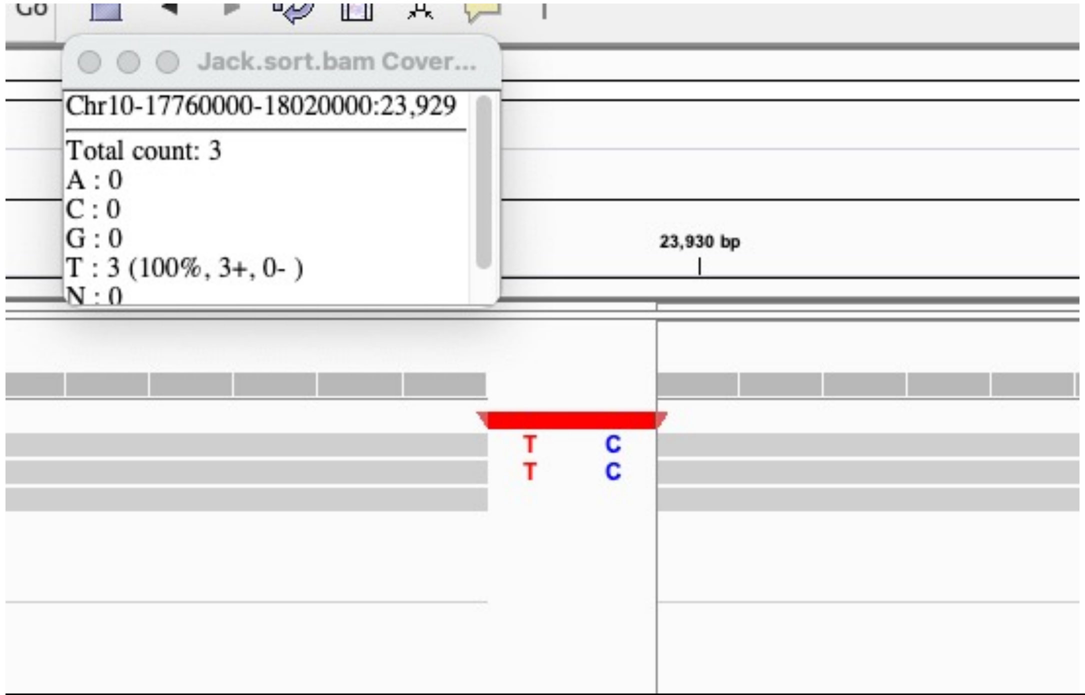
```
Jack.sort.bam Cover...
Chr10-17760000-18020000:18,468
Total count: 1
A : 0
C : 0
G : 1 (100%, 1+, 0- )
T : 0
N : 0
---------------
DEL: 2
INS: 0
```



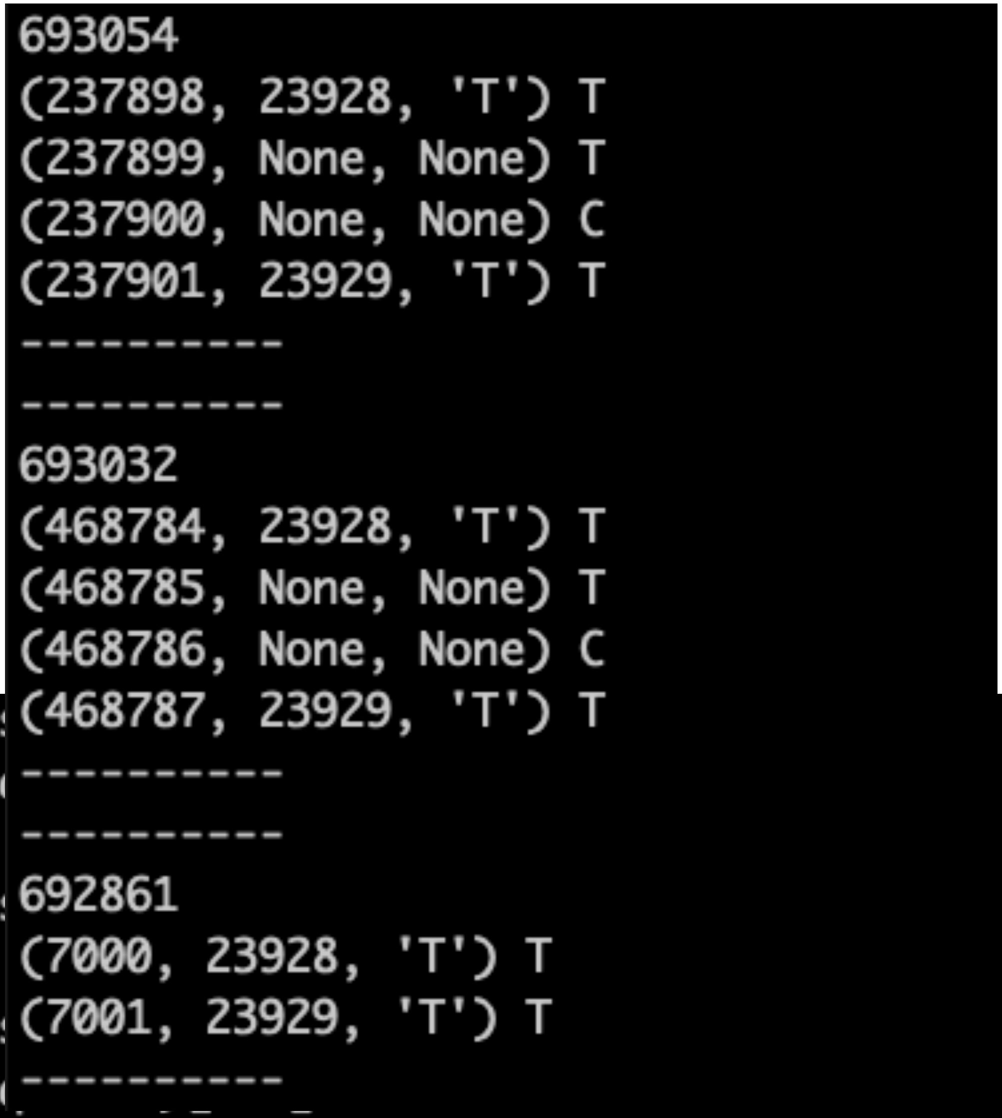
```
for i in r.get_aligned_pairs(with_seq=True):
        if i[1]==18467:
                print (i)
```



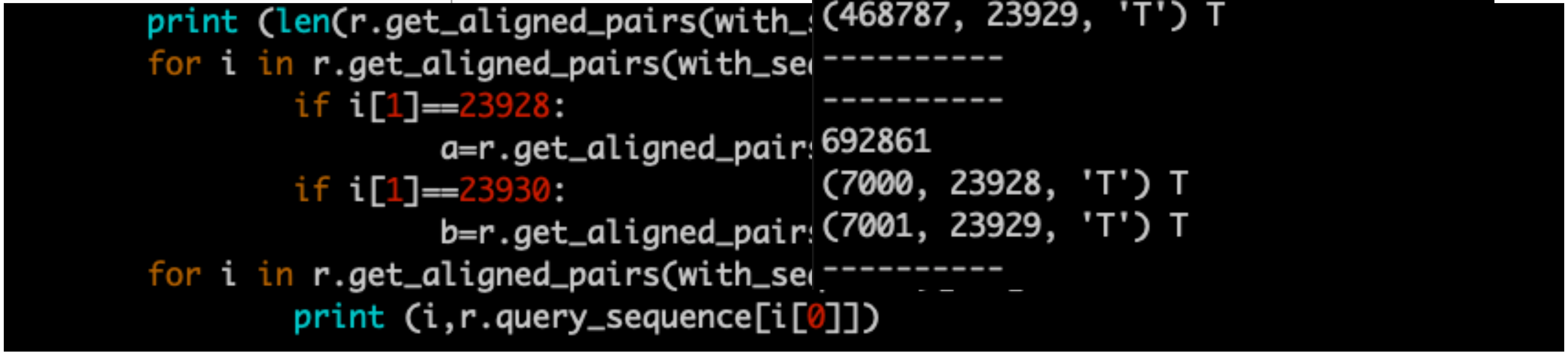
```
----------
(None, 18467, 'G')
----------
----------
(None, 18467, 'G')
----------
----------
(1539, 18467, 'G')
----------
```



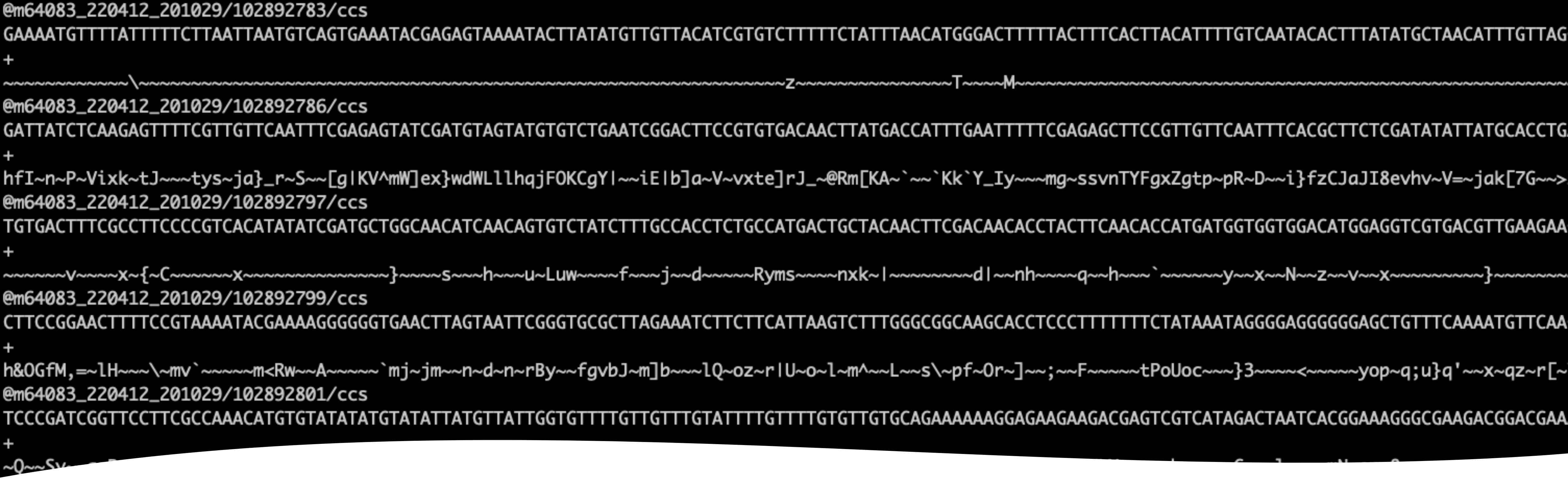
```
Chr10-17760000-18020000:23,929

Total count: 3
A : 0
C : 0
G : 0
T : 3 (100%, 3+, 0- )
N : 0
```



```
693054
(237898, 23928, 'T') T
(237899, None, None) T
(237900, None, None) C
(237901, 23929, 'T') T
----------
----------
693032
(468784, 23928, 'T') T
(468785, None, None) T
(468786, None, None) C
(468787, 23929, 'T') T
----------
----------
692861
(7000, 23928, 'T') T
(7001, 23929, 'T') T
----------
```



```
    print (len(r.get_aligned_pairs(with_
    for i in r.get_aligned_pairs(with_se
            if i[1]==23928:
                    a=r.get_aligned_pair
            if i[1]==23930:
                    b=r.get_aligned_pair
    for i in r.get_aligned_pairs(with_se
            print (i,r.query_sequence[i[0]])
```



```
@m64083_220412_201029/102892783/ccs
GAAAATGTTTTATTTTTCTTAATTAATGTCAGTGAAATACGAGAGTAAAATACTTATATGTTGTTACATCGTGTCTTTTTCTATTTAACATGGGACTTTTTACTTTCACTTACATTTTGTCAATACACTTTATATGCTAACATTTGTTAG
+
~~~~~~~~~~~~\~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~z~~~~~~~~~~~~~~~T~~~~M~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
@m64083_220412_201029/102892786/ccs
GATTATCTCAAGAGTTTTCGTTGTTCAATTTCGAGAGTATCGATGTAGTATGTGTCTGAATCGGACTTCCGTGTGACAACTTATGACCATTTGAATTTTTCGAGAGCTTCCGTTGTTCAATTTCACGCTTCTCGATATATTATGCACCTG
+
hfI~n~P~Vixk~tJ~~~tys~ja}_r~S~~[g|KV^mW]ex}wdWLllhqjFOKCgY|~~iE|b]a~V~vxte]rJ_~@Rm[KA~`~~`Kk`Y_Iy~~~mg~ssvnTYFgxZgtp~pR~D~~i}fzCJaJI8evhv~V=~jak[7G~~>
@m64083_220412_201029/102892797/ccs
TGTGACTTTCGCCTTCCCCGTCACATATATCGATGCTGGCAACATCAACAGTGTCTATCTTTGCCACCTCTGCCATGACTGCTACAACTTCGACAACACCTACTTCAACACCATGATGGTGGTGGACATGGAGGTCGTGACGTTGAAGAA
+
~~~~~~v~~~~x~{~C~~~~~~x~~~~~~~~~~~~~~}~~~~s~~~h~~~u~Luw~~~~f~~~j~~d~~~~~Ryms~~~~nxk~|~~~~~~~~d|~~nh~~~~q~~h~~~`~~~~~~y~~x~~N~~z~~v~~x~~~~~~~~~}~~~~~~
@m64083_220412_201029/102892799/ccs
CTTCCGGAACTTTTCCGTAAAATACGAAAAGGGGGTGAACTTAGTAATTCGGGTGCGCTTAGAAATCTTCTTCATTAAGTCTTTGGGCGGCAAGCACCTCCCTTTTTTTCTATAAATAGGGGAGGGGGGAGCTGTTTCAAAATGTTCAA
+
h&OGfM,=~lH~~~\~mv`~~~~~m<Rw~~A~~~~~`mj~jm~~n~d~n~rBy~~fgvbJ~m]b~~~lQ~oz~r|U~o~l~m^~~L~~s\~pf~Or~]~~;~~F~~~~~tPoUoc~~~}3~~~~<~~~~~yop~q;u}q'~~x~qz~r[~
@m64083_220412_201029/102892801/ccs
TCCCGATCGGTTCCTTCGCCAAACATGTGTATATATGTATATTATGTTATTGGTGTTTTGTTGTTTGTATTTTGTTTTGTGTTGTGCAGAAAAAAGGAGAAGAAGACGAGTCGTCATAGACTAATCACGGAAAGGGCGAAGACGGACGAA
+
~Q~~Sv
```

