python3

2019年2月11日 14:27

目录

==========

计算

/在python2中是整除，整数的话返回整数，直接去掉小数部分；python3中则会返回float。要实现整除，用//

```
print(8/5) # 1.6
print(8//5) # 1
```

命令行中，上一个计算结果存在_变量中，should be treated as read-only

comparisons can be chained. a < b == c检查的是a<b and b==c

[目录](#)

==========

string

如果不希望\被翻译成其他字符，可以用raw strings，即在字符串前面加r

'''和"""里面可以加\，并不显示，不加的话首尾会有空行，\n\t这种正常转义。

```
print("""\
Usage: thingy [OPTIONS]
-h Display this usage message
-H hostname Hostname to connect to\
""")
```

多个string literals(引号包围的)(放在一起时自动连接，如'Py' 'thon'，不论在命令行还是print()里都自动变为'Python'，但字符串literal和字符串变量不能自然连接。

常用函数：

str.upper()	str.lower()	str.replace('a', 'b')	str.count('a')

format

"{0} is {1}".format("Python", "fun!")

目录

==========

List

对于list，letters[2:5] = []可以删掉这部分元素。

x = int(input("Please enter an integer: "))

```
words = ['cat', 'window', 'defenestrate']
# 如果是words，会无限循环，所以用切片的方式先制造一个copy
for w in words[:]:
    if len(w) > 6:
        words.insert(0, w)
print(words) # ['defenestrate', 'cat', 'window', 'defenestrate']
```

enumerate返回的是iterator，每个里面是index+value，其实可以转化为list。

eg1: [idx for idx, e in enumerate(s) if e==1]

eg2:

```
words = ['cat', 'window', 'defenestrate']
print(list(enumerate(words))) # [(0, 'cat'), (1, 'window'), (2, 'defenestrate')]
```

range返回的是一个对象，所以直接print看到的并不是list，要先转为list再print。

print(range(10)) # range(0, 10)

常用函数：

list.remove(x)

list.pop([i])可以指定位置

list.count(x)

list.sort(key=None,reverse=False)这里的key可以是lamda函数，表明是对哪个进行排序。默认升序

list.reverse()是in-place的

del a[2:4] # 删除一段元素

del a[:] # a变为[]

del a # 再找a就error了。变量没了

For all mutable data structures in python, insert/remove/sort return default None.->principle

list comprehension: brackets containing an expression followed by a for clause, then 0 or more for or if clauses.

```
vec = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
[num for elem in vec for num in elem]
```

目录

==========

tuples/sets

tuple是逗号分隔的一组值，带不带括号都是，不可修改。
初始化空tuple：
empty = ()
初始化单个元素的tuple，末尾加逗号：
singleton = 'hello',

set是用{}包裹的一组无重复元素。
创建空set要用set()，因为{}是创建空字典的。
可以用string创建set，如a = set('helloeverybody')，常用操作有-（减）|（或）&（且）^（异或，不同时在两个集合的元素）
a = {x for x in 'hello' if x not in 'abcde'} #这里是创建set，注意set和dict是完全不同的，虽然都是{}包裹

deque:
常用函数
没有empty()这个函数！！！必须自己计算数量！！！
添加数值：append(x) appendleft(x) extend(iterable) extendleft(iterable) insert(i, x)
删除数值：pop() popleft() 这俩都会返回top值。remove(value)去掉第一次出现的地方。clear()
计算数目：count(x)
查找index：index(x[, start[, stop]])默认start=0,stop=len-1，指定范围仍然返回相对于full-region的index
方向：reverse()反转in-place，rotate(n=1)
maxlen

目录

==========

循环

可以用reversed()来逆转顺序
for i in reversed(range(1, 10, 2)):
    ...
loop的else是没有break的情况下执行的

目录

==========

函数

print()打空行

如果一个函数中没有return，实际还是会返回一个值，None

关于缺省参数的例子：

```
def ask_ok(prompt, retries=4, reminder='Please try again!'):
    while True:
        ok = input(prompt)
        if ok in ('y', 'ye', 'yes'): return True
        if ok in ('n', 'no', 'nop', 'nope'): return False
        retries = retries - 1
        if retries < 0: raise ValueError('invalid user response')
        print(reminder)

try:
    ask_ok("Input: ", 2)
except ValueError as error:
    print(error)
```

如果缺省值使用变量值指定的，那么只绑定一次。所以下面打印出5

```
i = 5
def f(arg=i):
    print(arg)
i = 6
f()
```

但是如果指定缺省值的变量是可变对象，如list，dictionary or instances of most classes，那么缺省值是会积累的，如下：

```
def f(a, L=[]):
    L.append(a)
    return L

print(f(1))
print(f(2))
print(f(3))
```

结果是：

```
[1]
[1, 2]
[1, 2, 3]
```

改成这样就可以了：

```
def f(a, L=None):
    if L is None:
        L = []
    L.append(a)
    return L

print(f(1))
print(f(2))
print(f(3))
```

结果是：

```
[1]
[2]
[3]
```

arguments表示接受变长无名参数，存在名为arguments的list中；keywords表示接受变长有名参数，并存在同名字典中。arguments是单星且要放在keywords前。

```
def cheeseshop(kind, *arguments, **keywords):
   print("-- Do you have any", kind, "?")
   print("-- I'm sorry, we're all out of", kind)
   for arg in arguments:
      print(arg)
   print("-" * 10)
   for kw in keywords:
      print(kw, ":", keywords[kw])
```

所以这里会把传进去的list和dict看作是arguments，'cheda'仍然看作是kind，keywords当前则是空的。

```
cheeseshop("cheda", ['larma', 'goat', 'mimikee'], {'larma':10, 'goat':20, 'mimikee':25, 'panda':100})
```

这里几个string literals是传给arguments了，shopkeeper这些则会传给keywords。

```
cheeseshop("Limburger", "It's very runny, sir.",\
"It's really very, VERY runny, sir.",\
shopkeeper="Michael Palin",\
client="John Cleese",\
sketch="Cheese Shop Sketch")
```

可变参数使用举例：

```
def write_multiple_items(file, separator, *args):
    file.write(separator.join(args))
def concat(*args, sep="/"):
   return sep.join(args)

>>> list(range(3, 6)) # normal call with separate arguments
[3, 4, 5]
>>> args = [3, 6]
>>> list(range(*args)) # call with arguments unpacked from a list
[3, 4, 5]
```

*是把list打包成传递给函数的可变参数，**是把dict打包成可变参数

```
>>> def parrot(voltage, state='a stiff', action='voom'):
... print("-- This parrot wouldn't", action, end=' ')
... print("if you put", voltage, "volts through it.", end=' ')
... print("E's", state, "!")
...
>>> d = {"voltage": "four million", "state": "bleedin' demised", "action": "VOOM"}
>>> parrot(**d)
-- This parrot wouldn't VOOM if you put four million volts through it. E's bleedin' demised !
```

lambda函数 lambda para: returnVal

```
>>> pairs = [(1, 'one'), (2, 'two'), (3, 'three'), (4, 'four')]
>>> pairs.sort(key=lambda pair: pair[1])
>>> pairs
[(4, 'four'), (1, 'one'), (3, 'three'), (2, 'two')]
```

目录

==========

错误处理

raise是主动抛出错误，中断时可以在Traceback中看到Exception。换个名字也行。

```
>>> x = 10
>>> if x > 5:
...     raise ValueError('{} should not exceed 5.'.format(x))
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
ValueError: 10 should not exceed 5.
>>>
```

assert则是条件正确时**不**中断，否则中断时可以在Traceback看到AssertionError,如本来不报错的语句在win下执行就报错。

python

```
(base) C:\windows\system32>python
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> assert ('linux' in sys.platform), "This code runs on Linux only."
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AssertionError: This code runs on Linux only.
>>>
```

```
def linux_interaction():
    assert ('linux' in sys.platform), "Function can only run on Linux systems."
    print('Doing something.')
```

try里放的是可能出问题的程序段，except处理的是对应的问题（尽量避免bare except），else是没有异常时执行的程序段，里面仍然可以嵌套try-except，finally里放的是无论如何都会执行的。

```
try:
    linux_interaction()
except AssertionError as error:
    print(error)
else:
    try:
        with open('file.log') as file:
            read_data = file.read()
    except FileNotFoundError as fnf_error:
        print(fnf_error)
    print('Executing the else clause.')
finally:
    print('Cleaning up, irrespective of any exceptions.')
```

documentation strings:__doc__

三引号包围，第一行简述purpose，如果是多行，第二行空行，第三行开始是多段，描述调用规则等。每行首字母开头，句号结束。三引号之内的indentation会原样打印，使用例子：

```
>>> def my_function():
... """Do nothing, but document it.
...
... No, really, it doesn't do anything.
... """
... pass
```

```
...
>>> print(my_function.__doc__)
Do nothing, but document it.
    No, really, it doesn't do anything.
```

函数的annotation: 把参数/返回值等存在名为__annotation__的字典中，对函数其他部分没有影响。例子：

```
>>> def f(ham: str, eggs: str = 'eggs') -> str:
... print("Annotations:", f.__annotations__)
... print("Arguments:", ham, eggs)
... return ham + ' and ' + eggs

>>> f('spam')
Annotations: {'ham': <class 'str'>, 'return': <class 'str'>, 'eggs': <class 'str'>}
Arguments: spam eggs
'spam and eggs'
```

目录

==========

coding style

不要用 tab

每行不超过79个字符

尽量单行注释

使用docstrings

class: CamelCase，functions&methods: lower_case_with_underscore

目录

==========

Modules

from fibo import *

这样可以导入除_开头的所有名字，但是不赞成使用这个是担心重名，破坏。

一个模块只能import一次，重新import的话用：

```
import importlib
importlib.reload(modulename)
```

目录

PostgreSQL

2019年1月15日 9:38

安装

sudo apt-get install postgresql postgresql-contrib

PostgreSQL默认创建一个linux用户postgres，改linux密码：

sudo passwd postgres

修改PostgreSQL用户的密码：

su - postgres

psql -d template1 -c "ALTER USER postgres WITH PASSWORD 'pxy7896';"

区别：linux用户是access数据库的，PostgreSQL用户是在数据库上执行administrative tasks

登入：

su - postgres

psql

```
postgres@pangxiaoyi-VirtualBox:~$ psql
psql (10.3 (Ubuntu 10.3-1))
Type "help" for help.

postgres=# help
You are using psql, the command-line interface to PostgreSQL.
Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit
postgres=#
```

退出：\q

退出postgres@pangxiaoyi-VirtualBox:~$，用ctrl+D或者exit也可以

导出数据库

pg_dump --host [**地址**] --port [**端口**] --username [**数据库的用户名**] > [**导出的文件**] [**数据库名字**]

os.system("pg_dump --host localhost --username postgres > test.sql suppliers")

导入数据库

psql -d [**数据库名字**] -f [**文件名**] [**用户名**]

创建PostgreSQL Roles

Roles apply globally，对多个数据库都是有效的

在postgres账户下（linux prompt）：

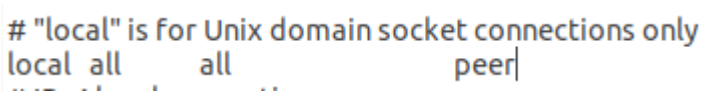
createuser examplerole --pwprompt

然后按提示输入密码

psql testdb进入数据库后，给examplerole所有权限

GRANT ALL ON employees TO examplerole;

删除role:
dropuser
因为原本默认的是peer authentication，也就是登入登出不需要密码，要改这个，可以sudo打开/etc/postgresql/10/main下的pg_hba.conf，然后将peer改成md5



修改完之后要重新开启postgresql服务：
sudo service postgresql restart
再登入的时候就要输密码了。
su postgres
psql -U examplerole -W testdb

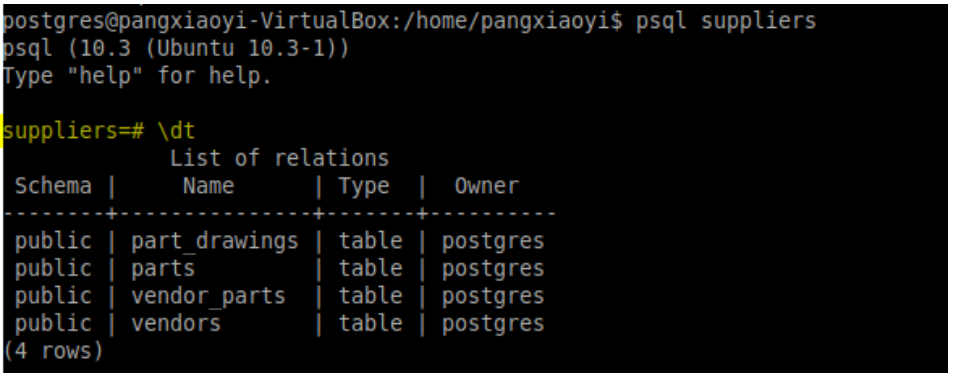
在table名称后加\z查看权限
employees\z

在没进数据库之前创建一个数据库：
createdb testdb
连接到该数据库：
psql testdb
或者进入数据库后也可以创建数据库（数据库名称自动小写）：
CREATE DATABASE probes;
删除数据库：（无法删除当前打开的数据库）
DROP DATABASE IF EXISTS postgres;