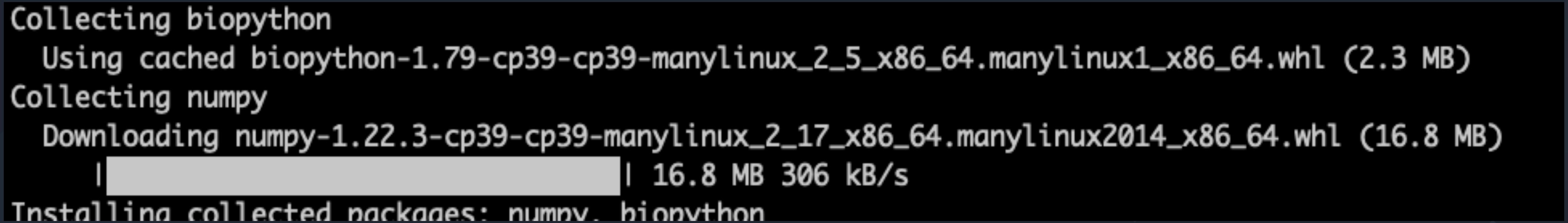
Fasta & Fastq & GFF …

biopython

- The Biopython Project is an international association of developers of freely available Python (https://www.python.org) tools for computational molecular biology. Python is an object oriented, interpreted, flexible language that is becoming increasingly popular for scientific computing. Python is easy to learn, has a very clear syntax and can easily be extended with modules written in C, C++ or FORTRAN.
- http://biopython.org/DIST/docs/tutorial/Tutorial.html

Biopython package

- Blast output – both from standalone and WWW Blast
- Clustalw
- FASTA
- GenBank
- PubMed and Medline
- ExPASy files, like Enzyme and Prosite
- SCOP, including ‘dom’ and ‘lin’ files
- UniGene
- SwissProt



```
Collecting biopython
  Using cached biopython-1.79-cp39-cp39-manylinux_2_5_x86_64.manylinux1_x86_64.whl (2.3 MB)
Collecting numpy
  Downloading numpy-1.22.3-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (16.8 MB)
     |████████████████████████████████| 16.8 MB 306 kB/s
Installing collected packages: numpy, biopython
```

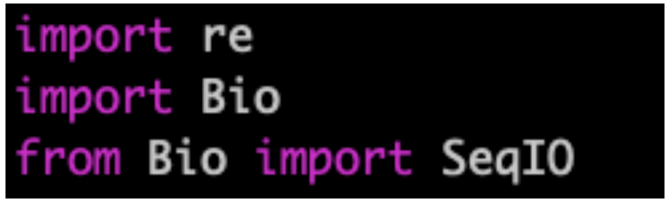
Install

pip install biopython

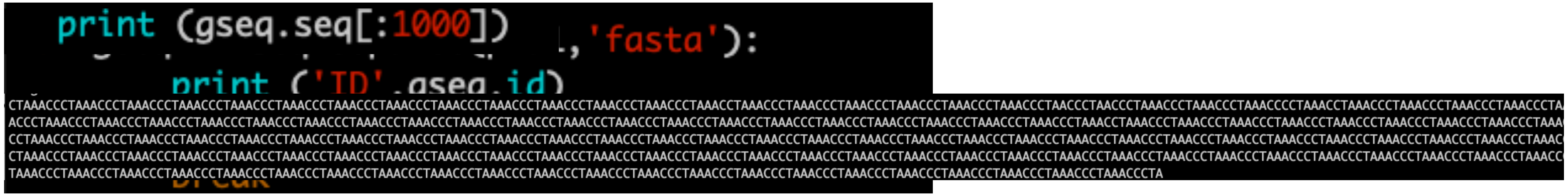
SeqIO Class

ABI sequence format	ACEDB sequence format	AGAVE sequence format	ALF sequence format	ASCIITree sequence format	BSML sequence format
CHADO sequence format	CHAOS sequence format	CTF sequence format	EMBL sequence format	EXP sequence format	EntrezGene sequence format
Excel sequence format	FASTA sequence format	FASTQ sequence format	GAME sequence format	GCG sequence format	GFF3 sequence format
GFF sequence format	GTF sequence format	GenBank sequence format	InterPro sequence format	KEGG sequence format	LocusLink sequence format
MetaFASTA sequence format	PHD sequence format	PIR sequence format	PLN sequence format	Qual sequence format	Raw sequence format
SCF sequence format	Swissprot sequence format	TIGR sequence format	Tab sequence format	Table sequence format	Tinyseq sequence format

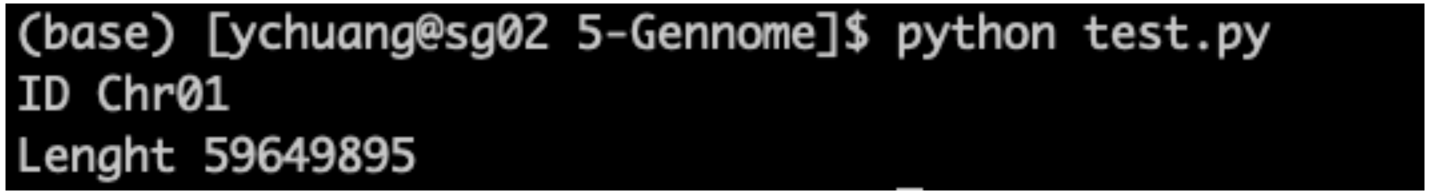
ZFF sequence format



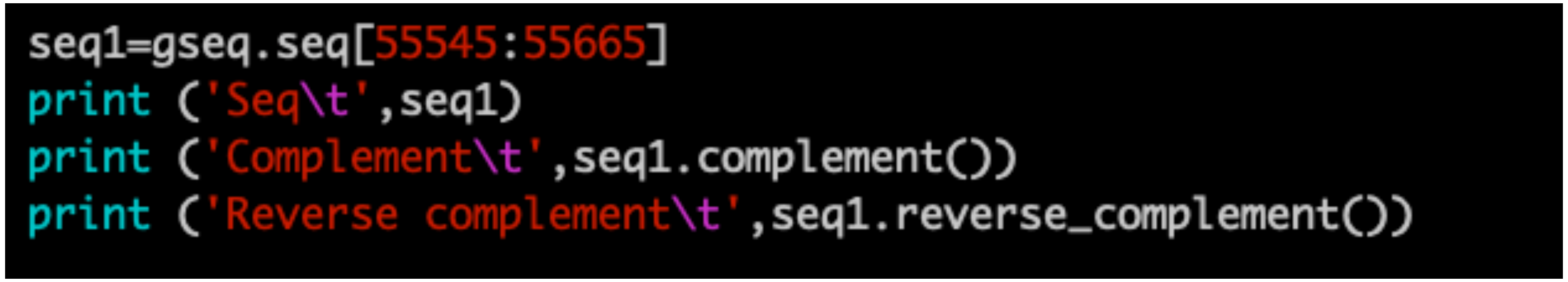
```
import re
import Bio
from Bio import SeqIO
```



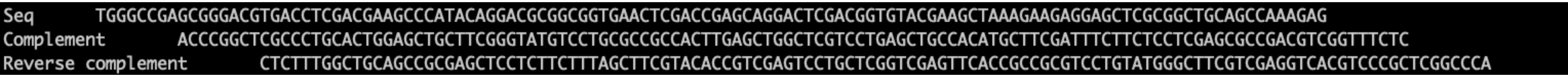
```
  print (gseq.seq[:1000])      ,'fasta'):
          print ('ID',gseq.id)
CTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAACCCTAACCCTAAACCCTAAACCCTAAACCCCTAAACCTAAACCCTAAACCCTAAACCCTAAACCCTA
ACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAA
CCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAAC
CTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACC
TAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTA
```



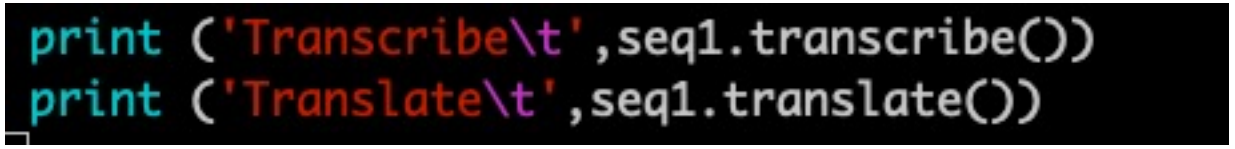
```
(base) [ychuang@sg02 5-Gennome]$ python test.py
ID Chr01
Lenght 59649895
```



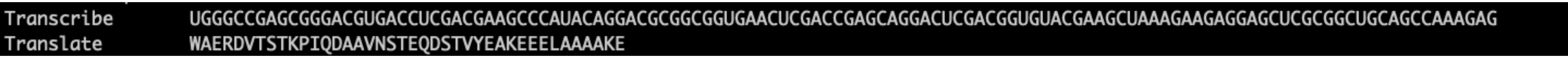
```python
seq1=gseq.seq[55545:55665]
print ('Seq\t',seq1)
print ('Complement\t',seq1.complement())
print ('Reverse complement\t',seq1.reverse_complement())
```



```
Seq	TGGGCCGAGCGGGACGTGACCTCGACGAAGCCCATACAGGACGCGGCGGTGAACTCGACCGAGCAGGACTCGACGGTGTACGAAGCTAAAGAAGAGGAGCTCGCGGCTGCAGCCAAAGAG
Complement	ACCCGGCTCGCCCTGCACTGGAGCTGCTTCGGGTATGTCCTGCGCCGCCACTTGAGCTGGCTCGTCCTGAGCTGCCACATGCTTCGATTTCTTCTCCTCGAGCGCCGACGTCGGTTTCTC
Reverse complement	CTCTTTGGCTGCAGCCGCGAGCTCCTCTTCTTTAGCTTCGTACACCGTCGAGTCCTGCTCGGTCGAGTTCACCGCCGCGTCCTGTATGGGCTTCGTCGAGGTCACGTCCCGCTCGGCCCA
```



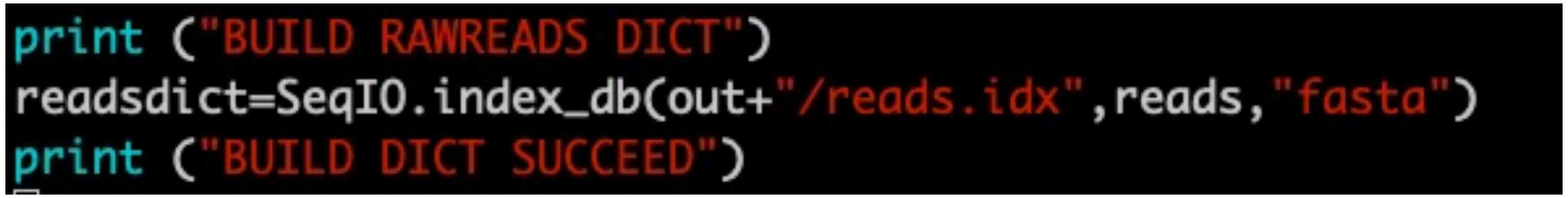
```python
print ('Transcribe\t',seq1.transcribe())
print ('Translate\t',seq1.translate())
```



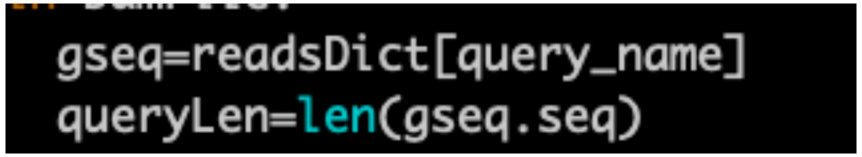
```
Transcribe	UGGGCCGAGCGGGACGUGACCUCGACGAAGCCCAUACAGGACGCGGCGGUGAACUCGACCGAGCAGGACUCGACGGUGUACGAAGCUAAAGAAGAGGAGCUCGCGGCUGCAGCCAAAGAG
Translate	WAERDVTSTKPIQDAAVNSTEQDSTVYEAKEEELAAAKE
```

Sequnce file as Dictionaries

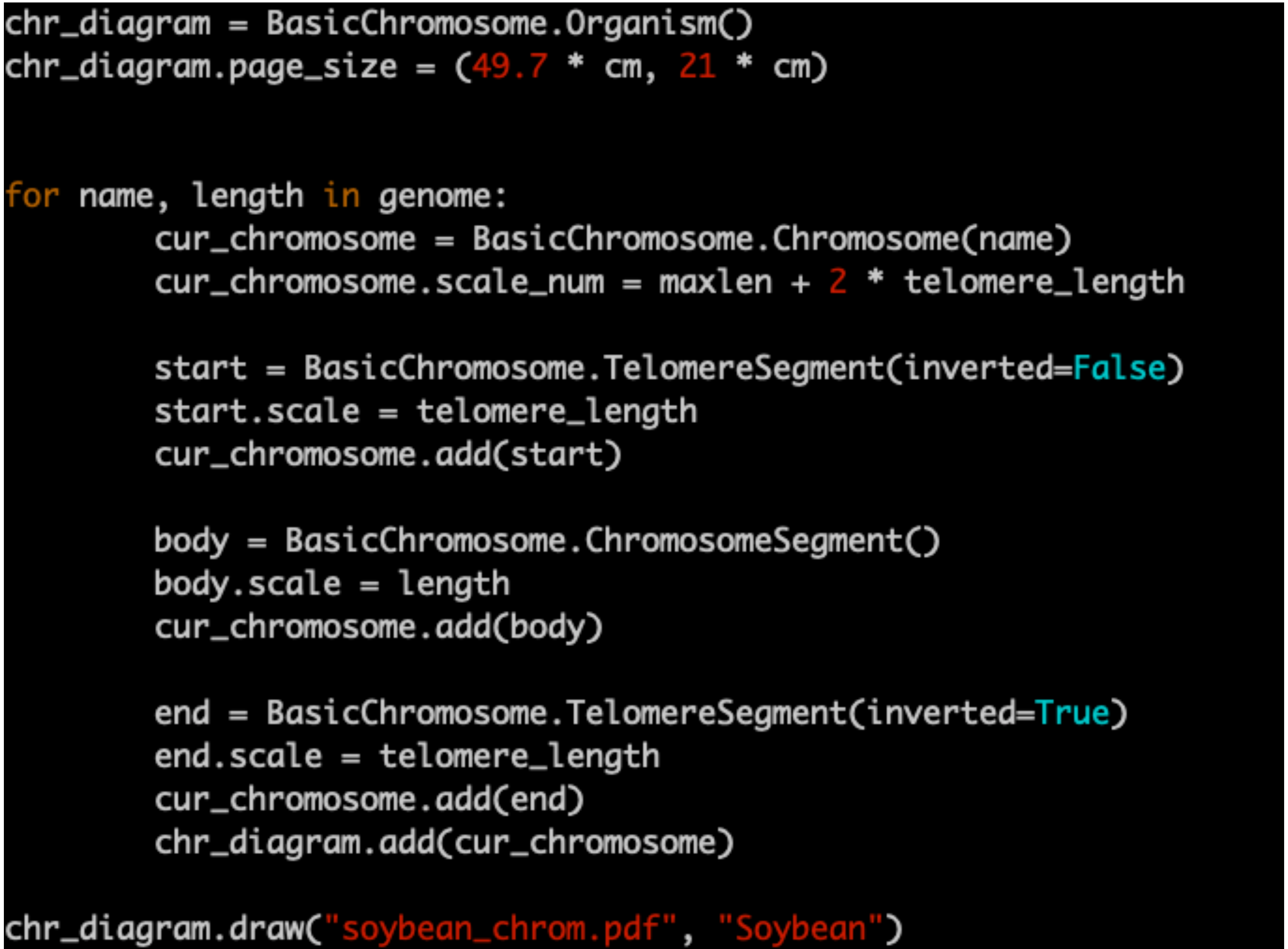
- dict1=SeqIO.to_dict(SeqIO.parse(reads.idx,readspwd,"fasta"))
- dict1 = SeqIO.index(reads.idx,readspwd,"fasta")



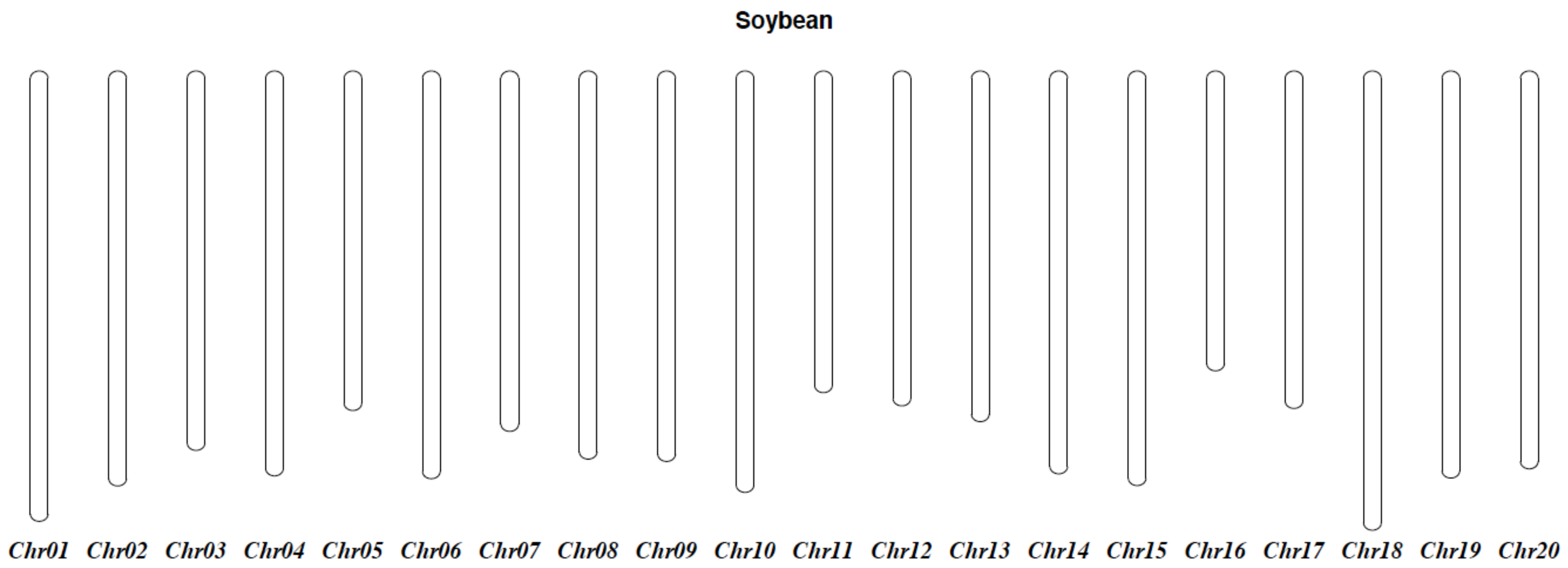
```
print ("BUILD RAWREADS DICT")
readsdict=SeqIO.index_db(out+"/reads.idx",reads,"fasta")
print ("BUILD DICT SUCCEED")
```



```
  gseq=readsDict[query_name]
  queryLen=len(gseq.seq)
```



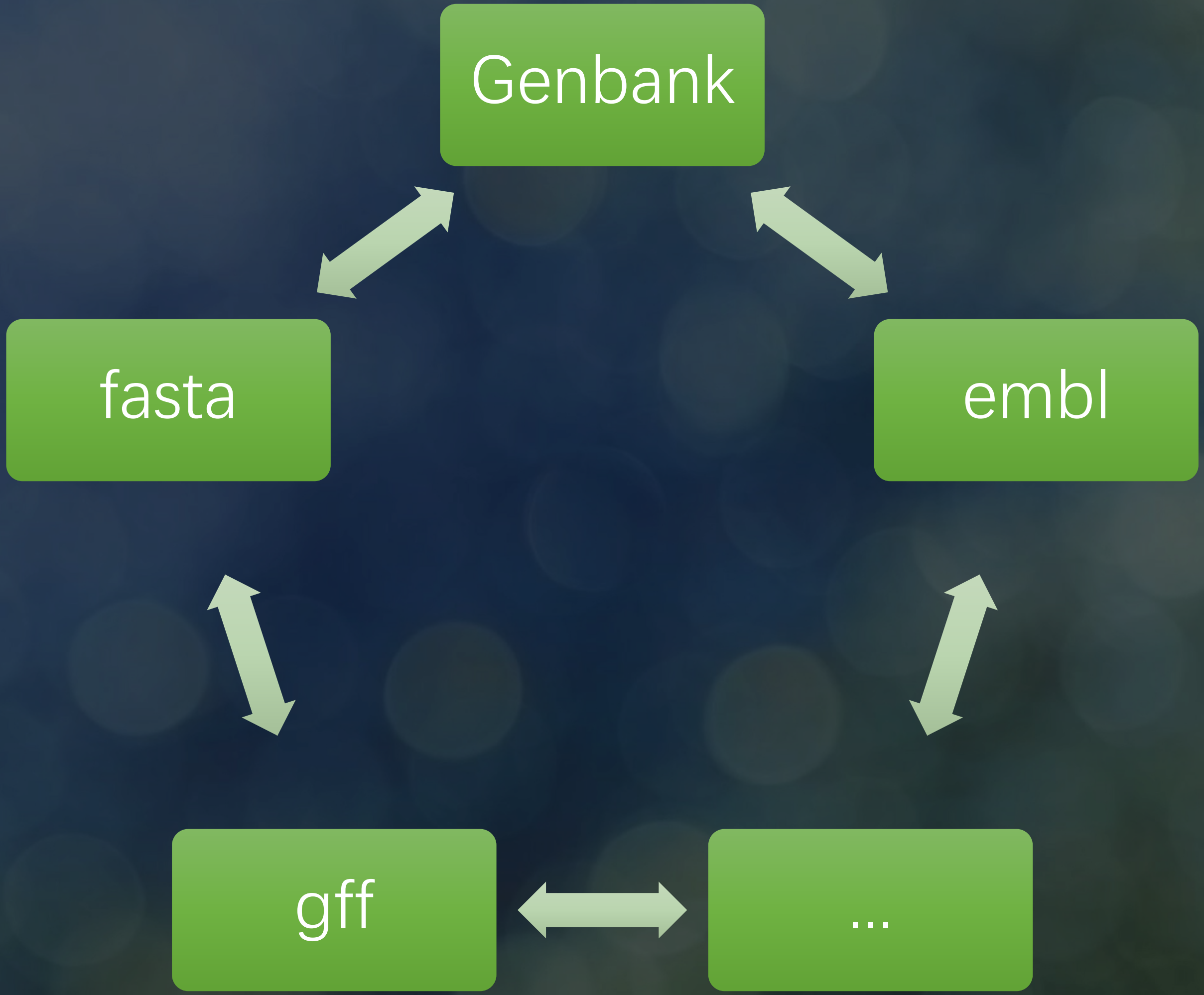
```
chr_diagram = BasicChromosome.Organism()
chr_diagram.page_size = (49.7 * cm, 21 * cm)


for name, length in genome:
        cur_chromosome = BasicChromosome.Chromosome(name)
        cur_chromosome.scale_num = maxlen + 2 * telomere_length

        start = BasicChromosome.TelomereSegment(inverted=False)
        start.scale = telomere_length
        cur_chromosome.add(start)

        body = BasicChromosome.ChromosomeSegment()
        body.scale = length
        cur_chromosome.add(body)

        end = BasicChromosome.TelomereSegment(inverted=True)
        end.scale = telomere_length
        cur_chromosome.add(end)
        chr_diagram.add(cur_chromosome)

chr_diagram.draw("soybean_chrom.pdf", "Soybean")
```


Soybean
Chr01
Chr02
Chr03
Chr04
Chr05
Chr06
Chr07
Chr08
Chr09
Chr10
Chr11
Chr12
Chr13
Chr14
Chr15
Chr16
Chr17
Chr18
Chr19
Chr20