创建表格：
CREATE TABLE employees (employee_id int, first_name varchar, last_name varchar);
注意：表名会自动转小写
插入一条记录：
INSERT INTO employees VALUES (1, 'John', 'Doe');
查询结果：
SELECT * from employees;

在python里使用：
import psycopg2

限制
ON UPDATE CASCADE ON DELETE CASCADE
RESTRICT prevents deletion of a referenced row
NO ACTION means that if any referencing rows still exist when the constraint is checked, an error is raised; this is the default behavior if you do not specify anything.
区别是NO ACTION允许后续操作时再检查
CASCADE 指这条记录被删掉后，所有引用它的也会删掉，内容会被填充为SET NULL and SET DEFAULT. 但是如果default值违反了其他约束，则不会生效。
ON UPDATE是值改变时引发的，同样有CASCADE限制，就是级联。



一般过程，先开一个connection，然后再用cursor进行业务处理。在关闭connection之前，如果要保存所有的更改，用commit()，如果不想保存，使用rollback()，或者用del作为隐式回滚。因为select有可能造成table bloat / locks，所以尽量在关闭之前都rollback()或commit()。
从psycopg2.5开始，支持with，就是没有任何exception的话，自动commit，否则rollback。但是不会自动关connection，要手动关一下：

```
with psycopg2.connect(dsn) as conn:
    with conn.cursor() as cur:
        cur.execute(sql)
```

或者：

```
conn = psycopg2.connect(dsn)
# transaction 1
with conn:
    with conn.cursor() as cur:
        cur.execute(sql)
# transaction 2
with conn:
    with conn.cursor() as cur:
        cur.execute(sql)
conn.close()
```

来自 <http://www.postgresqltutorial.com/postgresql-python/transaction/>

table bloat：表格膨胀是因为使用delete或者update时，都包含删除操作（update是先添加再删除），这个操作不是立即执行的，而是先把这个数据设为unavailable，此时这个空间不可用。当当前page中已经没有可用行的时候，才会有一个vacuum进程把空间变为available。不过也不交还个操作系统。如果unavilable远多于好的，检查是否可用就会浪费时间，index里也有很多事无效的索引，这就是bloat。此时查询时间会变慢。
使用pg_reorg/pg_repack/cluster/vacuum full整理空间，postgresql可以查看表格和索引膨胀的情况。

如果对数据库进行一系列复杂操作，比如要加循环，有函数之类的，可以用plpgsql，

```
CREATE OR REPLACE FUNCTION get_parts_by_vendor(id integer)
  RETURNS TABLE(part_id INTEGER, part_name VARCHAR) AS
$$
BEGIN
```

```
  RETURN QUERY

SELECT parts.part_id, parts.part_name
FROM parts
INNER JOIN vendor_parts on vendor_parts.part_id = parts.part_id
WHERE vendor_id = id;

END; $$

LANGUAGE plpgsql;
```

来自 <http://www.postgresqltutorial.com/postgresql-python/call-stored-procedures/>

使用

```
cur.callproc('get_parts_by_vendor', (vendor_id,))
```

cur.rowcount 返回记录的条数
fetchone()
fetchall()
fetchmany() 指定到底要几条记录，尽量返回，可能不够

BLOB：binary large object for storing binary data in the database 可能是图片或者文档
可以用BYTEA数据类型存储这些数据
核心：

```
cur.execute("INSERT INTO part_drawings(part_id,file_extension,drawing_data) " +
                "VALUES(%s,%s,%s)",
                (part_id, file_extension, psycopg2.Binary(drawing)))
```

使用：

```
write_blob(1, 'images/simtray.jpg', 'jpg')
```

现在已经存好了内容，后缀，就可以从数据库中查到，再写出来：
核心：

```
cur.execute(""" SELECT part_name, file_extension, drawing_data
                    FROM part_drawings
                    INNER JOIN parts on parts.part_id = part_drawings.part_id
                    WHERE parts.part_id = %s """,
                (part_id,))
blob = cur.fetchone()
open(path_to_dir + blob[0] + '.' + blob[1], 'wb').write(blob[2])
```

查找：

```
SELECT * FROM "annotation" WHERE "chr" LIKE '%alt'
```

==========

示例代码：

```
# !/usr/bin/python
import psycopg2
import ConfigParser
import os
#import warnings

#warnings.filterwarnings("ignore")

def config(filename='database.ini', section='postgresql'):
    # create a parser
    parser = ConfigParser.ConfigParser()
    # read config file
    parser.read(filename)
    # get section, default to postgresql
    db = {}
    if parser.has_section(section):
        params = parser.items(section)
        for param in params:
            db[param[0]] = param[1]
    else:
        raise Exception('Section {0} not found in the {1} file'.format(section, filename))

    return db


def connect():
    """ Connect to the PostgreSQL database server """
    conn = None
    try:
        # read connection parameters
        params = config()
        # connect to the PostgreSQL server
        print('Connecting to the PostgreSQL database...')
        conn = psycopg2.connect(**params)
        # create a cursor
        cur = conn.cursor()
        # execute a statement
        print('PostgreSQL database version:')
        cur.execute('SELECT version()')
        # display the PostgreSQL database server version
        db_version = cur.fetchone()
        print(db_version)
        # close the communication with the PostgreSQL
        cur.close()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
    finally:
        if conn is not None:
            conn.close()
            print('Database connection closed.')

def create_tables():
    """ create tables in the PostgreSQL database"""
```

```
    commands = (
        """
        CREATE TABLE vendors (
            vendor_id SERIAL PRIMARY KEY,
            vendor_name VARCHAR(255) NOT NULL
        )
        """,
        """
        CREATE TABLE parts (
            part_id SERIAL PRIMARY KEY,
            part_name VARCHAR(255) NOT NULL
            )
        """,
        """
        CREATE TABLE part_drawings (
            part_id INTEGER PRIMARY KEY,
            file_extension VARCHAR(5) NOT NULL,
            drawing_data BYTEA NOT NULL,
            FOREIGN KEY (part_id)
            REFERENCES parts (part_id)
            ON UPDATE CASCADE ON DELETE CASCADE
        )
        """,
        """
        CREATE TABLE vendor_parts (
            vendor_id INTEGER NOT NULL,
            part_id INTEGER NOT NULL,
            PRIMARY KEY (vendor_id , part_id),
            FOREIGN KEY (vendor_id)
                REFERENCES vendors (vendor_id)
                ON UPDATE CASCADE ON DELETE CASCADE,
            FOREIGN KEY (part_id)
                REFERENCES parts (part_id)
                ON UPDATE CASCADE ON DELETE CASCADE
        )
        """)
    conn = None
    try:
        # read the connection parameters
        params = config()
        # connect to the PostgreSQL server
        conn = psycopg2.connect(**params)
        cur = conn.cursor()
        # create table one by one
        for command in commands:
            cur.execute(command)
        # close communication with the PostgreSQL database server
        cur.close()
        # commit the changes
        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
    finally:
        if conn is not None:
            conn.close()

def insert_vendor_list(vendor_list):
```

```
    """ insert multiple vendors into the vendors table  """
    sql = "INSERT INTO vendors(vendor_name) VALUES(%s)"
    conn = None
    try:
        # read database configuration
        params = config()
        # connect to the PostgreSQL database
        conn = psycopg2.connect(**params)
        # create a new cursor
        cur = conn.cursor()
        # execute the INSERT statement
        cur.executemany(sql,vendor_list)

        # execute single statement
        #cur.execute()
        os.system("pg_dump --host localhost --username postgres > test.sql suppliers")
        # commit the changes to the database
        conn.commit()
        # close communication with the database
        cur.close()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
    finally:
        if conn is not None:
            conn.close()


def insert_vendor(vendor_name):
    """ insert a new vendor into the vendors table """
    sql = """INSERT INTO vendors(vendor_name)
             VALUES(%s) RETURNING vendor_id;"""
    conn = None
    vendor_id = None
    try:
        # read database configuration
        params = config()
        # connect to the PostgreSQL database
        conn = psycopg2.connect(**params)
        # create a new cursor
        cur = conn.cursor()
        # execute the INSERT statement
        cur.execute(sql, (vendor_name,))
        # get the generated id back
        vendor_id = cur.fetchone()[0]
        # vendor_id = cur.fetchall()
        # commit the changes to the database
        conn.commit()
        # close communication with the database
        cur.close()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
    finally:
        if conn is not None:
            conn.close()

    return vendor_id
```

```
# modify values. UPDATE command
def update_vendor(vendor_id, vendor_name):
    """ update vendor name based on the vendor id """
    sql = """ UPDATE vendors
         SET vendor_name = %s
         WHERE vendor_id = %s"""
    conn = None
    updated_rows = 0
    try:
        # read database configuration
        params = config()
        # connect to the PostgreSQL database
        conn = psycopg2.connect(**params)
        # create a new cursor
        cur = conn.cursor()
        # execute the UPDATE  statement
        cur.execute(sql, (vendor_name, vendor_id))
        # get the number of updated rows
        updated_rows = cur.rowcount
        # Commit the changes to the database
        conn.commit()
        # Close communication with the PostgreSQL database
        cur.close()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
    finally:
        if conn is not None:
            conn.close()

    return updated_rows

def select_vendor():
    cur.execute("""
            SELECT part_name, vendor_name
            FROM parts
            INNER JOIN vendor_parts ON vendor_parts.part_id = parts.part_id
            INNER JOIN vendors ON vendors.vendor_id = vendor_parts.vendor_id
            ORDER BY part_name;
        """)
    for row in iter_row(cur, 10):
        print(row)
    sql = """
    SELECT
    """
    conn = None
    params = config()
    conn = psycopg2.connect(**params)
    cur = conn.cursor()
    cur.execute(sql)
    rows = cur.fetchall()
    print cur.rowcount
    for row in rows:
        print row
    conn.commit()
    cur.close()
    conn.close()
```

```
if __name__ == '__main__':
    # insert multiple vendors
    insert_vendor_list([
        ('AKM Semiconductor Inc.',),
        ('Asahi Glass Co Ltd.',),
        ('Daikin Industries Ltd.',),
        ('Dynacast International Inc.',),
        ('Foster Electric Co. Ltd.',),
        ('Murata Manufacturing Co. Ltd.',)
    ])

    #v_id = result_single = insert_vendor('pangxiaoyi')
    print "v_id\n" + str(v_id)
    #create_tables()
```

NCBI/UCSC Notes

2018年12月24日 9:57

数据下载：
ftp://ftp.ncbi.nih.gov/genomes/Homo_sapiens
用filezilla，浏览器打不开

前缀们：
NT_ genomic scaffold
NW_ genomic scaffold
NC_ Primary Assembly

Prefixes	Molecule	Method
NC, NG	Genomic	Curated
NM	MRNA	Curated
NR	RNA	Curated
NP	Protein	Curated
NT, NW	Genomic	Automated
XM	MRNA	Automated
XR	RNA	Automated
XP	Protein	Automated

==========UCSC

gpe格式的最后一列是exonFrames，一列逗号隔开的数，可以取{0,1,2}或是-1。
-1，代表对应的exon全部位于UTR区，不参与翻译。
{0,1,2}，代表对应的exon在参与翻译时，需要向前一个exon的末尾取n={0,1,2}个碱基，从而组成正确的读码框。这里说的前一个exon，和转录本所在链方向一致，即5'端的exon。因此，第一个coding exon的exonFrame必然是0，不因start codon在这个exon内部的位置而变。

倒数第二列cdsEndStat和倒数第三列cdsStartStat：
表示的是CDS的情况：none表示没有CDS，unk表示不知道CDS的start或者end是不是完整的，incmpl表示CDS的start/end是不完整的，cmpl表示CDS的start/end是完整的。

win10快捷键

2018年12月11日 16:36

输入法：

繁体简体切换

ctrl+shift+F

全角半角切换

shit+空格

COSMIC API

2018年8月21日 14:27

变量、方法都是小写、_连接的单词；类、包是每个单词首字母大写的词。

_开头的方法是private，不能在定义它的类外面使用。

Methods which begin with get_all or fetch_all return references to lists.

```
# Iterate through all of the genes on a clone
foreach my $gene ( @{ $first_clone->get_all_Genes() } ) {
    print $gene->stable_id(), "\n";
}
# More memory efficient way of doing the same thing
my $genes = $first_clone->get_all_Genes();
while ( my $gene = shift @{$genes} ) {
    print $gene->stable_id(), "\n";
}
```

Slice

```
# get a slice adaptor for the human core database
my $slice_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Slice' );
# Obtain a slice covering the entire chromosome X
my $slice = $slice_adaptor->fetch_by_region( 'chromosome', 'X' );
# Obtain a slice covering the entire clone AL359765.6
$slice = $slice_adaptor->fetch_by_region( 'clone', 'AL359765.6' );
# Obtain a slice covering an entire scaffold
$slice = $slice_adaptor->fetch_by_region( 'scaffold', 'KI270510.1' );
# Obtain a slice covering the region from 1MB to 2MB (inclusively) of
# chromosome 20
$slice = $slice_adaptor->fetch_by_region( 'chromosome', '20', 1e6, 2e6 );
# Ensembl ID, 长度。
#It also returns 5000bp of flanking sequence at both the 5' and 3' ends,
# fetch_by_gene_stable_id() method always returns a slice on the forward strand
even if the gene is on the reverse strand.
my $slice = $slice_adaptor->fetch_by_gene_stable_id( 'ENSG00000099889', 5e3 );
```

生物知识补遗

2018年8月29日 9:09

BED files of ucsc

3 required BED fields:

1. **chrom** - name
2. **chromStart** - 0-based index.
3. **chromEnd** - For example, the first 100 bases of a chromosome are defined as *chromStart=0, chromEnd=100*, and span the bases numbered 0-99.