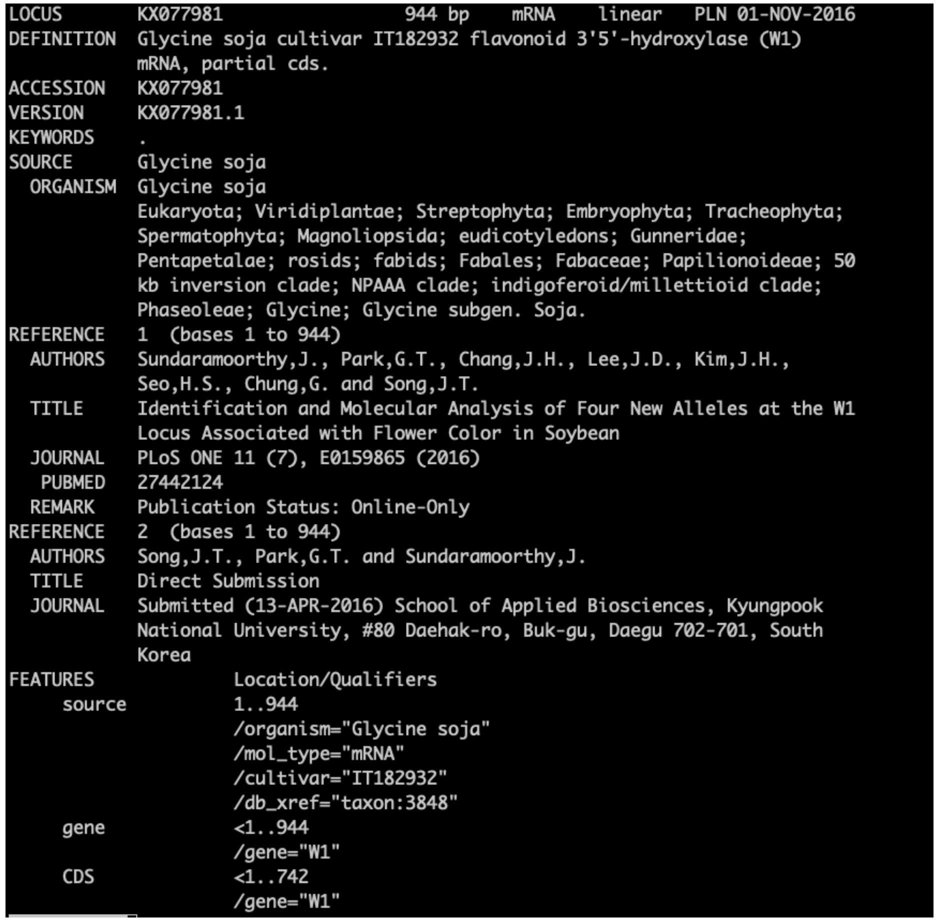
Annotation Files

SeqRecord class

- id
- name
- description
- dxrefs
- annotations
- features

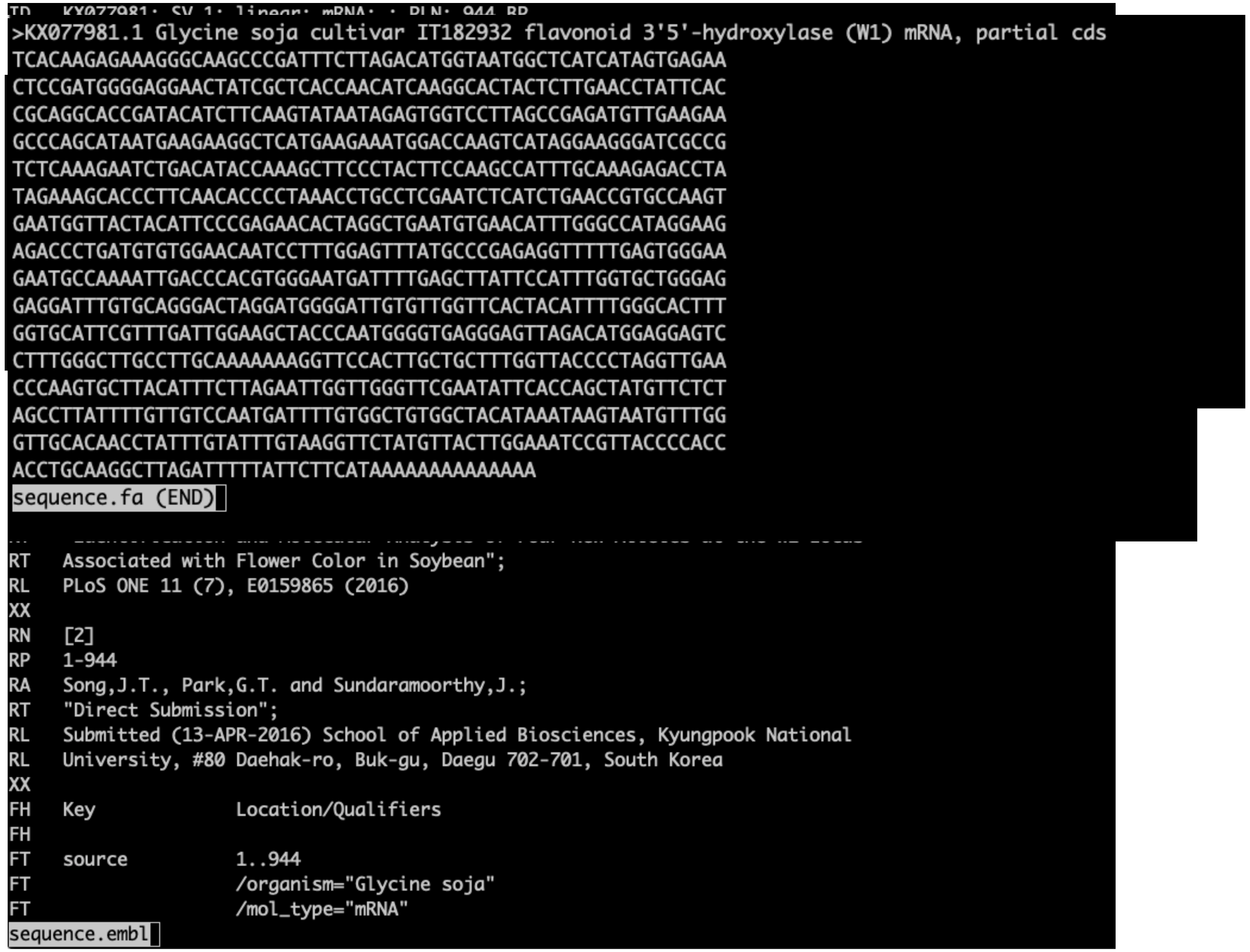

Genbank
fasta
embl
gff
...

Genbank



```
LOCUS       KX077981                 944 bp    mRNA    linear   PLN 01-NOV-2016
DEFINITION  Glycine soja cultivar IT182932 flavonoid 3'5'-hydroxylase (W1)
            mRNA, partial cds.
ACCESSION   KX077981
VERSION     KX077981.1
KEYWORDS    .
SOURCE      Glycine soja
  ORGANISM  Glycine soja
            Eukaryota; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta;
            Spermatophyta; Magnoliopsida; eudicotyledons; Gunneridae;
            Pentapetalae; rosids; fabids; Fabales; Fabaceae; Papilionoideae; 50
            kb inversion clade; NPAAA clade; indigoferoid/millettioid clade;
            Phaseoleae; Glycine; Glycine subgen. Soja.
REFERENCE   1  (bases 1 to 944)
  AUTHORS   Sundaramoorthy,J., Park,G.T., Chang,J.H., Lee,J.D., Kim,J.H.,
            Seo,H.S., Chung,G. and Song,J.T.
  TITLE     Identification and Molecular Analysis of Four New Alleles at the W1
            Locus Associated with Flower Color in Soybean
  JOURNAL   PLoS ONE 11 (7), E0159865 (2016)
   PUBMED   27442124
  REMARK    Publication Status: Online-Only
REFERENCE   2  (bases 1 to 944)
  AUTHORS   Song,J.T., Park,G.T. and Sundaramoorthy,J.
  TITLE     Direct Submission
  JOURNAL   Submitted (13-APR-2016) School of Applied Biosciences, Kyungpook
            National University, #80 Daehak-ro, Buk-gu, Daegu 702-701, South
            Korea
FEATURES             Location/Qualifiers
     source          1..944
                     /organism="Glycine soja"
                     /mol_type="mRNA"
                     /cultivar="IT182932"
                     /db_xref="taxon:3848"
     gene            <1..944
                     /gene="W1"
     CDS             <1..742
                     /gene="W1"
```

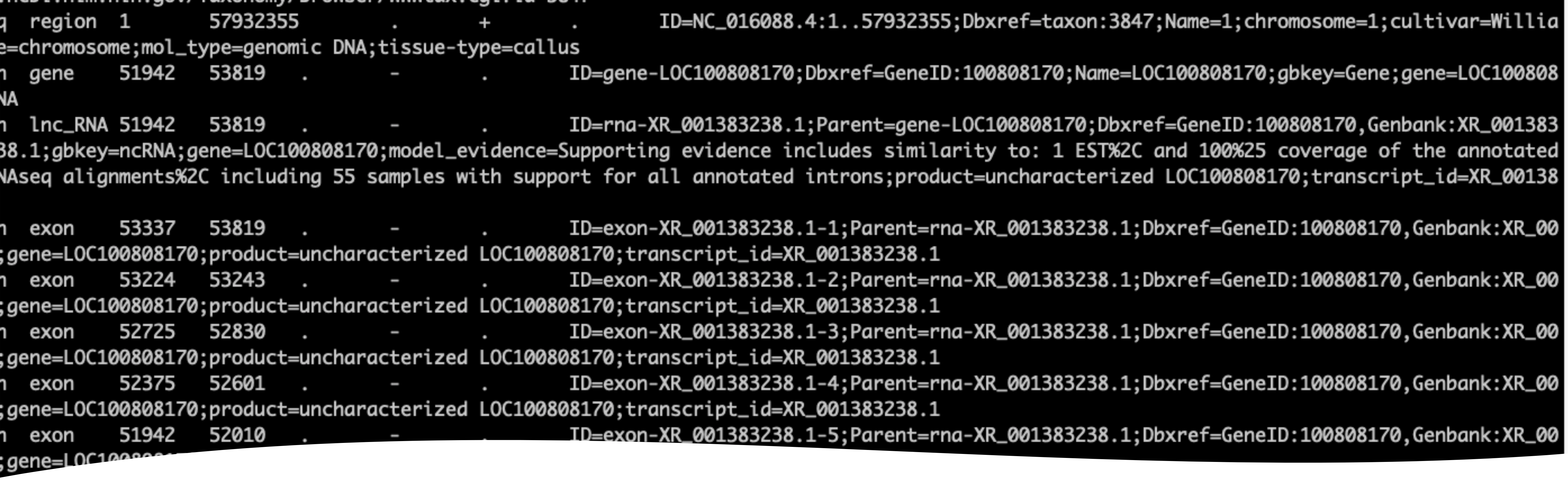
Genbank 2 fasta embl



```
ID   KX077981; SV 1; linear; mRNA; ; PLN; 944 BP.
```

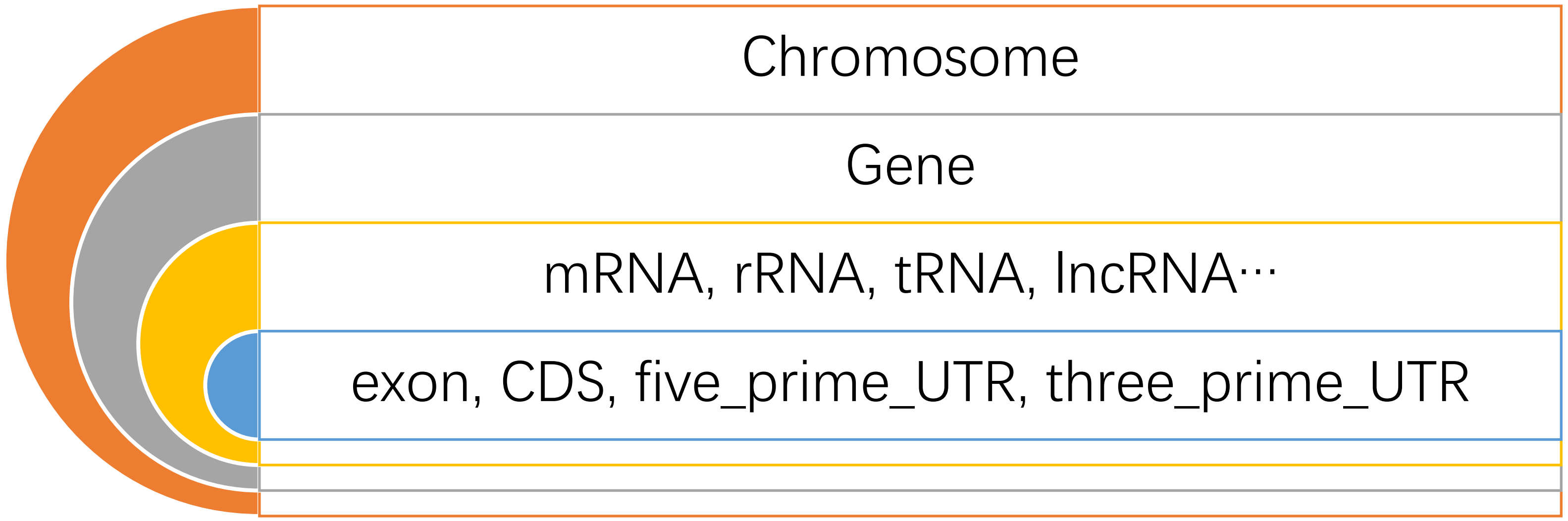
```
>KX077981.1 Glycine soja cultivar IT182932 flavonoid 3'5'-hydroxylase (W1) mRNA, partial cds
TCACAAGAGAAAGGGCAAGCCCGATTTCTTAGACATGGTAATGGCTCATCATAGTGAGAA
CTCCGATGGGGAGGAACTATCGCTCACCAACATCAAGGCACTACTCTTGAACCTATTCAC
CGCAGGCACCGATACATCTTCAAGTATAATAGAGTGGTCCTTAGCCGAGATGTTGAAGAA
GCCCAGCATAATGAAGAAGGCTCATGAAGAAATGGACCAAGTCATAGGAAGGGATCGCCG
TCTCAAAGAATCTGACATACCAAAGCTTCCCTACTTCCAAGCCATTTGCAAAGAGACCTA
TAGAAAGCACCCTTCAACACCCCTAAACCTGCCTCGAATCTCATCTGAACCGTGCCAAGT
GAATGGTTACTACATTCCCGAGAACACTAGGCTGAATGTGAACATTTGGGCCATAGGAAG
AGACCCTGATGTGTGGAACAATCCTTTGGAGTTTATGCCCGAGAGGTTTTTGAGTGGGAA
GAATGCCAAAATTGACCCACGTGGGAATGATTTTGAGCTTATTCCATTTGGTGCTGGGAG
GAGGATTTGTGCAGGGACTAGGATGGGGATTGTGTTGGTTCACTACATTTTGGGCACTTT
GGTGCATTCGTTTGATTGGAAGCTACCCAATGGGGTGAGGGAGTTAGACATGGAGGAGTC
CTTTGGGCTTGCCTTGCAAAAAAAGGTTCCACTTGCTGCTTTGGTTACCCCTAGGTTGAA
CCCAAGTGCTTACATTTCTTAGAATTGGTTGGGTTCGAATATTCACCAGCTATGTTCTCT
AGCCTTATTTTGTTGTCCAATGATTTTGTGGCTGTGGCTACATAAATAAGTAATGTTTGG
GTTGCACAACCTATTTGTATTTGTAAGGTTCTATGTTACTTGGAAATCCGTTACCCCACC
ACCTGCAAGGCTTAGATTTTTATTCTTCATAAAAAAAAAAAAAAA
sequence.fa (END)
```

```
RT   Associated with Flower Color in Soybean";
RL   PLoS ONE 11 (7), E0159865 (2016)
XX
RN   [2]
RP   1-944
RA   Song,J.T., Park,G.T. and Sundaramoorthy,J.;
RT   "Direct Submission";
RL   Submitted (13-APR-2016) School of Applied Biosciences, Kyungpook National
RL   University, #80 Daehak-ro, Buk-gu, Daegu 702-701, South Korea
XX
FH   Key             Location/Qualifiers
FH
FT   source          1..944
FT                   /organism="Glycine soja"
FT                   /mol_type="mRNA"
sequence.embl
```

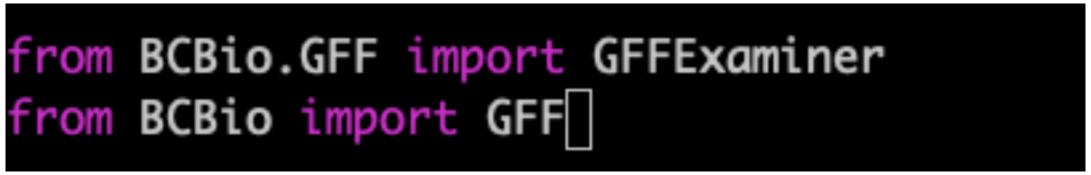


```
q  region  1       57932355        .       +       .       ID=NC_016088.4:1..57932355;Dbxref=taxon:3847;Name=1;chromosome=1;cultivar=Willia
e=chromosome;mol_type=genomic DNA;tissue-type=callus
n  gene    51942   53819   .       -       .       ID=gene-LOC100808170;Dbxref=GeneID:100808170;Name=LOC100808170;gbkey=Gene;gene=LOC100808
NA
n  lnc_RNA 51942   53819   .       -       .       ID=rna-XR_001383238.1;Parent=gene-LOC100808170;Dbxref=GeneID:100808170,Genbank:XR_001383
38.1;gbkey=ncRNA;gene=LOC100808170;model_evidence=Supporting evidence includes similarity to: 1 EST%2C and 100%25 coverage of the annotated
NAseq alignments%2C including 55 samples with support for all annotated introns;product=uncharacterized LOC100808170;transcript_id=XR_00138

n  exon    53337   53819   .       -       .       ID=exon-XR_001383238.1-1;Parent=rna-XR_001383238.1;Dbxref=GeneID:100808170,Genbank:XR_00
;gene=LOC100808170;product=uncharacterized LOC100808170;transcript_id=XR_001383238.1
n  exon    53224   53243   .       -       .       ID=exon-XR_001383238.1-2;Parent=rna-XR_001383238.1;Dbxref=GeneID:100808170,Genbank:XR_00
;gene=LOC100808170;product=uncharacterized LOC100808170;transcript_id=XR_001383238.1
n  exon    52725   52830   .       -       .       ID=exon-XR_001383238.1-3;Parent=rna-XR_001383238.1;Dbxref=GeneID:100808170,Genbank:XR_00
;gene=LOC100808170;product=uncharacterized LOC100808170;transcript_id=XR_001383238.1
n  exon    52375   52601   .       -       .       ID=exon-XR_001383238.1-4;Parent=rna-XR_001383238.1;Dbxref=GeneID:100808170,Genbank:XR_00
;gene=LOC100808170;product=uncharacterized LOC100808170;transcript_id=XR_001383238.1
n  exon    51942   52010   .       -       .       ID=exon-XR_001383238.1-5;Parent=rna-XR_001383238.1;Dbxref=GeneID:100808170,Genbank:XR_00
;gene=LOC10
```

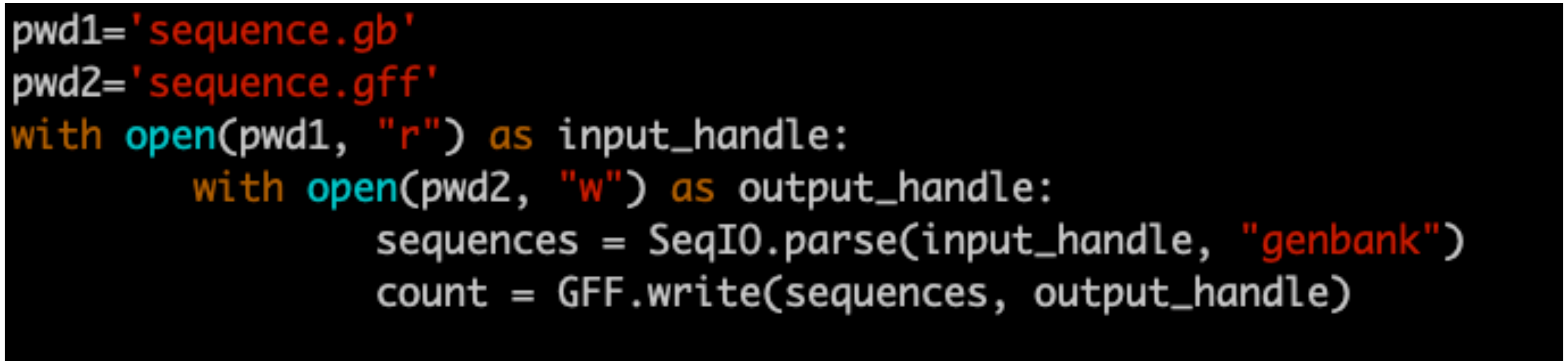
GFF file


Chromosome
Gene
mRNA, rRNA, tRNA, lncRNA…
exon, CDS, five_prime_UTR, three_prime_UTR

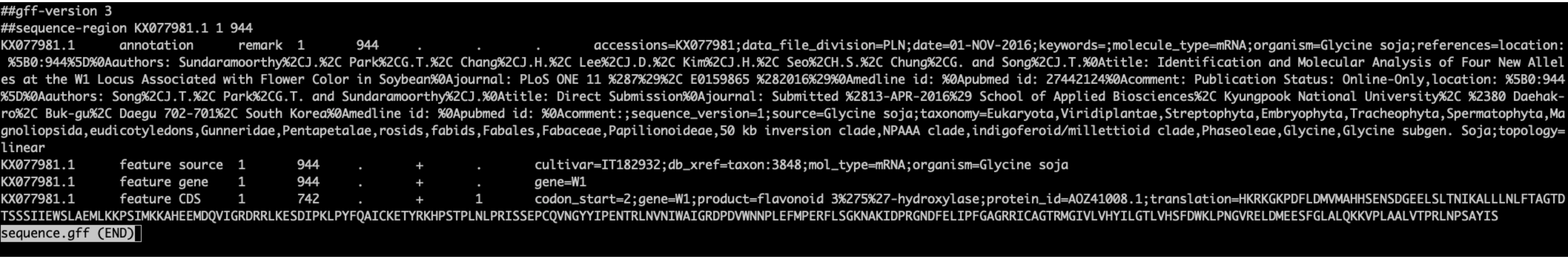
pip install bcbio-gff



```python
from BCBio.GFF import GFFExaminer
from BCBio import GFF
```

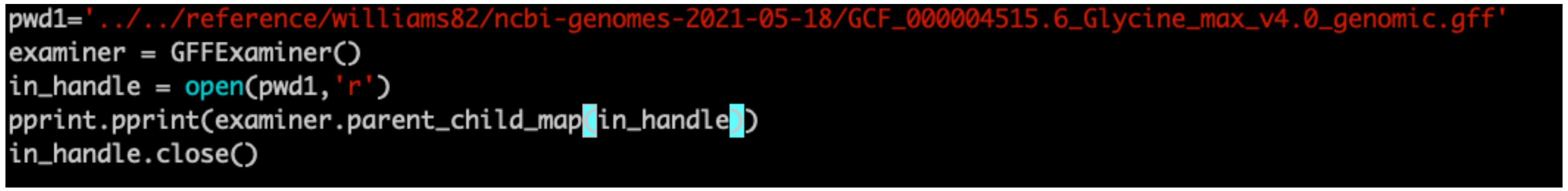


```python
pwd1='sequence.gb'
pwd2='sequence.gff'
with open(pwd1, "r") as input_handle:
        with open(pwd2, "w") as output_handle:
                sequences = SeqIO.parse(input_handle, "genbank")
                count = GFF.write(sequences, output_handle)
```

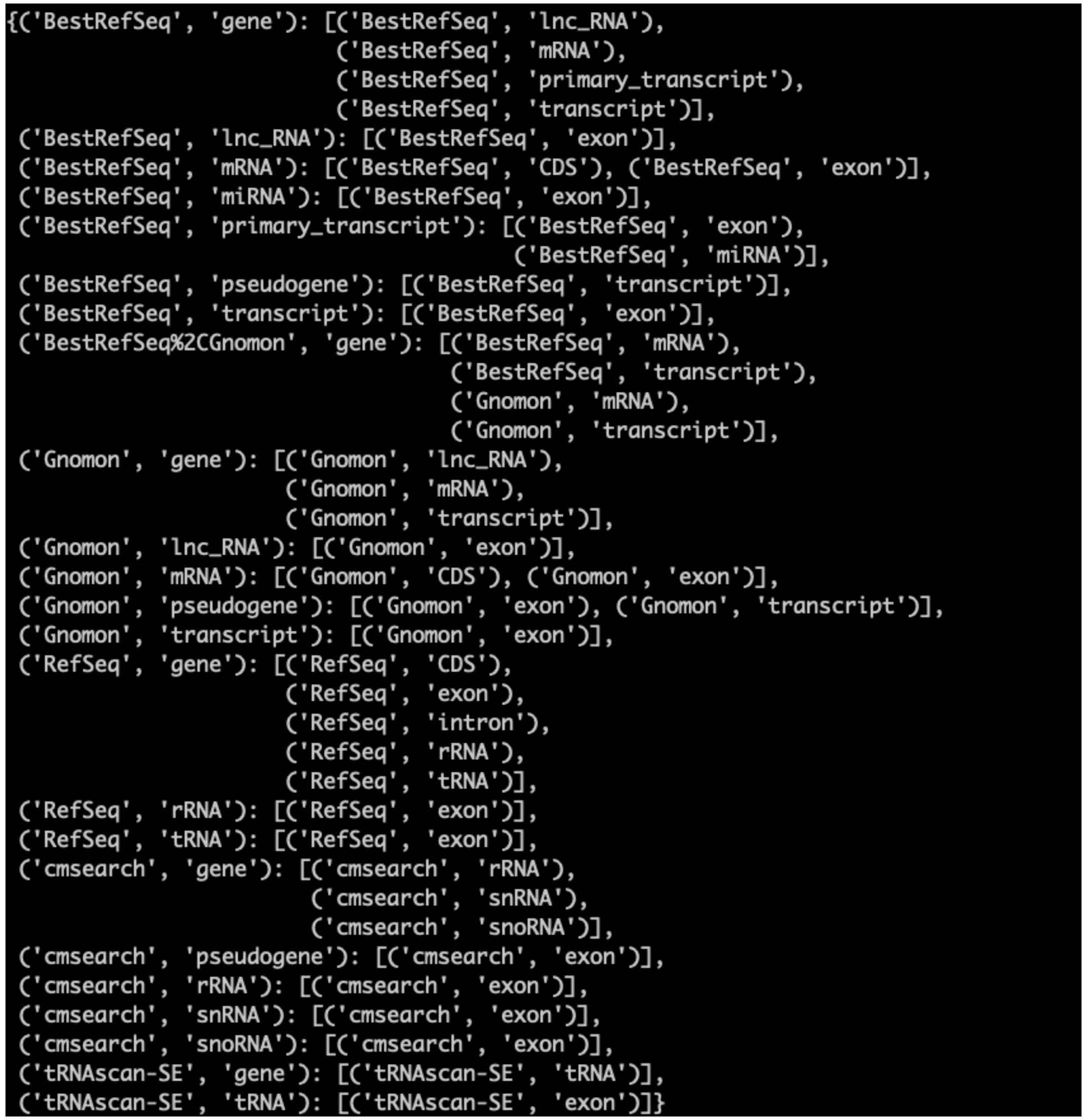


```
##gff-version 3
##sequence-region KX077981.1 1 944
KX077981.1      annotation      remark  1       944     .       .       .       accessions=KX077981;data_file_division=PLN;date=01-NOV-2016;keywords=;molecule_type=mRNA;organism=Glycine soja;references=location:
 %5B0:944%5D%0Aauthors: Sundaramoorthy%2CJ.%2C Park%2CG.T.%2C Chang%2CJ.H.%2C Lee%2CJ.D.%2C Kim%2CJ.H.%2C Seo%2CH.S.%2C Chung%2CG. and Song%2CJ.T.%0Atitle: Identification and Molecular Analysis of Four New Allel
es at the W1 Locus Associated with Flower Color in Soybean%0Ajournal: PLoS ONE 11 %287%29%2C E0159865 %282016%29%0Amedline id: %0Apubmed id: 27442124%0Acomment: Publication Status: Online-Only,location: %5B0:944
%5D%0Aauthors: Song%2CJ.T.%2C Park%2CG.T. and Sundaramoorthy%2CJ.%0Atitle: Direct Submission%0Ajournal: Submitted %2813-APR-2016%29 School of Applied Biosciences%2C Kyungpook National University%2C %2380 Daehak-
ro%2C Buk-gu%2C Daegu 702-701%2C South Korea%0Amedline id: %0Apubmed id: %0Acomment:;sequence_version=1;source=Glycine soja;taxonomy=Eukaryota,Viridiplantae,Streptophyta,Embryophyta,Tracheophyta,Spermatophyta,Ma
gnoliopsida,eudicotyledons,Gunneridae,Pentapetalae,rosids,fabids,Fabales,Fabaceae,Papilionoideae,50 kb inversion clade,NPAAA clade,indigoferoid/millettioid clade,Phaseoleae,Glycine,Glycine subgen. Soja;topology=
linear
KX077981.1      feature source  1       944     .       +       .       cultivar=IT182932;db_xref=taxon:3848;mol_type=mRNA;organism=Glycine soja
KX077981.1      feature gene    1       944     .       +       .       gene=W1
KX077981.1      feature CDS     1       742     .       +       1       codon_start=2;gene=W1;product=flavonoid 3%275%27-hydroxylase;protein_id=AOZ41008.1;translation=HKRKGKPDFLDMVMAHHSENSDGEELSLTNIKALLLNLFTAGTD
TSSSIIEWSLAEMLKKPSIMKKAHEEMDQVIGRDRRLKESDIPKLPYFQAICKETYRKHPSTPLNLPRISSEPCQVNGYYIPENTRLNVNIWAIGRDPDVWNNPLEFMPERFLSGKNAKIDPRGNDFELIPFGAGRRICAGTRMGIVLVHYILGTLVHSFDWKLPNGVRELDMEESFGLALQKKVPLAALVTPRLNPSAYIS
sequence.gff (END)
```