   The 9 additional optional BED fields are:

4. **name** - Defines the name of the BED line. This label is displayed to the left of the BED line in the Genome Browser window when the track is open to full display mode or directly to the left of the item in pack mode.
5. **score** - A score between 0 and 1000. If the track line *useScore* attribute is set to 1 for this annotation data set, the *score* value will determine the level of gray in which this feature is displayed (higher numbers = darker gray). This table shows the Genome Browser's translation of BED score values into shades of gray:

shade									
score	≤ 166	167-277	278-388	389-499	500-611	612-722	723-833	834-944	≥ 945

6. **strand** - Defines the strand. Either "." (=no strand) or "+" or "-".
7. **thickStart** - The starting position at which the feature is drawn thickly (for example, the start codon in gene displays). When there is no thick part, thickStart and thickEnd are usually set to the chromStart position.
8. **thickEnd** - The ending position at which the

feature is drawn thickly (for example the stop codon in gene displays).

9. **itemRgb** - An RGB value of the form R,G,B (e.g. 255,0,0). If the track line *itemRgb* attribute is set to "On", this RBG value will determine the display color of the data contained in this BED line. NOTE: It is recommended that a simple color scheme (eight colors or less) be used with this attribute to avoid overwhelming the color resources of the Genome Browser and your Internet browser.
10. **blockCount** - The number of blocks (**exons**) in the BED line.
11. **blockSizes** - A comma-separated list of the block sizes. The number of items in this list should correspond to *blockCount*.
12. **blockStarts** -

    Blocks may not overlap.

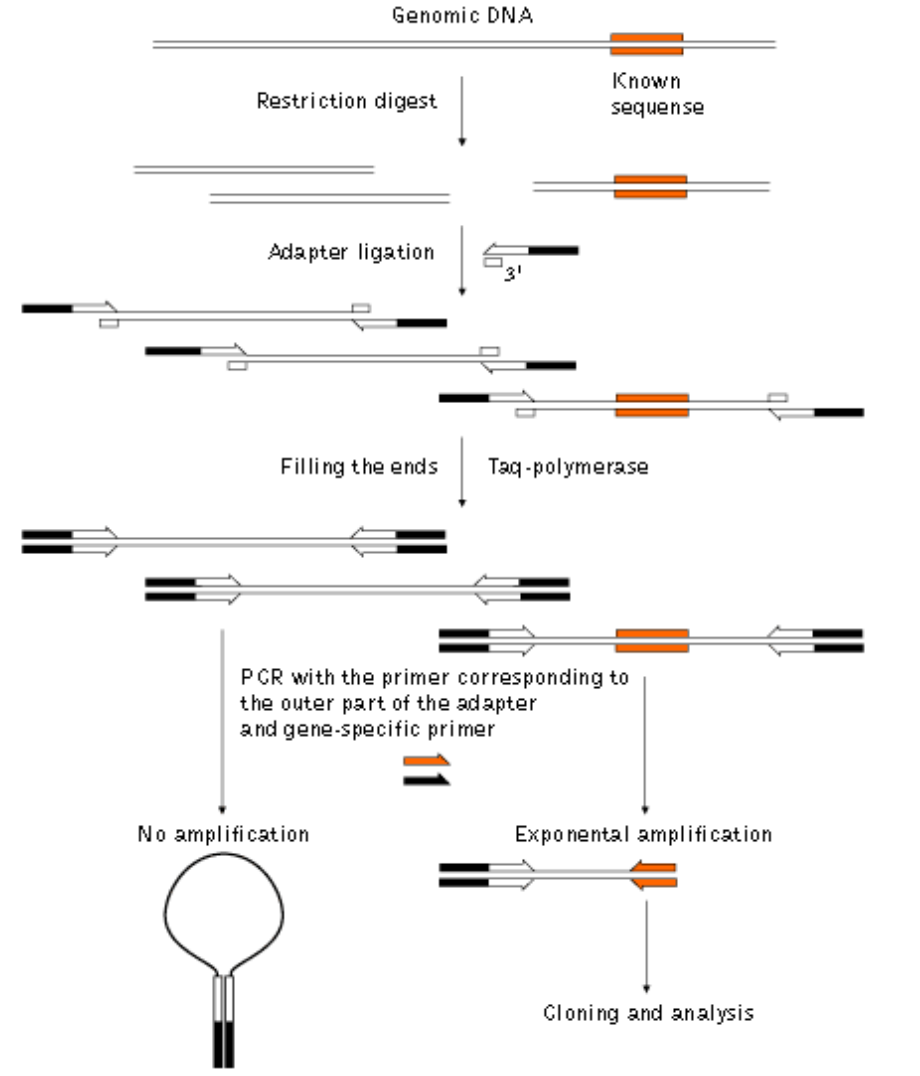
来自 <https://genome.ucsc.edu/FAQ/FAQformat.html#format1>

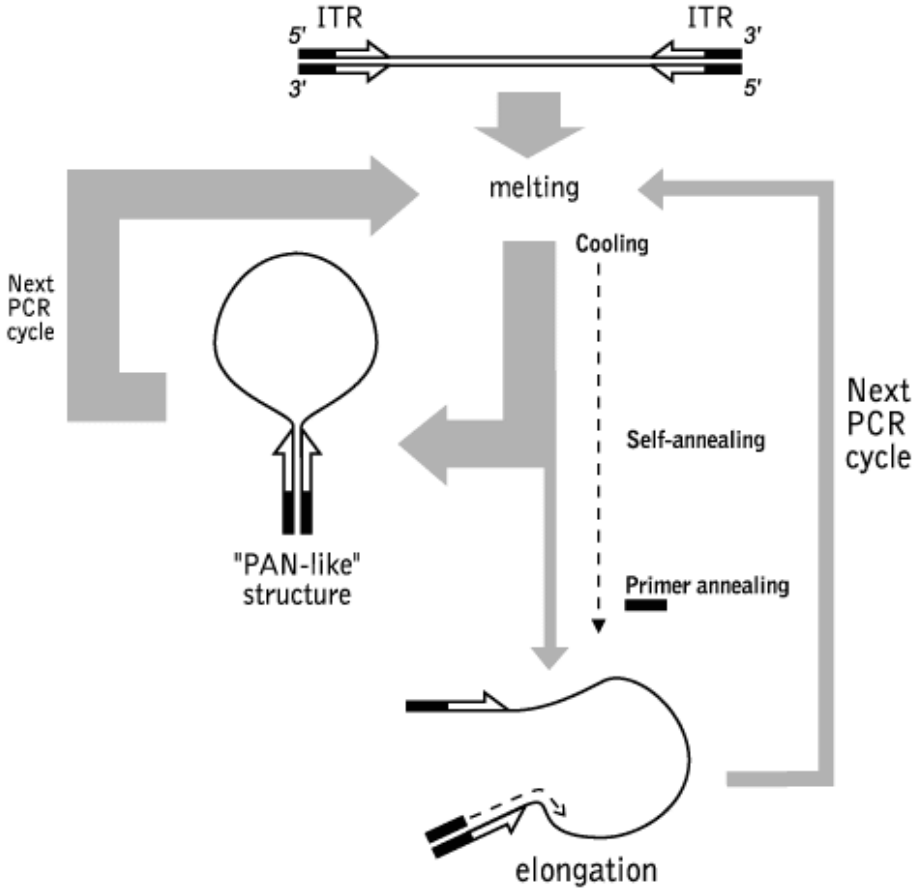
gene walking

http://evrogen.com/technologies/genome-walking.shtml

A method to isolate flanking genomic segments(e. g. promoter regions) adjacent to a known sequence.

左下的是PCR suppression by inverted terminal repeats，也就是图2


Genomic DNA
Known sequense
Restriction digest
Adapter ligation
3'
Filling the ends
Taq-polymerase
PCR with the primer corresponding to the outer part of the adapter and gene-specific primer
No amplification
Exponental amplification
Cloning and analysis


ITR
ITR
5'
3'
3'
5'
melting
Cooling
Next PCR cycle
Self-annealing
Primer annealing
"PAN-like" structure
Next PCR cycle
elongation

Signature primers are short pieces of DNA that are **only** conserved in target sequences.

exon junction span

可以设计一个引物bind到两个exon上。通过exon junction match可以设置这个引物在两个exon上占据的范围。more specific towards mRNA amplication and the primer should not amplify genomic DNA which contains introns(theoridically)

intron inclusion

勾选的话，可以判断是否有genomic DNA contamination。因为1.genomic DNA有intron，所以PCR产物会很大。2.如果intron非常大，PCR时产物可能不会延伸完全，就可以去掉。intron length range 的min可以设为200。

QPCR

Primer parameters

Change the 'Max' product size value to: 200.

Change the 'Primer melting temperatures (Tm)' to: 59 (Min); 62 (Opt); 65 (Max).

linux操作

2018年9月3日 8:58

服务器密码：
123456

窗口：
sftp://pangxiaoyi@10.1.1.25/data2/maoyibo/Share/Bam
先开窗口输入密码，然后linux下终端访问服务器
/run/user/1000/gvfs/sftp:host=
10.1.1.25,user=pangxiaoyi/data2/maoyibo/Share/Bam

从服务器下载文件
L是下载链接的
rsync -aPL pangxiaoyi@10.1.1.25:/data2/maoyibo/Share/slc2a2_total.bed .
-a, --archive archive mode; equals -rlptgoD (no -H,-A,-X)
-L, --copy-links transform symlink into referent file/dir
-P same as --partial --progress
Using the --partial option tells rsync to keep the partial file which should make a subsequent transfer of the rest of the file much faster.断点续传。
--progress show progress during transfer

不看warnings输出
python -W ignore test_psql.py

时间
显示年月日星期时分秒
date +%c
只显示时间，13:42:14
date +%T

得到file2中特有的行
awk 'NR==FNR{a[$0]}NR>FNR{ if(!($1 in a)) print $0}' query1.txt query2.txt > special.txt
得到共有的行，
awk 'NR==FNR{a[$0]++} NR>FNR&&a[$0]' 1.txt 2.txt
或者（要預先排序）
comm -12 file1.txt file2.txt > common.txt

计算某文件夹大小：按MB
du -m -sh folder1
不加sh给出的是文件夹下各文件的大小，分开的

去掉win下copy到linux的文件，每行末尾带的^M
不装dos2unix的话，用perl来过滤

```
perl -pi -e 's/\r\n/\n/g' INFILE
```

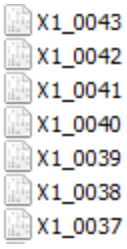
来自 <https://unix.stackexchange.com/questions/277217/how-to-install-dos2unix-on-linux-without-root-access>

统计目录下文件个数

ls -l |grep "^-"|wc -l

切开大文件

每个行数最多1500，结果文件前缀为X1_，编号为四位数字



parallel安装

(wget -O - pi.dk/3 || curl pi.dk/3/) | bash

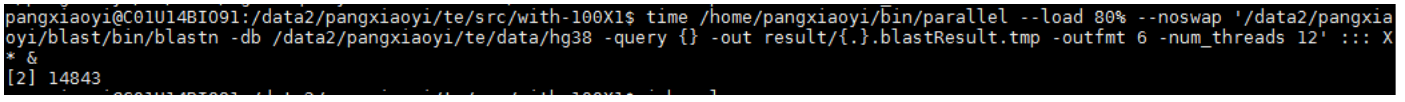
split -l 3000 *fasta -d -a 5 X_

使用parallel

time /home/pangxiaoyi/bin/parallel --load 80% --noswap '/data2/pangxiaoyi/blast/bin/blastn -db /data2/pangxiaoyi/te/data/hg38 -query {} -out result/{.}.blastResult.tmp -outfmt 6 -num_threads 12' ::: X* &
time /home/pangxiaoyi/bin/parallel --load 80% --noswap 'python run.py' ::: X_* &
cd hit
cat X*finalHitCount > finalHitCount1.txt
cp finalHitCount.txt ../../../with-100X1.finalHitCount.txt

time /home/pangxiaoyi/bin/parallel --load 80% --noswap 'python blast_merge.py' ::: X_* &