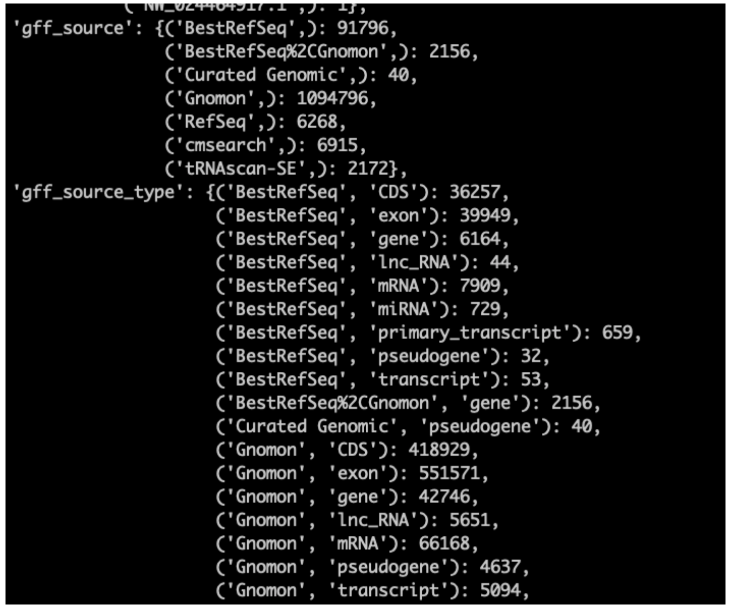
GFF



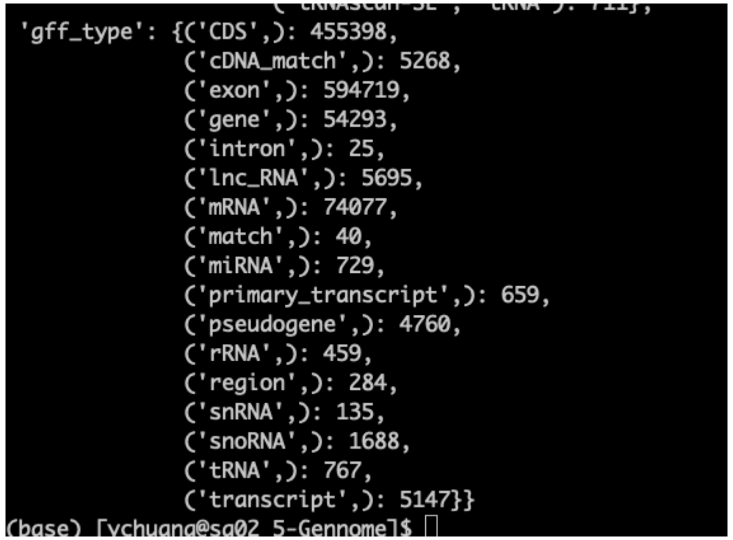
```
pwd1='../../reference/williams82/ncbi-genomes-2021-05-18/GCF_000004515.6_Glycine_max_v4.0_genomic.gff'
examiner = GFFExaminer()
in_handle = open(pwd1,'r')
pprint.pprint(examiner.parent_child_map(in_handle))
in_handle.close()
```



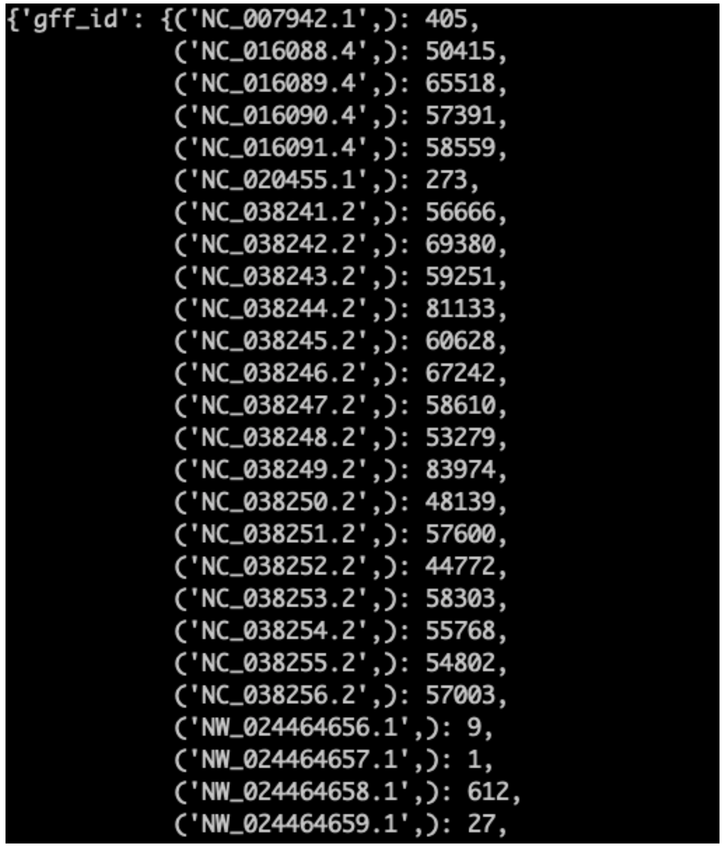
```
{('BestRefSeq', 'gene'): [('BestRefSeq', 'lnc_RNA'),
                          ('BestRefSeq', 'mRNA'),
                          ('BestRefSeq', 'primary_transcript'),
                          ('BestRefSeq', 'transcript')],
 ('BestRefSeq', 'lnc_RNA'): [('BestRefSeq', 'exon')],
 ('BestRefSeq', 'mRNA'): [('BestRefSeq', 'CDS'), ('BestRefSeq', 'exon')],
 ('BestRefSeq', 'miRNA'): [('BestRefSeq', 'exon')],
 ('BestRefSeq', 'primary_transcript'): [('BestRefSeq', 'exon'),
                                        ('BestRefSeq', 'miRNA')],
 ('BestRefSeq', 'pseudogene'): [('BestRefSeq', 'transcript')],
 ('BestRefSeq', 'transcript'): [('BestRefSeq', 'exon')],
 ('BestRefSeq%2CGnomon', 'gene'): [('BestRefSeq', 'mRNA'),
                                   ('BestRefSeq', 'transcript'),
                                   ('Gnomon', 'mRNA'),
                                   ('Gnomon', 'transcript')],
 ('Gnomon', 'gene'): [('Gnomon', 'lnc_RNA'),
                      ('Gnomon', 'mRNA'),
                      ('Gnomon', 'transcript')],
 ('Gnomon', 'lnc_RNA'): [('Gnomon', 'exon')],
 ('Gnomon', 'mRNA'): [('Gnomon', 'CDS'), ('Gnomon', 'exon')],
 ('Gnomon', 'pseudogene'): [('Gnomon', 'exon'), ('Gnomon', 'transcript')],
 ('Gnomon', 'transcript'): [('Gnomon', 'exon')],
 ('RefSeq', 'gene'): [('RefSeq', 'CDS'),
                      ('RefSeq', 'exon'),
                      ('RefSeq', 'intron'),
                      ('RefSeq', 'rRNA'),
                      ('RefSeq', 'tRNA')],
 ('RefSeq', 'rRNA'): [('RefSeq', 'exon')],
 ('RefSeq', 'tRNA'): [('RefSeq', 'exon')],
 ('cmsearch', 'gene'): [('cmsearch', 'rRNA'),
                        ('cmsearch', 'snRNA'),
                        ('cmsearch', 'snoRNA')],
 ('cmsearch', 'pseudogene'): [('cmsearch', 'exon')],
 ('cmsearch', 'rRNA'): [('cmsearch', 'exon')],
 ('cmsearch', 'snRNA'): [('cmsearch', 'exon')],
 ('cmsearch', 'snoRNA'): [('cmsearch', 'exon')],
 ('tRNAscan-SE', 'gene'): [('tRNAscan-SE', 'tRNA')],
 ('tRNAscan-SE', 'tRNA'): [('tRNAscan-SE', 'exon')]}
```



```
'gff_source': {('BestRefSeq',): 91796,
               ('BestRefSeq%2CGnomon',): 2156,
               ('Curated Genomic',): 40,
               ('Gnomon',): 1094796,
               ('RefSeq',): 6268,
               ('cmsearch',): 6915,
               ('tRNAscan-SE',): 2172},
'gff_source_type': {('BestRefSeq', 'CDS'): 36257,
                    ('BestRefSeq', 'exon'): 39949,
                    ('BestRefSeq', 'gene'): 6164,
                    ('BestRefSeq', 'lnc_RNA'): 44,
                    ('BestRefSeq', 'mRNA'): 7909,
                    ('BestRefSeq', 'miRNA'): 729,
                    ('BestRefSeq', 'primary_transcript'): 659,
                    ('BestRefSeq', 'pseudogene'): 32,
                    ('BestRefSeq', 'transcript'): 53,
                    ('BestRefSeq%2CGnomon', 'gene'): 2156,
                    ('Curated Genomic', 'pseudogene'): 40,
                    ('Gnomon', 'CDS'): 418929,
                    ('Gnomon', 'exon'): 551571,
                    ('Gnomon', 'gene'): 42746,
                    ('Gnomon', 'lnc_RNA'): 5651,
                    ('Gnomon', 'mRNA'): 66168,
                    ('Gnomon', 'pseudogene'): 4637,
                    ('Gnomon', 'transcript'): 5094,
```



```
 'gff_type': {('CDS',): 455398,
              ('cDNA_match',): 5268,
              ('exon',): 594719,
              ('gene',): 54293,
              ('intron',): 25,
              ('lnc_RNA',): 5695,
              ('mRNA',): 74077,
              ('match',): 40,
              ('miRNA',): 729,
              ('primary_transcript',): 659,
              ('pseudogene',): 4760,
              ('rRNA',): 459,
              ('region',): 284,
              ('snRNA',): 135,
              ('snoRNA',): 1688,
              ('tRNA',): 767,
              ('transcript',): 5147}}
(base) [ychuang@sa02 5-Gennome]$ 
```



```
{'gff_id': {('NC_007942.1',): 405,
            ('NC_016088.4',): 50415,
            ('NC_016089.4',): 65518,
            ('NC_016090.4',): 57391,
            ('NC_016091.4',): 58559,
            ('NC_020455.1',): 273,
            ('NC_038241.2',): 56666,
            ('NC_038242.2',): 69380,
            ('NC_038243.2',): 59251,
            ('NC_038244.2',): 81133,
            ('NC_038245.2',): 60628,
            ('NC_038246.2',): 67242,
            ('NC_038247.2',): 58610,
            ('NC_038248.2',): 53279,
            ('NC_038249.2',): 83974,
            ('NC_038250.2',): 48139,
            ('NC_038251.2',): 57600,
            ('NC_038252.2',): 44772,
            ('NC_038253.2',): 58303,
            ('NC_038254.2',): 55768,
            ('NC_038255.2',): 54802,
            ('NC_038256.2',): 57003,
            ('NW_024464656.1',): 9,
            ('NW_024464657.1',): 1,
            ('NW_024464658.1',): 612,
            ('NW_024464659.1',): 27,
```

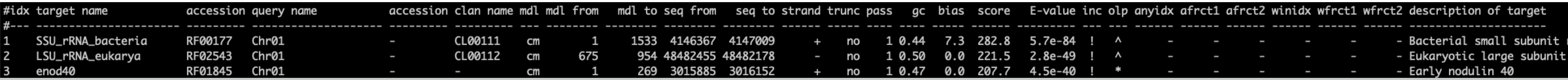
pprint.pprint(examiner.available_limits(in_handle))

From tblast to GFF

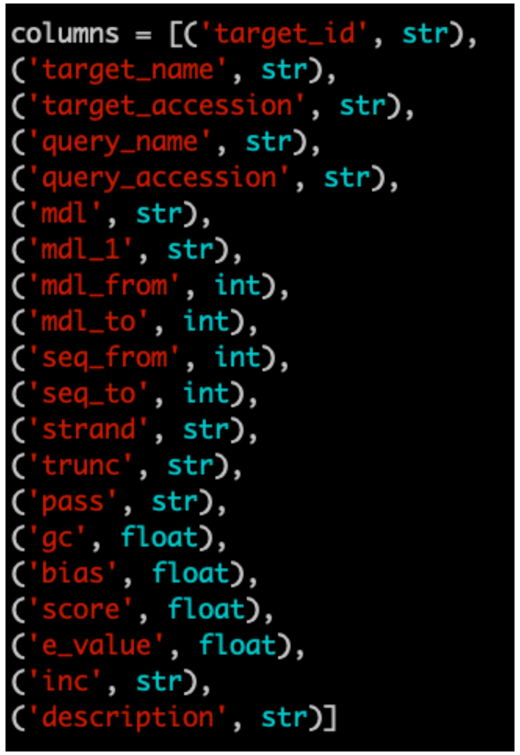
An example

cmscan.tblout

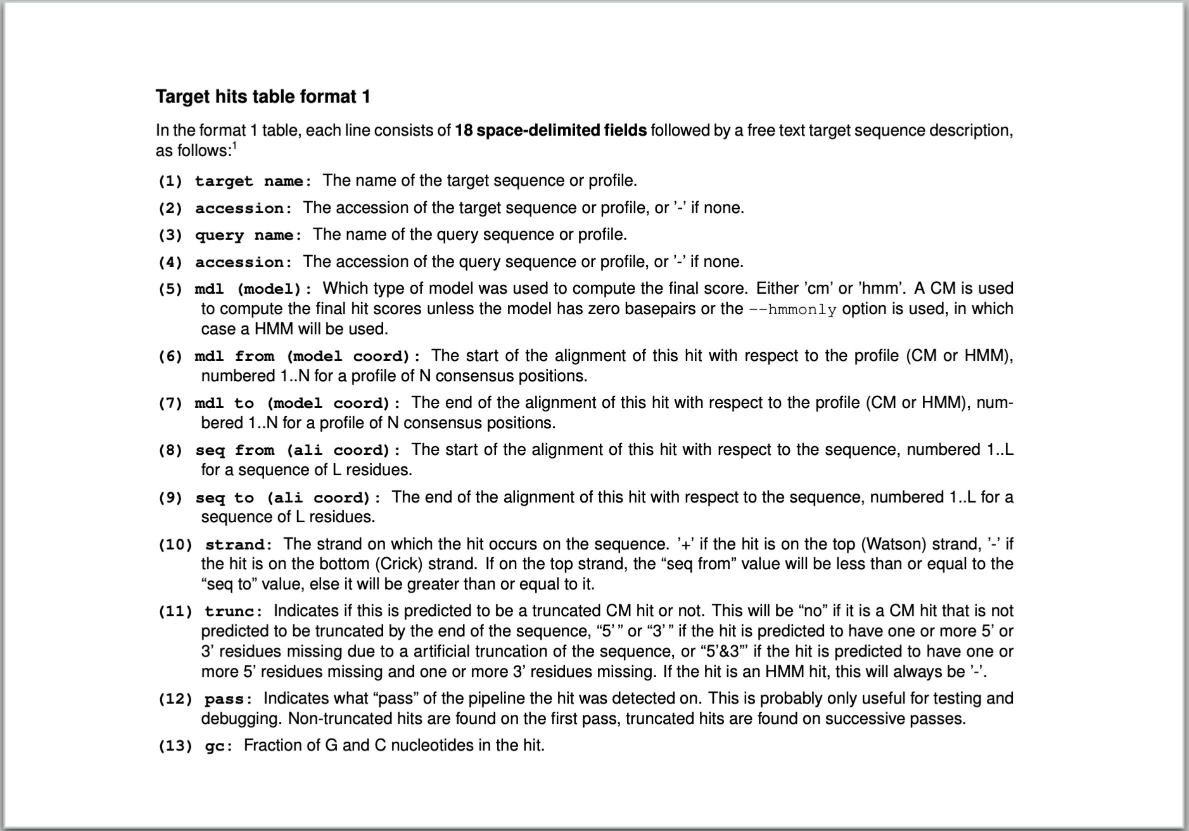
http://eddylab.org/infernal/Userguide.pdf



```
#idx target name          accession query name           accession clan name mdl mdl from   mdl to seq from   seq to strand trunc pass   gc  bias  score   E-value inc olp anyidx afrct1 afrct2 winidx wfrct1 wfrct2 description of target
#--- -------------------- --------- -------------------- --------- --------- --- -------- -------- -------- -------- ------ ----- ---- ---- ----- ------ --------- --- --- ------ ------ ------ ------ ------ ------ ---------------------
1    SSU_rRNA_bacteria    RF00177   Chr01                -         CL00111    cm        1     1533  4146367  4147009      +    no    1 0.44   7.3  282.8   5.7e-84 !   ^        -      -      -      -      -      - Bacterial small subunit
2    LSU_rRNA_eukarya     RF02543   Chr01                -         CL00112    cm      675      954 48482455 48482178      -    no    1 0.50   0.0  221.5   2.8e-49 !   ^        -      -      -      -      -      - Eukaryotic large subunit
3    enod40               RF01845   Chr01                -         -          cm        1      269  3015885  3016152      +    no    1 0.47   0.0  207.7   4.5e-40 !   *        -      -      -      -      -      - Early nodulin 40
```



```
columns = [('target_id', str),
('target_name', str),
('target_accession', str),
('query_name', str),
('query_accession', str),
('mdl', str),
('mdl_1', str),
('mdl_from', int),
('mdl_to', int),
('seq_from', int),
('seq_to', int),
('strand', str),
('trunc', str),
('pass', str),
('gc', float),
('bias', float),
('score', float),
('e_value', float),
('inc', str),
('description', str)]
```



Target hits table format 1

In the format 1 table, each line consists of **18 space-delimited fields** followed by a free text target sequence description, as follows:[1]

(1) target name: The name of the target sequence or profile.

(2) accession: The accession of the target sequence or profile, or '-' if none.

(3) query name: The name of the query sequence or profile.

(4) accession: The accession of the query sequence or profile, or '-' if none.

(5) mdl (model): Which type of model was used to compute the final score. Either 'cm' or 'hmm'. A CM is used to compute the final hit scores unless the model has zero basepairs or the `--hmmonly` option is used, in which case a HMM will be used.

(6) mdl from (model coord): The start of the alignment of this hit with respect to the profile (CM or HMM), numbered 1..N for a profile of N consensus positions.

(7) mdl to (model coord): The end of the alignment of this hit with respect to the profile (CM or HMM), numbered 1..N for a profile of N consensus positions.

(8) seq from (ali coord): The start of the alignment of this hit with respect to the sequence, numbered 1..L for a sequence of L residues.

(9) seq to (ali coord): The end of the alignment of this hit with respect to the sequence, numbered 1..L for a sequence of L residues.

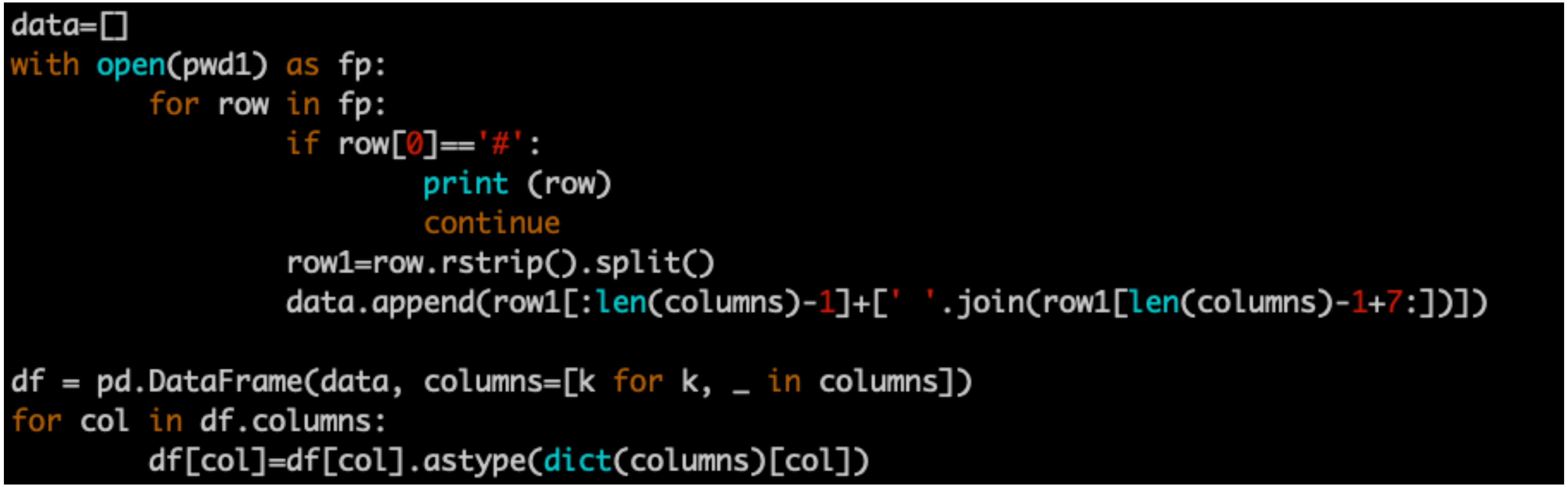
(10) strand: The strand on which the hit occurs on the sequence. '+' if the hit is on the top (Watson) strand, '-' if the hit is on the bottom (Crick) strand. If on the top strand, the "seq from" value will be less than or equal to the "seq to" value, else it will be greater than or equal to it.

(11) trunc: Indicates if this is predicted to be a truncated CM hit or not. This will be "no" if it is a CM hit that is not predicted to be truncated by the end of the sequence, "5' " or "3' " if the hit is predicted to have one or more 5' or 3' residues missing due to a artificial truncation of the sequence, or "5'&3'" if the hit is predicted to have one or more 5' residues missing and one or more 3' residues missing. If the hit is an HMM hit, this will always be '-'.

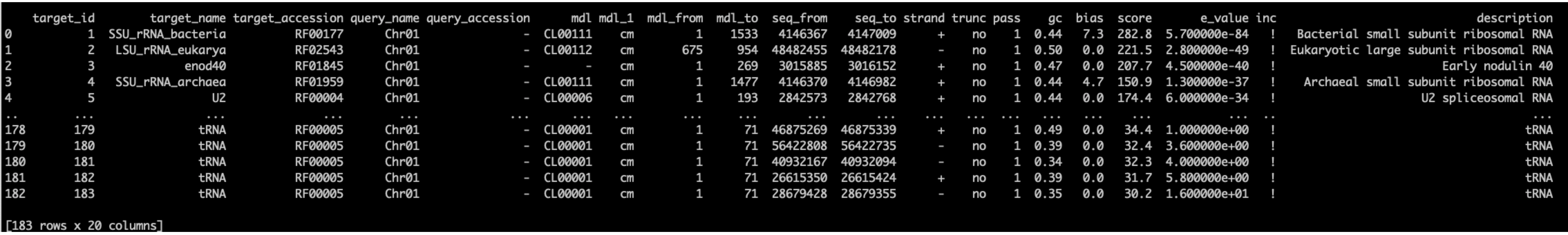
(12) pass: Indicates what "pass" of the pipeline the hit was detected on. This is probably only useful for testing and debugging. Non-truncated hits are found on the first pass, truncated hits are found on successive passes.

(13) gc: Fraction of G and C nucleotides in the hit.

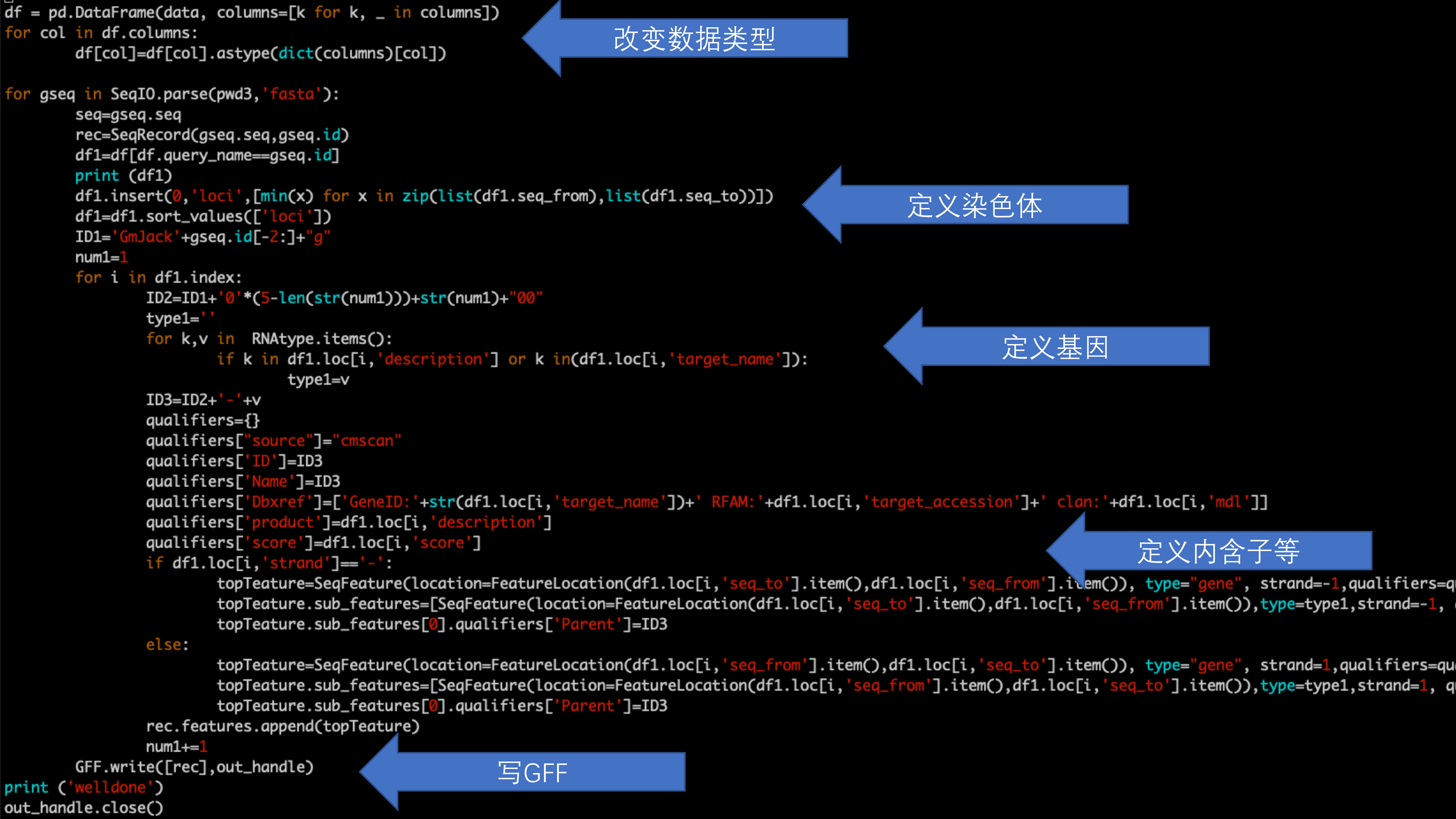
Read tblast File



```
data=[]
with open(pwd1) as fp:
        for row in fp:
                if row[0]=='#':
                        print (row)
                        continue
                row1=row.rstrip().split()
                data.append(row1[:len(columns)-1]+[' '.join(row1[len(columns)-1+7:])])

df = pd.DataFrame(data, columns=[k for k, _ in columns])
for col in df.columns:
        df[col]=df[col].astype(dict(columns)[col])
```