将后台job搞回前台：



用jobs -l也可以看到编号。注意这里要用的是[2]

fg 2

即可移到前台，然后ctrl+z就可以停止了

bg 2移到后台继续

$ps aux 输出结果中的STAT列的可能标志

S 睡眠。通常是在等待某个事件的发生，如一个信号或有输入可用

R 运行。 严格来说，应是“可运行” ，即在运行队列中，处于正在执行或即将运行状态

D 不可中断的睡眠（等待）。通常是在等待输入或输出完成

T （terminate）停止。通常是被shell作业控制所停止，或者进程正处于调试器的控制之下

Z （zombie)僵尸进程，通常是该进程已经死亡，但父进程没有调用wait类函数来释放该进程的资源

N （nice)低优先级任务

s 进程是会话期首进程

+ 进程属于前台进程组

l 进程是多线程的

< 高优先级任务

sort 举例

awk 举例

awk '($4>130 && $3>90){print $0}' IN > OUT

其中$4指的就是第四列

$0 就是指整行

awk '($2<$3){print $0}' final.utr.raw.bed > final.utr.bed

查看本机外网ip地址及归属地

curl ip.cn

```
pangxiaoyi@pangxiaoyi-VirtualBox:~$ curl ip.cn
当前 IP: 222.190.125.210 来自: 江苏省南京市 电信
```

网络测速

sudo pip install speedtest-cli

speedtest-cli --bytes

```
pangxiaoyi@pangxiaoyi-VirtualBox:~$ speedtest-cli --bytes
Retrieving speedtest.net configuration...
Testing from China Telecom jiangsu (222.190.125.210)...
Retrieving speedtest.net server list...
Selecting best server based on ping...
Hosted by China Unicom (Nanjing) [1.80 km]: 13.401 ms
Testing download speed................................................
.....................
Download: 0.35 Mbyte/s
Testing upload speed..................................................
..................................
Upload: 42.00 Mbyte/s
```

查看磁盘空间

df

```
# 总核数 = 物理CPU个数 X 每颗物理CPU的核数
# 总逻辑CPU数 = 物理CPU个数 X 每颗物理CPU的核数 X 超线程数
# 查看物理CPU个数
cat /proc/cpuinfo| grep "physical id"| sort| uniq| wc -l
# 查看每个物理CPU中core的个数(即核数)
cat /proc/cpuinfo| grep "cpu cores"| uniq
# 查看逻辑CPU的个数
cat /proc/cpuinfo| grep "processor"| wc -l
```

python3 控制台进不了，报错：
Error processing line 1 of /usr/lib/python3/dist-packages/zope.interface-4.3.2-nspkg.pth
进到路径下，把这个文件删了就可以了。

pip3的安装
sudo apt-get install python3-pip
升级
sudo pip3 install --upgrade pip
卸载
sudo apt-get remove python3-pip
查看版本：pip3 -V

python-docx
Python 2.6, 2.7, 3.3, or 3.4
lxml >= 2.3.2
始终装不上，python2.7.13下lxml会报错：（使用pip安装）
```
>>> import lxml
>>> from lxml import etree
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: cannot import name etree
```

控制台输出重定向：
0、1和2分别表示标准输入、标准输出和标准错误信息输出
/dev/null相当于无底洞，重定向到它的信息都会消失，所以如果不想看标准输出和标准错误信息输出，则可以：
ls 1>/dev/null 2>/dev/null
ls -lg >! list 将执行“ls - lg” 命令的结果覆盖写入文件list 中。
ls - lag >> list 将执行“ls - lag” 命令的结果附加到文件list 中。
cc file2.c >>& error 将编译file2.c 文件时屏幕所产生的任何信息附加到文件error 中。

man command
调出command的用户手册

命令行历史：
模糊查询
history|grep -i "XXX"

让history有时间戳显示
export HISTTIMEFORMAT='%F %T '

清除history记录
history -c

搜索历史命令

Control+R，然后输入过往命令的key word

进入root模式

sudo su

切回用户模式

exit或者ctrl + d

列出当前目录下所有以i开头的文件

ls -l i*

文件更名或者剪切

mv

只要是文件，不管是否隐藏，或是文件使用权限设置成只读，rm皆可删除，在此要注意的是已删除的文件是无法恢复的

列举当前目录及其子目录下所有扩展文件名是c的文件

find . -name "*.c"

在文件中查找字符串

grep 字符串 文件名

安全地关闭或重启linux系统

shutdown

一次以一个page显示文件

more 文件名

当看到屏幕左下有“--more--”时，回车显示下一行，空格显示下一个page

less 文件名

除了空格和回车，按b可以上翻到前一个page

逐级进入指定目录的每一个子目录，并显示该目录占用文件系统数据块的情况

du 目录名

显示磁盘使用情况

df

显示进程

pa aux

无条件删除进程号为pid#的进程

kill -9 PID#

---

vim操作

hjkl分别为左下上右

ctrl+b 上滚一屏	ctrl+f 下滚一屏
ctrl+u 上滚半屏	ctrl+d 下滚半屏

G 移到文件最后（shift+G）

w 移到下个字开头 b 移到上个字开头
删除当前光标所在后面#个字符
#x
如果是删除当前光标所在后面一个字符，x即可
删除当前光标所在后面#行，如果只删一行dd即可
#dd
删当前光标的左字符
X
删除至行尾
D
复原到上一个操作
u
列出行号及相关信息
g

三、更改

cw： 更改光标处的字到此单字的字尾处。

c#w： 例如，c3w表示更改3个字。

cc： 修改行。

C： 替换到行尾。

四、取代

r： 取代光标处的字符。

R： 取代字符直到按ESC为止。

五、复制

yw： 拷贝光标处的字到字尾至缓冲区。

P： 把缓冲区的资料贴上来。

yy： 拷贝光标所在之行至缓冲区。

#yy： 例如,5yy，拷贝光标所在之处以下5行至缓冲区。

PEP8

2018年9月3日 11:17

https://www.python.org/dev/peps/pep-0008/

不遵守pep8的情况：与当前项目保持一致；历史原因；向后兼容；不遵守更易读。

Code lay-out

4 spaces per indentation level. 避免用tab，也避免tab和空格混合

每行最多79个字符，换行可以在()[]{}内，用反斜杠\

类定义和top-level函数之间隔2行

类中定义方法时用1个空行隔开

有关的单行函数定义，空行可以省略

Files using ASCII (in Python 2) or UTF-8 (in Python 3) should not have an encoding declaration.

using \x, \u, \U, or \N escapes is the preferred way to include non-ASCII data in string literals

```
# More indentation included to distinguish this from the rest.
def long_function_name(
        var_one, var_two, var_three,
        var_four):
    print(var_one)

# Hanging indents should add a level.
foo = long_function_name(
    var_one, var_two,
    var_three, var_four)
```

if 多行的情况

```
# Add a comment, which will provide some distinction in editors
# supporting syntax highlighting.
if (this_is_one_thing and
    that_is_another_thing):
    # Since both conditions are true, we can frobnicate.
    do_something()
```

列表多行

```
my_list = [
    1, 2, 3,
    4, 5, 6,
]
```

运算符不能放在行尾

```
# Yes: easy to match operators with operands
income = (gross_wages
          + taxable_interest
          + (dividends - qualified_dividends)
```

- ira_deduction
- student_loan_interest)

文档编排：

1.每个库单独import

2.顺序：module comments & docstring ---> import ---> globals & constants --->其他定义

3.import的顺序：standard library ---> related 3rd party --> local app/lib，组间空行分隔

4.最好一直用绝对路径import

5.所有public modules/functions/classes/methods都要写docstrings，non-public可以不写，但是def行之后要写一下注释

第一个三引号不要单独一行

```
"""This is the example module.

This module does stuff.
"""
# __future__模块，把下一个新版本的特性导入到当前版本
from __future__ import barry_as_FLUFL
# __XX__这种要写在import前面
__all__ = ['a', 'b', 'c']
__version__ = '0.1'
__author__ = 'Cardinal Biggles'

import os
import sys
```

避免空格的情况：

```
foo = (0,)
ham[lower+offset : upper+offset]
```

如果一条语句中有多个运算符，在最低优先级的那个运算符左右留空格，如

```
hypot2 = x*x + y*y
c = (a+b) * (a-b)
```

而不要在所有运算符左右都留空格

函数的默认参数，或者indicate a keyword argument，=左右不加空格

```
def complex(real, imag=0.0):
    return magic(r=real, i=imag)
```

如果默认参数有annotation，=左右要加空格

```
def munge(input: AnyStr, sep: AnyStr = None, limit=1000): ...
```

if/for/while不要写成单行形式，保持结构

```
FILES = [
    'setup.cfg',
    'tox.ini',
    ]
```

不要写成单行的FILES = ['setup.cfg', 'tox.ini',]

注释

英文，首字母大写，句后又结束符，句号后空2个格，再开始下句。短语可以省略句号。#和一个空格开始

块注释

```
# Description : Module config.
#
# Input : None
#
# Output : None
```

尽量避免单行注释。至少与代码隔开2个空格

命名规范

避免单独用小写的 l ，大写的O和大写的 i

模块命名：短，小写，下划线ok

包命名：短，小写，避免下划线

如果有扩展模块是c/c++写的，则模块名前面要加一个下划线

类命名：Capwords

异常：CapWords+Error

模块内部使用的类：_CapWords

全局变量尽量只在模块内有效，实现时：前缀一个下划线，all机制

Setting __all__ to an empty list indicates that the module has no public API.
__all__:放置在__init__.py里面,执行代码 from foldername import * 的时候,实际导入的包由__all__决定.考虑以下情况:

```
folder/
        __init__.py
        a.py
        b.py

#__init__.py
__all__=['a']

#b.py
valueb=12

#test1.py
from folder import *
#此处会报错
print b.valueb

#test2.py
from folder import a,b
print b.valueb
```

函数和变量命名：小写+下划线

常量：全部大写+下划线

公有属性前面不能用下划线

类的属性如果与关键字名字冲突，可以在后缀加下划线

类的方法第一个参数必须是self，静态方法第一个参数必须是cls

non-public methods和instance variables用前缀下划线

为避免与子类属性命名冲突，在类的一些属性（不希望子类继承的）前，前缀2条下划线

Comparisons to singletons like None should always be done with is or is not, never the equality operators.
Always use a def statement instead of an assignment statement that binds a lambda expression directly to an identifier.

Derive exceptions from Exception rather than BaseException.

异常处理这一块可以再翻文档

When raising an exception in Python 2, use raise ValueError('message')

Be consistent in return statements. Either all return statements in a function should return an expression, or none of them should.

Use string methods instead of the string module.
Use ''.startswith() and ''.endswith() instead of string slicing to check for prefixes or suffixes.
Yes: if foo.startswith('bar'):
No: if foo[:3] == 'bar':

Object type comparisons should always use isinstance() instead of comparing types directly.
Yes: if isinstance(obj, int):
No: if type(obj) is type(1):

For sequences, (strings, lists, tuples), use the fact that empty sequences are false.

Don't compare boolean values to True or False using ==.

blastn使用

2018年9月7日 9:05

BLASTn tabular output format 6

Column headers:

qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore

1.	qseqid	query (e.g., gene) sequence id
2.	sseqid	subject (e.g., reference genome) sequence id
3.	pident	percentage of identical matches
4.	length	alignment length
5.	mismatch	number of mismatches
6.	gapopen	number of gap openings
7.	qstart	start of alignment in query
8.	qend	end of alignment in query
9.	sstart	start of alignment in subject
10.	send	end of alignment in subject
11.	evalue	expect value
12.	bitscore	**bit score**

The bit score, S', is derived from the raw alignment score, S, taking the statistical properties of the scoring system into account. Because bit scores are normalized with respect to the scoring system, they can be used to compare alignment scores from different searches.

来自 <https://www.ncbi.nlm.nih.gov/books/NBK62051/>

在线工具使用：
多条序列，上传文件，下载multiple json，然后用脚本解析

bedtools getfasta -fo target_seq.fasta -fi hg38.fa -bed 6842_exons_extend.bed
makeblastdb -in target_seq.fasta -dbtype nucl -parse_seqids -out target_seq
blastn -query primer_seq.txt -db target_seq -out target_seq_result.txt -task blastn-short -max_target_seqs 1 -evalue 0.5 -outfmt 6

bedtools getfasta -fo target_seq_1.fasta -fi hg38.fa -bed exons_extend.bed
makeblastdb -in target_seq_1.fasta -dbtype nucl -parse_seqids -out test/target_seq_1
blastn -query primer_seq.txt -db test/target_seq_1 -out target_seq_result_1.txt -task blastn-short -max_target_seqs 1 -evalue 0.5 -outfmt 6

makeblastdb -in hg38refMrna.fa -dbtype nucl -parse_seqids -out test/hg38refMrna
blastn -query primer_seq.txt -db test/hg38refMrna -out hg38_result.txt -task blastn-short -dust no -soft_masking false -evalue 0.5 -outfmt 6

blastn -query test/primer_18.fasta -db test/wang -dust no -task blastn-short -evalue 0.5 -soft_masking false -out 18_result.txt -outfmt 6

bedtools getfasta -fo target_seq.fasta -fi hg38.fa -bed target.bed

makeblastdb -in hg38.fa -dbtype nucl -parse_seqids -out hg38
blastn -query test0927.fasta -db hg38 -out test0927.result.txt -evalue 0.5 -outfmt 6

From Sheng Xia
"/app_data/program/blast+/bin/blastn -query %s -db refMrna.fa -out %d.out -outfmt 6 -num_threads 12 -task 'blastn-short' -dust no"

用的db是refMrna.fa，一个200多Mb的fasta，好像也是从ucsc下载的

dust: 过滤low complexity regions from nucleic acid sequences
These include homopolymeric runs, short-period repeats, and subtler over representation of one or a few residues.

-dust for masking query
-db_soft_mask and -db_soft_mask for masking the database
removal of repeated or low complexity regions
-soft_masking
 Apply filtering locations as soft masks (i.e., only for finding initial matches).

disable all the sequence masking:
blastn -query query.fa -db db.fa **-dust no -soft_masking false** -outmft 6

blastn -query primer_seq.txt -db test/hg38refMrna -out hg38_result.txt -dust no -evalue 0.5 -soft_masking false -outfmt 6

TTTGCGGCAGACCAGGCA
TGGCACAGTCTACAAGGGCA
CCGCAGCCGCAGGTGGAA
ACTCGGTGGCAGTCACCA
CAGGGTAGCTGCTCCAGCA

GRCh38/hg38

no match in ncbi refseq	no match in GENCODE v24
VEGFD	AFDN
PTPA	COQ8A
CENPX	PRUNE1
AFDN	MIGA2
COQ8A	
PRUNE1	
METTL20	

KNL1 LARGE1 MIGA2 SLMO2	

GENCODE v24 for

VEGFD

PTPA

CENPX

METTL20

KNL1

LARGE1

SLMO2

BLAST 2.7.1+

blastn [-h] [-help] [-version]

输入的查找选项

-query <infile>

-quey_loc <start-stop> 1-based offsets, 查询序列的位置

-strand <String, 'both', 'minus', 'plus'>查询的链，默认是both

一般搜索选项

-**task** <String, 'blastn', 'blastn-short', 'dc-megablast', 'megablast', 'rmblastn'>默认是megablast

blastn	Traditional BLASTN **word_size=11, length of initial exact match**
blastn-short	BLASTN program optimized for sequences shorter than **50** bases **word_size=7, length of initial exact match**
megablast	Traditional megablast used to find very similar (e.g., intraspecies or closely related species) sequences **word_size=28, length of initial exact match**
dc-megablast	Discontiguous megablast used to find more distant (e.g., interspecies) sequences.**word_size=11, length of initial exact match**

-db <String>数据库名称

-out <outfile>

-evalue <Real> 默认是10。score表征相似度。evalue表征s值可靠性，指在随机情况下，其它序列与目标序列相似度大于s值得可能性，分越低越好。

局限性：1. 当目标序列过小时，e值会偏大，因为无法得到较高的s值；2.当两序列同源性虽高，但有较大的gap时，s值会下降，此时gap scores有用；3. 有些序列的非功能区有较低的随机性，这可能造成两序列有较高的同源性。

应用：e值适合于有一定长度且复杂度不能太低的序列。

当e值小于1e-5时，表明两序列有较高同源性，而不是因为计算错误；当e值小于1e-6时，表面同源性非常高。

-word_size <Integer, >=4> length of best perfect match

-gapopen <Integer> cost to open a gap

-gapextend <Integer> cost to extend a gap
-penalty <Integer, <=0> mismatch penalty
-reward <Integer, >=0> reward for matching
-outfmt format <String>
常用6 = Tabular, 7 = Tabular with comment lines, 8 = Seqalign (Text ASN.1), 10 = Comma-separated values, 17 = Sequence Alignment/Map (SAM)
The supported format specifiers for options 6, 7 and 10 are:

qlen	Query sequence length
sseqid	Subject Seq-id
sallseqid	All subject Seq-id(s), separated by a ';'
qstart	Start of alignment in query
sstart	Start of alignment in subject
qseq	Aligned part of query sequence
sseq	Aligned part of subject sequence
score	Raw score
pident	Percentage of identical matches
nident	Number of identical matches
positive	Number of positive-scoring matches
ppos	Percentage of positive-scoring matches
frames	Query and subject frames separated by a '/'
btop	Blast traceback operations (BTOP)
staxid	Subject Taxonomy ID
ssciname	Subject Scientific Name
qcovs	Query Coverage Per Subject
for option 17	SQ means Include Sequence Data SR means Subject as Reference Seq

BLAST-2-Sequences 选项
[-subject subject_input_file][-subject_loc range]
-num_alignments <Integer, >=0> number of db seq to show alignments for.默认250
*** Query filtering options
-dust <String>
  Filter query sequence with DUST (Format: 'yes', 'level window linker', or
  'no' to disable)
  Default = `20 64 1'
-filtering_db <String>
  BLAST database containing filtering elements (i.e.: repeats)
-window_masker_taxid <Integer>
  Enable WindowMasker filtering using a Taxonomic ID
-window_masker_db <String>
  Enable WindowMasker filtering using this repeats database.
-soft_masking <Boolean>
  Apply filtering locations as soft masks
  Default = `true'
-lcase_masking
  Use lower case filtering in query and subject sequence(s)?

*** Restrict search or results
-gilist <String>

Restrict search of database to list of GI's
* Incompatible with: negative_gilist, seqidlist, negative_seqidlist,
remote, subject, subject_loc
-seqidlist <String>
Restrict search of database to list of SeqId's
* Incompatible with: gilist, negative_gilist, negative_seqidlist, remote,
subject, subject_loc
-negative_gilist <String>
Restrict search of database to everything except the listed GIs
* Incompatible with: gilist, seqidlist, remote, subject, subject_loc
-negative_seqidlist <String>
Restrict search of database to everything except the listed SeqIDs
* Incompatible with: gilist, seqidlist, remote, subject, subject_loc
-entrez_query <String>
Restrict search with the given Entrez query
* Requires: remote
-db_soft_mask <String>
Filtering algorithm ID to apply to the BLAST database as soft masking
* Incompatible with: db_hard_mask, subject, subject_loc
-db_hard_mask <String>
Filtering algorithm ID to apply to the BLAST database as hard masking
* Incompatible with: db_soft_mask, subject, subject_loc
-perc_identity <Real, 0..100>
-qcov_hsp_perc <Real, 0..100>
Percent query coverage per hsp
-max_hsps <Integer, >=1>
Set maximum number of HSPs per subject sequence to save for each query
-culling_limit <Integer, >=0>
If the query range of a hit is enveloped by that of at least this many
higher-scoring hits, delete the hit
* Incompatible with: best_hit_overhang, best_hit_score_edge
-best_hit_overhang <Real, (>0 and <0.5)>
Best Hit algorithm overhang value (recommended value: 0.1)
* Incompatible with: culling_limit
<Real, (>0 and <0.5)>
Best Hit algorithm score edge value (recommended value: 0.1)
* Incompatible with: culling_limit
-max_target_seqs <Integer, >=1>
Maximum number of aligned sequences to keep
Not applicable for outfmt <= 4
Default = `500'
* Incompatible with: num_descriptions, num_alignments

*** Discontiguous MegaBLAST options
-template_type <String, `coding', `coding_and_optimal', `optimal'>
Discontiguous MegaBLAST template type
* Requires: template_length
-template_length <Integer, Permissible values: '16' '18' '21' >
Discontiguous MegaBLAST template length
* Requires: template_type

*** Statistical options
-dbsize <Int8>
Effective length of the database
-searchsp <Int8, >=0>
Effective length of the search space
-sum_stats <Boolean>

Use sum statistics

*** Search strategy options
-import_search_strategy <File_In>
Search strategy to use
* Incompatible with: export_search_strategy
-export_search_strategy <File_Out>
File name to record the search strategy used
* Incompatible with: import_search_strategy

*** Extension options
-xdrop_ungap <Real>
X-dropoff value (in bits) for ungapped extensions
-xdrop_gap <Real>
X-dropoff value (in bits) for preliminary gapped extensions
-xdrop_gap_final <Real>
X-dropoff value (in bits) for final gapped alignment
-no_greedy
Use non-greedy dynamic programming extension
-min_raw_gapped_score <Integer>
Minimum raw gapped score to keep an alignment in the preliminary gapped and traceback stages
-ungapped
Perform ungapped alignment only?
-window_size <Integer, >=0>
Multiple hits window size, use 0 to specify 1-hit algorithm
-off_diagonal_range <Integer, >=0>
Number of off-diagonals to search for the 2nd hit, use 0 to turn off
Default = `0'

*** Miscellaneous options
-parse_deflines
Should the query and subject defline(s) be parsed?
-num_threads <Integer, (>=1 and =<1)>
Number of threads (CPUs) to use in the BLAST search
Default = `1'
* Incompatible with: remote
-remote
Execute search remotely?
* Incompatible with: gilist, seqidlist, negative_gilist, negative_seqidlist, subject_loc, num_threads

excel操作

2018年10月3日 9:03

快捷键们

1. shift+箭头
   将选定区域扩展一个单元格宽度
2. ctrl+shift+箭头
   将选定区域扩展到与当前单元格同一行或同一列的最后一个非空白单元格
3. shift+home
   将选定区域扩展到行首
4. ctrl+shift+home
   将选定区域扩展到工作表的开始（A1）
5. ctrl+shift+end
   将选定区域扩展到表最后一个使用的单元格
6. shift+pagedown/pageup
   将选定区域向上/下扩展一屏
7. ctrl+*
   选定有效工作区
8. 自动填充
   选中要填充的单元格，输入公式，ctrl+enter

数据有效性

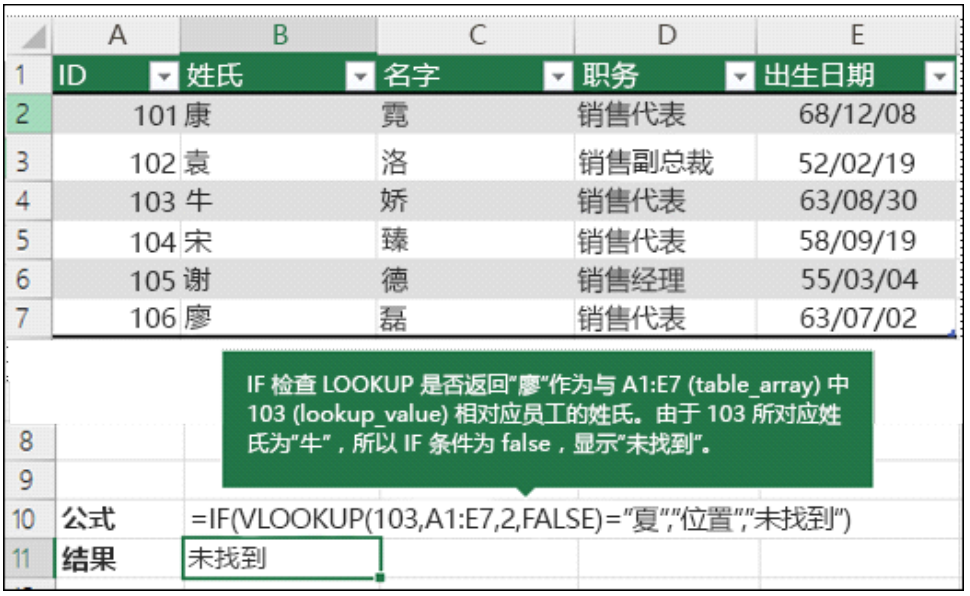
数据-数据工具-数据有效性验证

数据分列

选定-数据-分列-按字段长度/符号分列-拆分结果会自动覆盖后几列，所以要确保后面有空位

函数

=开始输入
1-based index
函数引用单元格时，默认使用的是相对引用，即扩展时会按相对位置返回计算结果，如果要使用绝对引用，需要输入符号$或者直接按F4
RANK(num, ref, [order])
num指要找位置的数字，ref是在哪里找，order=0/不写，则是降序，不为0就是升序。
如果有重复值，相同值会有相同的rank，但是会有几个rank被跳过（留空）
MID(str, start, len)
截取字符串的
字符串函数还有LEN, LEFT, RIGHT,VLOOKUP



VLOOKUP中FALSE要求精确匹配，TRUE是近似匹配

逻辑函数

IF, OR, AND

混合函数

Sumif Countif

A:A

选取整列

运算符顺序

: 空格，	引用运算符	=SUM(A1:B10) 区域运算符	1
		=SUM(A1:B5 A4:D9) 交叉运算符，运算的是overlap的区域	
		=RANK(A1, (A1:A10, C1:C10)) 联合运算符，指合在一起	
+-*/	算术运算符		2
&	文本运算符	="Excel"&"Home" 返回"ExcelHome"，连接作用	3
<>=<=	比较运算符	=(A1=A2)判断相等 =(B1<>"ABC")判断不相等	4

文本匹配中的通配符

*	任何字	?	任何单个字	~	解除字符的通配性，如~*就是查找带*这个字符本身的

错误的类型（常与IF嵌套使用）

ISERROR( )

括号中为：#N/A、#VALUE、#REF、#DIV/0、#NUM、#NAME？或#NULL时为TRUE

ISNA( )

括号中为：#N/A时为TRUE

IFERROR(value, value_if_error)

AVERAGEIF(range, criteria, average_range)

average_range是实际求值得区域，如果忽略会使用range。range是criteria作用的区域。criteria可以

使用通配符，也可以是字符串。如果两个范围大小和形状不同，则会使用 average_range 中左上方的单元格作为起始单元格，然后加入与 range 的大小和形状相对应的单元格确定。