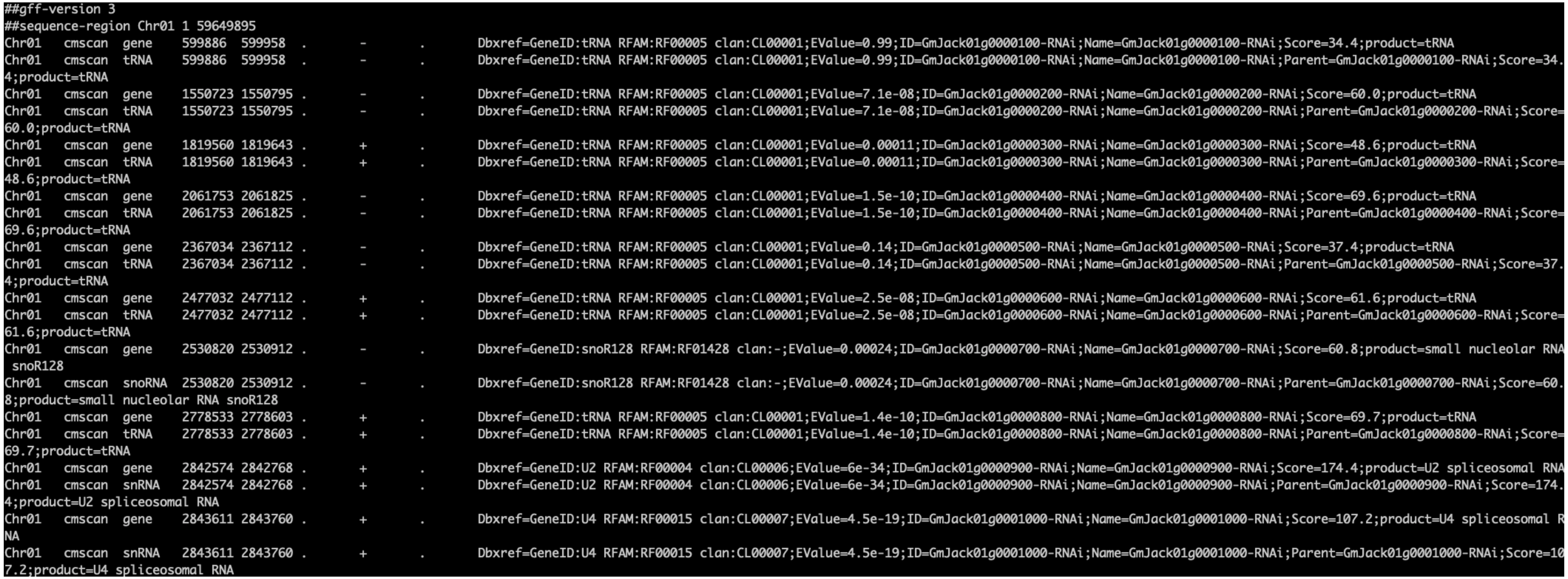


```
    target_id        target_name target_accession query_name query_accession      mdl mdl_1  mdl_from  mdl_to  seq_from    seq_to strand trunc pass    gc  bias  score       e_value inc                                 description
0           1  SSU_rRNA_bacteria           RF00177      Chr01               -  CL00111    cm         1    1533   4146367   4147009      +    no    1  0.44   7.3  282.8  5.700000e-84   !     Bacterial small subunit ribosomal RNA
1           2   LSU_rRNA_eukarya           RF02543      Chr01               -  CL00112    cm       675     954  48482455  48482178      -    no    1  0.50   0.0  221.5  2.800000e-49   !  Eukaryotic large subunit ribosomal RNA
2           3             enod40           RF01845      Chr01               -        -    cm         1     269   3015885   3016152      +    no    1  0.47   0.0  207.7  4.500000e-40   !                        Early nodulin 40
3           4   SSU_rRNA_archaea           RF01959      Chr01               -  CL00111    cm         1    1477   4146370   4146982      +    no    1  0.44   4.7  150.9  1.300000e-37   !      Archaeal small subunit ribosomal RNA
4           5                 U2           RF00004      Chr01               -  CL00006    cm         1     193   2842573   2842768      +    no    1  0.44   0.0  174.4  6.000000e-34   !                      U2 spliceosomal RNA
..        ...                ...               ...        ...             ...      ...   ...       ...     ...       ...       ...    ...   ...  ...   ...   ...    ...           ...  ..                                     ...
178       179               tRNA           RF00005      Chr01               -  CL00001    cm         1      71  46875269  46875339      +    no    1  0.49   0.0   34.4  1.000000e+00   !                                    tRNA
179       180               tRNA           RF00005      Chr01               -  CL00001    cm         1      71  56422808  56422735      -    no    1  0.39   0.0   32.4  3.600000e+00   !                                    tRNA
180       181               tRNA           RF00005      Chr01               -  CL00001    cm         1      71  40932167  40932094      -    no    1  0.34   0.0   32.3  4.000000e+00   !                                    tRNA
181       182               tRNA           RF00005      Chr01               -  CL00001    cm         1      71  26615350  26615424      +    no    1  0.39   0.0   31.7  5.800000e+00   !                                    tRNA
182       183               tRNA           RF00005      Chr01               -  CL00001    cm         1      71  28679428  28679355      -    no    1  0.35   0.0   30.2  1.600000e+01   !                                    tRNA

[183 rows x 20 columns]
```



```
df = pd.DataFrame(data, columns=[k for k, _ in columns])
for col in df.columns:
        df[col]=df[col].astype(dict(columns)[col])

for gseq in SeqIO.parse(pwd3,'fasta'):
        seq=gseq.seq
        rec=SeqRecord(gseq.seq,gseq.id)
        df1=df[df.query_name==gseq.id]
        print (df1)
        df1.insert(0,'loci',[min(x) for x in zip(list(df1.seq_from),list(df1.seq_to))])
        df1=df1.sort_values(['loci'])
        ID1='GmJack'+gseq.id[-2:]+"g"
        num1=1
        for i in df1.index:
                ID2=ID1+'0'*(5-len(str(num1)))+str(num1)+"00"
                type1=''
                for k,v in  RNAtype.items():
                        if k in df1.loc[i,'description'] or k in(df1.loc[i,'target_name']):
                                type1=v
                ID3=ID2+'-'+v
                qualifiers={}
                qualifiers["source"]="cmscan"
                qualifiers['ID']=ID3
                qualifiers['Name']=ID3
                qualifiers['Dbxref']=['GeneID:'+str(df1.loc[i,'target_name'])+' RFAM:'+df1.loc[i,'target_accession']+' clan:'+df1.loc[i,'mdl']]
                qualifiers['product']=df1.loc[i,'description']
                qualifiers['score']=df1.loc[i,'score']
                if df1.loc[i,'strand']=='-':
                        topTeature=SeqFeature(location=FeatureLocation(df1.loc[i,'seq_to'].item(),df1.loc[i,'seq_from'].item()), type="gene", strand=-1,qualifiers=q
                        topTeature.sub_features=[SeqFeature(location=FeatureLocation(df1.loc[i,'seq_to'].item(),df1.loc[i,'seq_from'].item()),type=type1,strand=-1, 
                        topTeature.sub_features[0].qualifiers['Parent']=ID3
                else:
                        topTeature=SeqFeature(location=FeatureLocation(df1.loc[i,'seq_from'].item(),df1.loc[i,'seq_to'].item()), type="gene", strand=1,qualifiers=qu
                        topTeature.sub_features=[SeqFeature(location=FeatureLocation(df1.loc[i,'seq_from'].item(),df1.loc[i,'seq_to'].item()),type=type1,strand=1, q
                        topTeature.sub_features[0].qualifiers['Parent']=ID3
                rec.features.append(topTeature)
                num1+=1
        GFF.write([rec],out_handle)
print ('welldone')
out_handle.close()
```

Result



```
##gff-version 3
##sequence-region Chr01 1 59649895
Chr01   cmscan  gene    599886  599958  .       -       .       Dbxref=GeneID:tRNA RFAM:RF00005 clan:CL00001;EValue=0.99;ID=GmJack01g0000100-RNAi;Name=GmJack01g0000100-RNAi;Score=34.4;product=tRNA
Chr01   cmscan  tRNA    599886  599958  .       -       .       Dbxref=GeneID:tRNA RFAM:RF00005 clan:CL00001;EValue=0.99;ID=GmJack01g0000100-RNAi;Name=GmJack01g0000100-RNAi;Parent=GmJack01g0000100-RNAi;Score=34.
4;product=tRNA
Chr01   cmscan  gene    1550723 1550795 .       -       .       Dbxref=GeneID:tRNA RFAM:RF00005 clan:CL00001;EValue=7.1e-08;ID=GmJack01g0000200-RNAi;Name=GmJack01g0000200-RNAi;Score=60.0;product=tRNA
Chr01   cmscan  tRNA    1550723 1550795 .       -       .       Dbxref=GeneID:tRNA RFAM:RF00005 clan:CL00001;EValue=7.1e-08;ID=GmJack01g0000200-RNAi;Name=GmJack01g0000200-RNAi;Parent=GmJack01g0000200-RNAi;Score=
60.0;product=tRNA
Chr01   cmscan  gene    1819560 1819643 .       +       .       Dbxref=GeneID:tRNA RFAM:RF00005 clan:CL00001;EValue=0.00011;ID=GmJack01g0000300-RNAi;Name=GmJack01g0000300-RNAi;Score=48.6;product=tRNA
Chr01   cmscan  tRNA    1819560 1819643 .       +       .       Dbxref=GeneID:tRNA RFAM:RF00005 clan:CL00001;EValue=0.00011;ID=GmJack01g0000300-RNAi;Name=GmJack01g0000300-RNAi;Parent=GmJack01g0000300-RNAi;Score=
48.6;product=tRNA
Chr01   cmscan  gene    2061753 2061825 .       -       .       Dbxref=GeneID:tRNA RFAM:RF00005 clan:CL00001;EValue=1.5e-10;ID=GmJack01g0000400-RNAi;Name=GmJack01g0000400-RNAi;Score=69.6;product=tRNA
Chr01   cmscan  tRNA    2061753 2061825 .       -       .       Dbxref=GeneID:tRNA RFAM:RF00005 clan:CL00001;EValue=1.5e-10;ID=GmJack01g0000400-RNAi;Name=GmJack01g0000400-RNAi;Parent=GmJack01g0000400-RNAi;Score=
69.6;product=tRNA
Chr01   cmscan  gene    2367034 2367112 .       -       .       Dbxref=GeneID:tRNA RFAM:RF00005 clan:CL00001;EValue=0.14;ID=GmJack01g0000500-RNAi;Name=GmJack01g0000500-RNAi;Score=37.4;product=tRNA
Chr01   cmscan  tRNA    2367034 2367112 .       -       .       Dbxref=GeneID:tRNA RFAM:RF00005 clan:CL00001;EValue=0.14;ID=GmJack01g0000500-RNAi;Name=GmJack01g0000500-RNAi;Parent=GmJack01g0000500-RNAi;Score=37.
4;product=tRNA
Chr01   cmscan  gene    2477032 2477112 .       +       .       Dbxref=GeneID:tRNA RFAM:RF00005 clan:CL00001;EValue=2.5e-08;ID=GmJack01g0000600-RNAi;Name=GmJack01g0000600-RNAi;Score=61.6;product=tRNA
Chr01   cmscan  tRNA    2477032 2477112 .       +       .       Dbxref=GeneID:tRNA RFAM:RF00005 clan:CL00001;EValue=2.5e-08;ID=GmJack01g0000600-RNAi;Name=GmJack01g0000600-RNAi;Parent=GmJack01g0000600-RNAi;Score=
61.6;product=tRNA
Chr01   cmscan  gene    2530820 2530912 .       -       .       Dbxref=GeneID:snoR128 RFAM:RF01428 clan:-;EValue=0.00024;ID=GmJack01g0000700-RNAi;Name=GmJack01g0000700-RNAi;Score=60.8;product=small nucleolar RNA
 snoR128
Chr01   cmscan  snoRNA  2530820 2530912 .       -       .       Dbxref=GeneID:snoR128 RFAM:RF01428 clan:-;EValue=0.00024;ID=GmJack01g0000700-RNAi;Name=GmJack01g0000700-RNAi;Parent=GmJack01g0000700-RNAi;Score=60.
8;product=small nucleolar RNA snoR128
Chr01   cmscan  gene    2778533 2778603 .       +       .       Dbxref=GeneID:tRNA RFAM:RF00005 clan:CL00001;EValue=1.4e-10;ID=GmJack01g0000800-RNAi;Name=GmJack01g0000800-RNAi;Score=69.7;product=tRNA
Chr01   cmscan  tRNA    2778533 2778603 .       +       .       Dbxref=GeneID:tRNA RFAM:RF00005 clan:CL00001;EValue=1.4e-10;ID=GmJack01g0000800-RNAi;Name=GmJack01g0000800-RNAi;Parent=GmJack01g0000800-RNAi;Score=
69.7;product=tRNA
Chr01   cmscan  gene    2842574 2842768 .       +       .       Dbxref=GeneID:U2 RFAM:RF00004 clan:CL00006;EValue=6e-34;ID=GmJack01g0000900-RNAi;Name=GmJack01g0000900-RNAi;Score=174.4;product=U2 spliceosomal RNA
Chr01   cmscan  snRNA   2842574 2842768 .       +       .       Dbxref=GeneID:U2 RFAM:RF00004 clan:CL00006;EValue=6e-34;ID=GmJack01g0000900-RNAi;Name=GmJack01g0000900-RNAi;Parent=GmJack01g0000900-RNAi;Score=174.
4;product=U2 spliceosomal RNA
Chr01   cmscan  gene    2843611 2843760 .       +       .       Dbxref=GeneID:U4 RFAM:RF00015 clan:CL00007;EValue=4.5e-19;ID=GmJack01g0001000-RNAi;Name=GmJack01g0001000-RNAi;Score=107.2;product=U4 spliceosomal R
NA
Chr01   cmscan  snRNA   2843611 2843760 .       +       .       Dbxref=GeneID:U4 RFAM:RF00015 clan:CL00007;EValue=4.5e-19;ID=GmJack01g0001000-RNAi;Name=GmJack01g0001000-RNAi;Parent=GmJack01g0001000-RNAi;Score=10
7.2;product=U4 spliceosomal RNA
```

Thanks