	A	B
1	财产值	佣金
2	100,000	7,000
3	200,000	14,000
4	300,000	21,000
5	400,000	28,000
6	公式	说明（结果）
7	=AVERAGEIF(B2:B5,"<23000")	求所有佣金小于 23,000 的平均值 (14,000)
8	=AVERAGEIF(A2:A5,"<95000")	求所有财产值小于 95,000 的平均值 (#DIV/0!)
9	=AVERAGEIF(A2:A5,">250000",B2:B5)	求所有财产值大于 250,000 的佣金的平均值 (24,500)

若要在计算中包含引用中的逻辑值和代表数字的文本，请使用 AVERAGEA 函数。
若要只对符合某些条件的值计算平均值，请使用 AVERAGEIF 函数或 AVERAGEIFS 函数。

TRIMMEAN(array, percent)
从头尾一共去掉perc的数据，再计算平均值

LARGE(array, k)
返回第k个最大值，降序
SMALL(array, k)

FIND(文本, 范围,数值)
查找一个字符在另一个字符串中的位置
数值表示查找第几个.
=FIND（ “a”, “abcaef”,1 ）= 1
=FIND（ “a”, “abcaef”,2 ）= 4

SUBSTITUTE(text, old_text, new_text[, instance_num])
对将text中的old_text替换为new_text，如果指定了instance_num，则替换第instance_num个old_text，否则替换所有的。

ROUND(number, num_digits)
四舍五入，num_digits是位数
num_digits为0指四舍五入到最接近的整数；num_digits<0，则在小数点左侧四舍五入
ROUND(21.5, -1) 得20
Round(-1.475, 2)得-1.48
ROUNDUP始终 向上舍入，远离0
ROUNDDOWN 始终向下舍入，朝向0
MROUND(num, multiple)
=MROUND(1.3, 0.2)将 1.3 四舍五入到最接近 0.2 的倍数。1.4
=MROUND(5, -2)
返回错误值 #NUM!， 因为 -2 和 5 的符号不同。
向 Multiple 参数提供十进制值时，中点数字的舍入方向不明确。例如，MROUND (6.05,0.1) 返回

6.0，而 MROUND (7.05,0.1) 返回7.1。

COUNT
统计数字个数，不包含空值
COUNTA
统计非空单元格数
COUNTBLANK
统计空格数
COUNTIF

DATE
根据数值返回日期
DAY/MONTH/YEAR/WEEKDAY
返回日期是一个月中的第几天/月/年
NOW
返回当前日期和时间
TODAY
返回当前日期

公式求值可以debug公式是否有错误

颜色

开始-条件格式：为满足条件的单元格设置格式
可以为单元格设置颜色，然后利用颜色做筛选

------练习----------
=IF(COUNTIF(G:G,L3)>0,"有","") 判断L3是否在G行之中
=COUNTIF(M:M,"有")

判断两列是否相等（染色体区域是否是同一块）数据分列，然后
=IF(AND(A1=C1, B1=D1), "", "FAIL")
然后下拉至最后
=COUNTIF(E1:E7837, "FAIL")
注意： AND放前面，countif条件是字符的话直接写，如果是其他，如"=0"这样写

在一列中查找另一列的数据，并且标出来：
选中目标列，选择条件格式-新规则-使用公式确定要格式化的单元格-
=NOT(ISNA(VLOOKUP(A1,$B:$B,1,FALSE)))
A1是要突出显示的目标列的第一个单元格，B是筛选条件列

Head First Java

2018年10月3日 13:11

目录

基础

执行过程：源代码test.java--->编译器编译为test.class（字节码）--->JVM将字节码转换为机器码

每个应用程序只有一个main函数

```
public class MyFirstApp {

    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
} // 这是注释
```

int和boolean不相容，不能用int来作判断条件

instance variable/methods

测试一个类可以命名为类名TestDrive

main()的用途：测试真正的类/启动java应用程序
应该是对象与对象交互。
创建对象时，对象被存在称为堆的内存区域中。JVM清理回收。
Java中没有全局变量，但是任何变量只要加上public/static/final就可以变成常数。
jar文件中可以引入一个文字文件manifest，里面定义jar中哪一个文件带有main方法。
目录

变量

primitive主数据类型

类型	位数	值域
boolean与char		
boolean (Java虚拟机决定)		true或false
char	16 bits	0 ~ 65535
数值 (带正负号)		
integer		
byte	8 bits	-128~127
short	16 bits	-32768~32767
int	32 bits	-2147483648~2147483647
long	64 bits	-很大 ~ +很大
浮点数		
float	32 bits	范围规模可变
double	64 bits	范围规模可变

float f = 32.5f // 如果不加f，会被默认为double

变量命名：字母/_/$开头，不能用数字开头
没有对象变量。引用变量只是引用对象。（Dog myDog = new Dog()）
对于Primitive来说，变量值就是所代表的值；对于引用变量来说，变量值是取得特定对象的位表示法。
目录

数组

数组是对象，不是primitive。没有primitive的数组，只有装载primitive的数组。

String[] WordList = {"one", "two"};
int size = WordList.length;
Dog[] pets = new Dog[7];
现在pets是一个有7个Dog引用的数组，需要为引用们赋值（pets[0] = new Dog();）否则就还是null值。而失去引用的对象，会被回收。
目录

字符串

String s = WordList[0]
s = s + " lol"; // 字符串连接
目录

函数

Java在函数传参的时候是通过拷贝传递的
getter&setter(accessor&mutator)
封装：将实例变量标记为私有，并提供公有的getXXX和setXXX来控制存取动作。
== 比较两个primitive或判断两个引用是否引用自同一个对象
equals() 判断两个对象是否在意义上相等
目录

类

类定义对象所知及所为。对象所知者是实例变量，对象所为者是方法。
实例变量永远都会有默认值(但是局部变量没有默认值，未初始化就直接使用，编译器会报错)

integers/char	0
floating points	0.0
booleans	false
references/String/array(自然后俩都是引用)	null

目录

案例 SimpleDotCom

0. 确定实例变量和方法
SimpleDotCom
int [] locationCells
int numOfHits
String checkYourself(String guess)
void setLocationCells(int[] loc)

1. *编写伪代码：注重逻辑*

   说明实例变量的数据类型，作用；

   说明方法的输入/输出/作用，详细描述逻辑。

2. *编写测试代码*

   确定要测试的部分；

```
public class SimpleDotComTestDrive {
    public static void main(String[] args) {
        SimpleDotCom dot = new SimpleDotCom(); // test initialization
        int[] locations = {2,3,4};
        dot.setLocationCells(locations); // test setter
        String userGuess = "2";
        String result = dot.checkYourself(userGuess);
        String testResult = "failed"; // show result of check
        if (result.equals("hit")) {
            testResult = "passed";
        }
        System.out.println(testResult);
    }
}
```

3. *实现类*

   每个类写在单独的文件中，文件名与类名一致。

```
public class SimpleDotCom {
    private int[] locationCells;
    private int numOfHits = 0;
    public String checkYourself(String stringGuess) {
        int guess = Integer.parseInt(stringGuess);
        String result = "miss"; // initialize local variable
        // for(declaration : arrayName)
        // declaration: new local variable, like iterator
        for (int cell : locationCells) {
            if (guess == cell) {
                result = "hit";
                numOfHits++;
                break;
            }
        }
        if (numOfHits == locationCells.length) {
            result = "kill";
        }
        System.out.println(result);
```

```
            return result;
        }
        public void setLocationCells(int[] locs) {
            locationCells = locs; // array assignment
        }

}
import java.io.*;
public class GameHelper {
        public String getUserInput(String prompt) {
            String inputLine = null;
            System.out.print(prompt + " ");
            try {
                BufferedReader is = new BufferedReader(
                new InputStreamReader(System.in));
                inputLine = is.readLine();
                if (inputLine.length() == 0)
                    return null;
            } catch (IOException e) {
                System.out.println("IOException: " + e);
            }
            return inputLine;
        }
}

public class SimpleDotComGame {
        public static void main(String[] args) {
            int numOfGuesses = 0;
            GameHelper helper = new GameHelper();
            SimpleDotCom theDotCom = new SimpleDotCom();
        // Math.random() return double in 0-1. But <5 needed.
            int randomNum = (int)(Math.random() * 5);
            int[] locations = {randomNum, randomNum+1, randomNum+2};
            theDotCom.setLocationCells(locations);
            boolean isAlive = true;
            while(isAlive == true) {
                String guess = helper.getUserInput("enter a number");
                String result = theDotCom.checkYourself(guess);
                numOfGuesses++;
                if (result.equals("kill")) {
```

```
                    isAlive = false;
                    System.out.println("You took " + numOfGuesses + "
                    guesses.");
                }
            }

        }
    }
    public class MyFirstApp {
        public static void main(String[] args) {
            //SimpleDotComTestDrive test = new SimpleDotComTestDrive();
            //test.main(null);
            //SimpleDotComGame game = new SimpleDotComGame();
            //game.main(null);
        }
    }
```

4. 测试并改错

ArrayList

ArrayList test = new ArrayList<Integer>();
这里的Integer实际是对int的一个包装，是一个类，带有一个int的变量。ArrayList并没有add(int)的方法，但是java5后，编译器会自动包装autoboxing和解包装，但是在声明的时候仍然要用Boolean/Character/Byte/Short/Integer/Long/Float/Double。这种对应关系，在Interger对象是方法的参数/返回值/布尔表达式等等时仍然有效。
int num1 = new Integer(100);
test.add(num1);
test.contains(num1) // return true/false
可以动态地改变长度

这样包装的话，就可以带方法了。
String s = "2";
int x = Integer.parseInt(s);
double d = Double.parseDouble("420.24")
boolean b = new Boolean("true").booleanValue();
因此也可以将字符串转为数字：
String doubleString = Double.toString(d);
// 当然也可以String doubleString = "" + d;
// "+"是Java中唯一重载过的运算符。

目录

Math
random()

&&和||运算时会有shortcut，而&和|则不会，但它们通常用来做为位运算。

包

java.lang是一个预先被引用的包，所以可以不指定名称。import不会让程序变大或者变慢。

final

如果要防止特定的方法被覆盖，可以将方法加上final修饰符。如果将整个类标识为final则表示它是不能被继承的，则方法都不能被改变。

区分继承和重载

子类对父类方法的覆盖，要求参数一致，返回值兼容，存取权限也一致。这是多态的体现。
重载则是指方法名称相同，参数不同，这与多态无关。但是重载的话一定要求参数不同，如果参数相同只是返回值不同，则是不允许的。也可以更爱存取权限。
目录

接口

接口是无法初始化的抽象类。

```
abstract class Canine extends Animals {
        public void roam(){}
}
```

抽象类可以有static成员。
如有有抽象的方法，这个类就必须标记为抽象类。非抽象类中不能有抽象方法。
抽象类可以带有非抽象的方法，但是具体类必须实现所有的抽象方法。
所有类都是继承自Object这个类的。（是具体类，常用在进程同步化）

```
ArrayList<Obeject> myDogs = new ArrayList<Object>();
Dog aDog = new Dog();
//能把aDog放进去，因为aDog继承自Object
myDogs.add(aDog);
//这里会报错，因为get()方法返回的是Object类型，编译器无法确认它是dog。
Dog d = myDogs.get(0);
```

编译器通过引用类型判断有哪些方法可以使用。
instanceof运算符可以检查类型。

java不允许多重继承（有多个父类），但是可以通过接口解决。接口就是抽象类，一个类可以实现多个接口。
public interface Pet {……} //接口只写方法的声明，必须是abstract的。
public class Dog extends Canine implements Pet {……}

使用super.XXX可以调用父类的方法。
目录

构造器和垃圾收集器

栈上存放的是方法调用和局部变量。
堆上存放的是所有的对象。（每个对象都有实例变量）

调用的函数会放在栈顶，用完之后会被弹出。
对象引用变量与primitive

构造函数
没有返回类型，不能被继承。
如果有一个有参数的构造函数，则编译器不会再编写无参数的构造函数，如果需要，要手动写。

```
public Duck() {
    size = 34;
} //初始化对象的状态
```

抽象类也有构造函数，创建新对象时，所有继承下来的构造函数都会执行。
子类的构造函数可以通过super()或this()调用父类的构造函数。必须在第一行。

static方法：不依靠实例变量也就不需要对象的方法。
通过类名.调用，不能调用任何非静态变量，不能调用非静态方法。

静态变量在类被加载时初始化，所以会在任何对象创建前/任何静态方法执行前完成初始化。
如果没有赋值，会被设为默认值。

final变量表示一旦被初始化之后就不会被改动，所以static final在有初值之后就不会再变。如Math.PI。这样的常数变量要全大写。
static final初始化（如果不初始化会报错）
1.声明的时候赋值
2.在static{}里赋值

final的方法表示不能被覆盖，final的类表示不能被继承。

如果类已经是final了，那么不需要再标记为final方法

静态方法只能读取静态变量和存取静态方法，非静态方法可以读取静态变量，但不能调用静态方法。

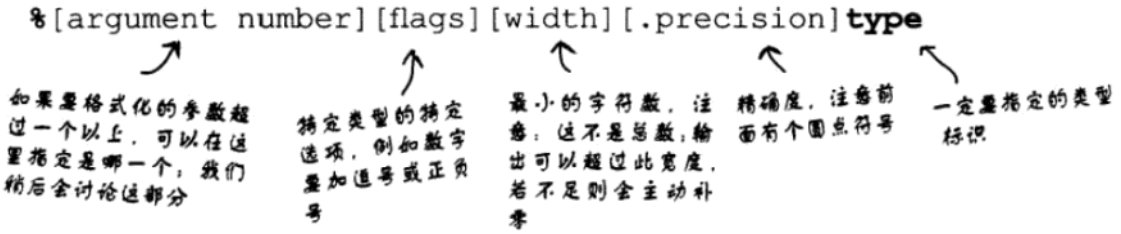
如果类只有静态方法，则可以将构造函数标记为private以避免初始化。

目录

数字格式化

可以使用String的format方法。

```
// 1,000,000,000 为数字加逗号分隔。类似c的printf，d指示数字类型
String s = String.format("%,d", 1000000000);
System.out.println(s);
// 加逗号，然后保留两位小数
format("I have %,.2f bugs to fix.", 476578.09876);
```



%x 十六进制

%c 字符

Date类型-栗子

```
import java.util.Date;

public class haha {
	public static void main(String[] args) {
		String s = String.format("%tc", new Date());
		// 报错：Type Date cannot be resolved to a type
		// 原因：java.uti.Date 和 java.sql.Date中都有Date。所以要么import一个，要么加上包名。
		System.out.println(s); // 完整日期　星期二 十月 30 10:22:41 CST 2018
		System.out.println(String.format("%tr", new Date())); // 只有时间　10:22:41 上午
		Date today = new Date();
		// 这里逗号直接输出了
		System.out.println(String.format("%tA, %tB %td",today,today,today)); // 周月日　星期二, 十月 30
		System.out.println(String.format("%tA, %<tB %<td", today)); // 周月日　星期二, 十月 30
	}
}
```

java.util.Date很多Date功能被弃用了。使用java.util.Calendar来操作日期。
Calendar是一个抽象类，所以new Calendar是无法通过编译的，如果一定要取实例，要用静态方法：
Calendar cal = Calendar.getInstance();
这里其实取的还是Calendar的子类的实例，可能是java.util.GregorianCalenda。(会根据locale特性返回合适的子类)
关键字段：

DATE/ DAY_OF_MONTH	HOUR/HOUR_OF_DAY	MILLISECOND
MINUTE	MONTH	YEAR
ZONE_OFFSET	etc.....	

重要方法：

add(int field, int amount)	get(int field)	getInstance()	getTimeInMills()
roll(int field, int amount)			
set(int field, int value)	set(year,month,day,hour,minute)		setTimeInMills()

```
import java.util.Calendar;

public class haha {
	public static void main(String[] args) {
		Calendar c = Calendar.getInstance();
		c.set(2019,0,11,13,59); // 年月日时分，月是0-based
		long day1 = c.getTimeInMillis(); // 微秒，从1970.1.1计数
		day1 += 1000 * 60 * 60; // 加上一个小时
		c.setTimeInMillis(day1);
		System.out.println("new hour " + c.get(c.HOUR_OF_DAY)); // new hour 14
		c.add(c.DATE, 35); // 加上35天
		// add 35 days Fri Feb 15 14:59:57 CST 2019
		System.out.println("add 35 days " + c.getTime()); // 返回日期
		// roll 35 days Fri Feb 22 14:59:57 CST 2019
		c.roll(c.DATE, 35); // 只有日期会动，月份不变
		System.out.println("roll 35 days " + c.getTime());
		c.set(c.DATE, 1); // 直接设置日期
		// set to 1 Fri Feb 01 14:59:57 CST 2019
		System.out.println("set to 1 " + c.getTime());
	}
```

```
}
```

static import的写法

用来import静态的类/变量或enum。如果只用一两次，不要用这个。

如果用很多次，且名称不会混淆，可以用static import。

```
import static java.lang.System.out;
import static java.lang.Math.*  // 可以用*
class WithStatic {
    public static void main(String [] args) {
        out.println("sqrt " + sqrt(2.0)); // 省略一些
    }
}
```

目录

异常处理

异常是一个Exception类型的对象

```
// 会抛出异常的方法必须提前声明
        public void takeRisk() throws BadException {
                if (abandonAllHope) {
                        throw new BadException();
                }
        }
        public void crossFingers() {
                try {
                        anObject.takeRisk(); // 这个函数会抛出异常，已声明
                } catch (BadException ex) {
                        System.out.println("Aaargh!");
                        // 如果无法从异常中恢复，要列出有用信息
                        ex.printStackTrace();
                }  finally { // finally块是无论是否有异常都要执行的程序
                       close();
                }
        }
```

编译器不会检查RuntimeException，同样，继承了它的类也不会被检查。

try/catch处理的是异常，而不是程序的逻辑错误，它的目标也是尽力恢复。

thows处可以写多个异常，逗号连接，这样就可以写多个catch语句来分别处理了。但是也不一定全部分别处理，此时要为几种异常搞一个父类，只抛出父类异常，只catch父类异常就好了。对于需要额外处理的子类，只catch子类的异常就好了。同时，特殊的子类处理要放在父类的上面，让它先被catch到。

当然也可以throws了，但是后面没有catch，这是一种duck策略，会一直向栈的下方传递，

如果一直到main，main还是throws掉，那么可以通过编译，但是没意义。
规则：
catch和finally里不能有try
catch必须紧跟try
try必须有catch或者finally，如果只有finally的话，这个外面的函数还是要throws异常。
目录

图形用户接口

```
import javax.swing.*;
// import ActionListener and ActionEvent
import java.awt.event.*;

// 事件只会通知有实现ActionListener的类
public class haha implements ActionListener {
	JButton button;

	public static void main(String[] args) {
		haha gui = new haha();
		gui.go();
	}
	public void go() {
		JFrame frame = new JFrame();
		button = new JButton("click me");
		// 向按钮注册
		button.addActionListener(this);
		frame.getContentPane().add(button);
		// exit when closing window
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.setSize(400, 200);
		// display, yeah
		frame.setVisible(true);
	}
	// 实现interface上的方法，即处理事件的方法
	public void actionPerformed(ActionEvent event) {
		button.setText("I've been clicked!");
	}
}
```

分析：Event对象携带信息，在实现接口的方法时作为参数被用到。事件源发出事件，接受注册，取得用户事件，并在用户采取操作时调用监听的事件的处理方法，这里的事件源就是按钮。

在GUI上加东西的三种方法：
1.在frame上放置widget
加上按钮/窗体/radio button等
frame.getContentPane().add(myButton);
2.在widget上绘制2D图形
graphics.fillOval(70,70,100,100);
3.在widget上绘制JPEG图
graphics.drawImage(myPic, 10, 10, this);

如果要在屏幕上放自己的图形，应该自己创建出有绘图功能的widget，然后像放普通widget一样放到frame上。此时应该创建JPanel的子类并覆盖paintComponent()方法。这个paintComponent不是自己调用的，要由系统调用。

```
import javax.swing.*;
import java.awt.*;

class MyDrawPanel extends JPanel {
        public void paintComponent(Graphics g) {
            g.setColor(Color.orange);
            // Fills the specified rectangle. The left and right edges of the
            rectangle are
            // at x and x + width - 1. The top and bottom edges are at y and y +
            height - 1
            g.fillRect(20, 50, 100, 100);
        }
    }
```

使用的时候在frame上用

```
JFrame frame = new JFrame();
frame.add(new MyDrawPanel());
// 注意不要被覆盖了=。=
```

paintComponent可以有多种应用，下面是栗子：

```
// 添加jpeg图像
public void paintComponent(Graphics g) {
            Image image = new ImageIcon("cat.jpg").getImage();
            //图像左上角的位置离panel的左边缘3个像素，离顶端向下4个像素
            g.drawImage(image, 3, 4, this);
        }
public void paintComponent(Graphics g) {
            // 前两个参数是起点坐标，后面是宽和高，现在取的是panel的，会填满。
            g.fillRect(0,0, this.getWidth(), this.getHeight());
```

```
            int red = (int) (Math.random() * 255);
            int green = (int) (Math.random() * 255);
            int blue = (int) (Math.random() * 255);
            // 但其实用random的话，当窗口大小改变时，颜色也会发生变化
            Color randomColor = new Color(red, green, blue);
            g.setColor(randomColor);
            // pos并不是圆心，而是其左上边界
            g.fillOval(0, 0, 50, 100);
        }
```

g所引用的对象其实是一个Graphics2D的实例。Graphics2D对象可以做的事比Graphics对象更多。但是要引用Graphics2D的方法时，还是要转化，不能直接用g参数。这就像Animal a = new Dog();虽然a是实际是一个Dog，但是它却是一个Animal的引用，要用Dog的方法，必须(Dog)a做强制类型转换。

可以对Graphics引用调用的方法如：drawImage() / drawLine() / drawPolygon() / drawRect() / drawOval() / fillRect() / fillRoundRect() / setColor()

类型转换：

```
Graphics2D g2d = (Graphics2D) g;
```

可以对Graphics2D引用调用的方法如：

fill3DRect() / draw3DRect() / rotate() / scale() / shear() / transform() / setRenderingHints()

其实frame.getContentPane().add(BorderLayout.CENTER, button)才是正确写法。不指定位置的时候，button被放在了默认的center区域，其实还有north/south/west/east。

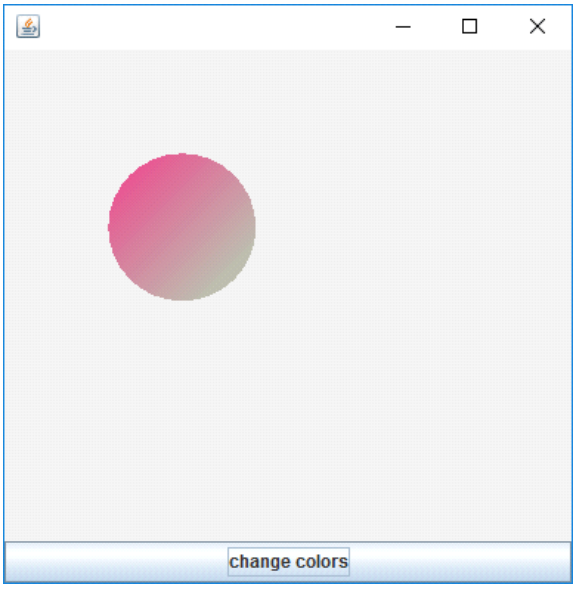
```
import javax.swing.*;
import java.awt.*;
// for ActionListener
import java.awt.event.*;
// set button and panel in different area
// and click button to change color
public class haha implements ActionListener{
    JButton button;
    JFrame frame;

    public static void main(String[] args) {
        haha gui = new haha();
        gui.go();
    }
    public void go() {
        frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // set button
        button = new JButton("change colors");
```

```
        button.addActionListener(this);
        // add componets
        frame.getContentPane().add(BorderLayout.CENTER, new MyDrawPanel());
        frame.getContentPane().add(BorderLayout.SOUTH, button);
        // MUST BE HERE. LAST STATEMENT.
        frame.setSize(400, 400);
        frame.setVisible(true);
    }
    public void actionPerformed(ActionEvent event) {
    // 按钮响应的了点击事件，重新绘制，其实调用的是paintComponent()
        frame.repaint();
    }

    class MyDrawPanel extends JPanel {
        public void paintComponent(Graphics g) {
            Graphics2D g2d = (Graphics2D) g;
            int i = 0;
            // 复习一下数组的写法：）
            int[] c = new int[6];
            for(;i <6; i++) {
                c[i] = (int)(Math.random()*255);
            }
            // 设置一个笔刷，起点，起始颜色，终点，终止颜色
            GradientPaint gradient = new GradientPaint(70,70, new
            Color(c[0],c[1],c[2]), 150,150, new Color(c[3],c[4],c[5]));
            g2d.setPaint(gradient);
            g2d.fillOval(70, 70, 100, 100);
        }
    }
}
```



内部类（就是完全包含在一个类之中的）可以使用外部所有的方法与变量，包括私有的。
这种存取，指的是内部类的实例可以存取，其所属的外部类实例的内容。这是因为我们是使用外部类的实例来创建内部类的实例的。

你也可以从外部类以外的程序代码来初始内部实例，但这要使用特殊的语法。通常不太会有机会要这么做，但还是先让你知道一下。

```
class Foo {
    public static void main (String[] args) {
        MyOuter outerObj = new MyOuter();
        MyOuter.MyInner innerObj = outerObj.new MyInner();
    }
}
```

```
import javax.swing.*;
import java.awt.*;
// for ActionListener
import java.awt.event.*;
// two buttons now. One for changing color,
// One for changing label
// let inner class implements ActionListener, not outer class
public class haha {
	JLabel label;
	JFrame frame;

	public static void main(String[] args) {
		haha gui = new haha();
		gui.go();
	}
	public void go() {
		frame = new JFrame();
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        // set label button
        JButton labelButton = new JButton("change label");
        // make LabelListener implements actionPerformed()
        labelButton.addActionListener(new LabelListener());
        // set color button
        JButton colorButton = new JButton("change color");
        colorButton.addActionListener(new ColorListener());
        // need to initialize :)
        label = new JLabel("Fool label");
        // add componets
        frame.getContentPane().add(BorderLayout.CENTER, new MyDrawPanel());
        frame.getContentPane().add(BorderLayout.SOUTH, colorButton);
        frame.getContentPane().add(BorderLayout.WEST, labelButton);
        frame.getContentPane().add(BorderLayout.EAST, label);
        // MUST BE HERE. LAST STATEMENT.
        frame.setSize(400, 400);
        frame.setVisible(true);
}
class LabelListener implements ActionListener {
        public void actionPerformed(ActionEvent event) {
                label.setText("aloha");
        }
}
class ColorListener implements ActionListener {
        public void actionPerformed(ActionEvent event) {
                frame.repaint();
                }
}
// 其实它不存取haha的变量和方法，完全可以写成独立类
class MyDrawPanel extends JPanel {
        public void paintComponent(Graphics g) {
                Graphics2D g2d = (Graphics2D) g;
                int i = 0;
                // 复习一下数组的写法： )
                int[] c = new int[6];
                for(;i <6; i++) {
                        c[i] = (int)(Math.random()*255);
                }
                // 设置一个笔刷，起点，起始颜色，终点，终止颜色
                GradientPaint gradient = new GradientPaint(70,70, new
```

```
                Color(c[0],c[1],c[2]), 150,150, new Color(c[3],c[4],c[5]));
                g2d.setPaint(gradient);
                g2d.fillOval(70, 70, 100, 100);
            }
        }
}
```

模拟小球的运动（假装是动画）

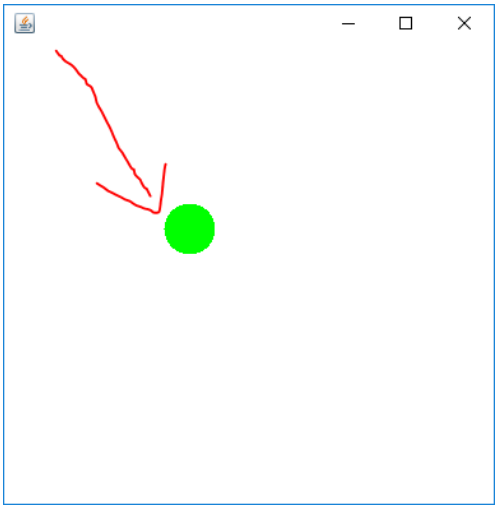
```
import javax.swing.*;
import java.awt.*;
//import java.awt.event.*;
// A ball run through the panel
public class haha {
    int x = 70;
    int y = 70;

    public static void main(String[] args) {
        haha gui = new haha();
        gui.go();
    }
    public void go() {
        JFrame frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        MyDrawPanel drawPanel = new MyDrawPanel();
        frame.getContentPane().add(drawPanel);

        frame.setSize(400, 400);
        frame.setVisible(true);
        // simulate movement
        for (int i = 0; i < 150; i++) {
            x++;
            y++;
            drawPanel.repaint();
            try {
                // slow down
                Thread.sleep(50);
            } catch(Exception ex) {} // empty
        }
    }
    // use x and y, then MyDrawPanel must be inner class
    class MyDrawPanel extends JPanel {
```

```
            public void paintComponent(Graphics g) {
                Graphics2D g2d = (Graphics2D) g;
                g.setColor(Color.white);
                // get() is from JPanel
                // erase last picture, or the trail will remian in panel
                g.fillRect(0, 0, this.getWidth(), this.getHeight());
                g2d.setColor(Color.green);
                g.fillOval(x, y, 40, 40);
            }
        }
}
```

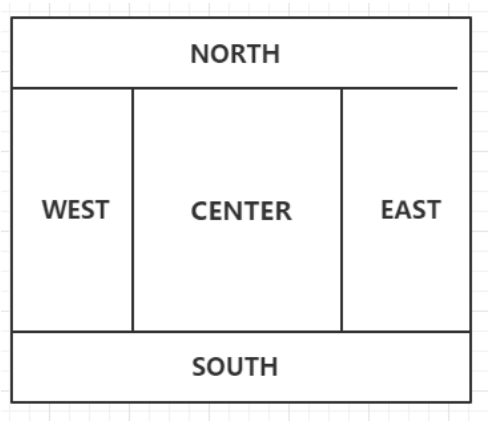


目录

Swing

布局管理器：

BorderLayout：把背景组件，如panel和frame分成五个区域，每个区域只能放一个组件，是frame默认的布局管理器。不会使用组件的大小，所以会出现按钮占据一整个区域的情况。放在东西的组件会取得组件的宽度，高度受组件管理器限制；放在南北的会取得组件的高度，宽度受组件管理器限制，中间的，只能取剩下的。



FlowLayout：每个组件会按照理想的大小呈现，从左到右按加入顺序，必要时会自动换行。是

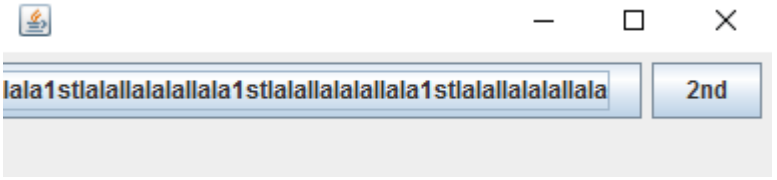
panel默认的。

BoxLayout：使用组件默认的大小，垂直方向，每行一个，不会自动换行。

举个例子

```
import javax.swing.*;
import java.awt.*;
//import java.awt.event.*;
// A ball run through the panel
public class haha {
	public static void main(String[] args) {
		haha gui = new haha();
		gui.go();
	}
	public void go() {
		JFrame frame = new JFrame();
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	   JPanel panel = new JPanel();
		// 这句表明使用BoxLayout
		panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
		JButton but1 = new JButton("1stlalallalalallala");
		JButton but2 = new JButton("2nd");
		panel.add(but1);
		panel.add(but2);
		// 现在会竖排。而不使用BoxLayout的话，不管but1多长，都会排一排
		frame.getContentPane().add(BorderLayout.EAST, panel);
		// 这个可以让窗口符合组件集合的大小，就是全部包好
		frame.pack();
		// 可以注释掉，不用设计窗口大小了
		frame.setSize(400, 400);
		frame.setVisible(true);
	}
}
```

下面展示了一种极端情况，就算长度已经超出了BorderLayout.EAST的宽度，仍然是排一排的。



目录

下载CDS数据

2018年10月18日 8:32

1. UCSC table browser
   https://genome.ucsc.edu/cgi-bin/hgTables

group	Genes and Gene Predictions
track	GENCODE v24/NCBI RefSeq
identifiers	贴ID list
	CDS的话只选CDS exons

   >danRer10_ncbiRefSeq_NM_131031.1 range=chr1:7965492-7967497 5'pad=0 3'pad=0 strand=- repeatMasking=none
2. batch entrez
   https://www.ncbi.nlm.nih.gov/sites/batchentrez
   贴上文件，retrieve，然后在结果页面发送到fasta即是全部结果。
   >lcl|NM_173247.1_cds_NP_775354.1_1 [gene=slc25a5] [db_xref=GeneID:192321,ZFIN:ZDB-GENE-020419-9] [protein=ADP/ATP translocase 2] [protein_id=NP_775354.1] [location=68..964] [gbkey=CDS]
3. eEnsembl biomart
   http://asia.ensembl.org/biomart/martview
   biomart也有R接口，再看
   >ENSDARG00000009212|ENSDART00000025620

Dataset	choose database: Ensembl Genes 94
Filters	Gene: Input external references ID list [Max 500 advised] transcript stable ID(s) ENSTxxxxxxxx
Attributes	Sequences coding sequence
	然后选count再查看results

获取各种ID？
uniprot retrieve/ID mapping
https://www.uniprot.org/uploadlists/

ucsc下载的hg38.fa数据是0-based
ncbi是0-based
bedtools getfasta -fo -fi -bed下载的是前闭后开的，所以ucsc截取后
>chr3:11102837-11102841
agat
对应ncbi的[11102838，111102841]

biopython

2018年10月18日 11:26

Seq对象

使用前 from Bio.Seq import Seq

字母表定义在Bio.Alphabet模块，可以查看

```
from Bio.Alphabet import IUPAC
help(IUPAC)
```

IUPAC是仅仅针对大写字母构成的序列的，所以将序列转为小写后，字母表会自动变化。

```
>>> dna_seq = Seq("ACGT", IUPAC.unambiguous_dna)
>>> dna_seq
Seq('ACGT', IUPACUnambiguousDNA())
>>> dna_seq.lower()
Seq('acgt', DNAAlphabet())
```

在创建序列对象时尽量指明其字母表，

```
>>> my_seq = Seq("AGTACACTGGT", IUPAC.ExtendedIUPACDNA)
>>> my_seq
Seq('AGTACACTGGT', <class 'Bio.Alphabet.IUPAC.ExtendedIUPACDNA'>)
>>> my_seq.alphabet
<class 'Bio.Alphabet.IUPAC.ExtendedIUPACDNA'>
```

对Seq对象切片与python里的字符串一样，[::-1]也是逆序，但会保留字母表。而且Seq不能修改，转字符串直接str()即可，打印直接自动转换。格式化输出时，Seq和%s一致。

```
fasta_format_string = ">Name\n%s\n" % my_seq
my_seq.lower() # 或者upper()也是很有用的
```

可变字符串

```
mutable_seq = my_seq.tomutable()
```

或者

```
from Bio.Seq import MutableSeq
mutable_seq = MutableSeq("GCCATTGTAATGGGCCGCTGAAAGGGTGCCCGA",
IUPAC.unambiguous_dna)
```

MutableSeq不可作为字典的key

变回Seq只要.toseq()即可

UnknownSeq保存时节省内存，只存长度和字母（核苷酸序列默认N，蛋白质默认X）

```
unk_dna = UnknownSeq(20, alphabet=IUPAC.ambiguous_dna)
```

Bio.Seq模块级别的函数（如reverse_complement，transcribe，back_transcribe，translate等）可以直接接受string

获取互补序列：

```
my_seq.complement()
```

反向互补序列：
my_seq.reverse_complement()

from Bio import SeqUtils
计算GC
SeqUtils.GC()
计算分子量

```
>>> print("%0.2f" % molecular_weight("AGC", "DNA"))
    949.61
>>> print("%0.2f" % molecular_weight("AGC", "RNA"))
    997.61
>>> print("%0.2f" % molecular_weight("AGC", "protein"))
    249.29
```

不指明默认DNA

```
nt_search(seq, subseq)
    Search for a DNA subseq in sequence, return list of [subseq, positions].
    Use ambiguous values (like N = A or T or C or G, R = A or G etc.),
    searches only on forward strand.
seq1(seq, custom_map=None, undef_code='X')
    Convert protein sequence from three-letter to one-letter code.
seq3(seq, custom_map=None, undef_code='Xaa')
    Convert protein sequence from one-letter to three-letter code
six_frame_translations(seq, genetic_code=1)
    Return pretty string showing the 6 frame translations and GC content.
```

python3的

```
xGC_skew(seq, window=1000, zoom=100, r=300, px=100, py=100)
    Calculate and plot normal and accumulated GC skew (GRAPHICS !!!).
```

读取压缩文件

```
import gzip
handle = gzip.open("ls_orchid.gbk.gz", "r")

import bz2
handle = bz2.BZ2File("ls_orchid.gbk.bz2", "r")

from Bio import SeqIO
records = (rec for rec in SeqIO.parse("test.fasta", "fasta") if len(rec) > 100)
SeqIO.write(records, "ls_orchid.tab", "tab")


from Bio import Entrez, SeqIO

def get_seq(inFile, outFile):
   ids = []
   raw = inFile.readlines()
   for item in raw:
      ids.append(item.split("\n")[0])

   Entrez.email = "pxy7896@163.com"

   for seq_id in ids:
      handle = Entrez.efetch(db="nucleotide", id=str(seq_id), rettype="fasta", retmode="text")
      record = handle.read()
      outFile.write(record.rstrip('\n'))
```

```
    handle.close()
```

```
from Bio.Blast import NCBIXML
result_handle = open("my_blast.xml")
# 输入只有一条结果
blast_record = NCBIXML.read(result_handle)
# 输入有多条结果
blast_records = NCBIXML.parse(result_handle)
# 可以用next读下一条
blast_record = next(blast_records)
....
blast_record = next(blast_records)
# 或者直接用循环
for blast_record in blast_records:
...

# 可以在biopython里调用本地blast
from Bio.Blast import NCBIStandalone
result_handle, error_handle = NCBIStandalone.blastall(my_blast_exe, "blastn",
my_blast_db, my_blast_file)
# 或者在线blast
from Bio.Blast import NCBIWWW result_handle = NCBIWWW.qblast("blastn", "nr",
"8332116")
# 然后把blast结果写入文件
blast_results = result_handle.read()
with open( "my_blast.xml", "w" ) as save_file:
    save_file.write( blast_results )
```

SLC2A2 codes reading-python

2018年10月18日 13:38

ExtractExonWithinCDS_1.py

joinSet 对于每行记录，每个exon只提取位于CDS区域之间的范围。

但是输出的output.txt是乱序的。

```
from collections import defaultdict
exon = defaultdict(set)
# 以类型为参数初始化，key不存在时返回set([])，不会报错。
print exon['sca']
```

```
>>>a = [1,2,3]
>>> b = [4,5,6]
>>> c = [4,5,6,7,8]
>>> zipped = zip(a,b)
 # 打包为元组的列表 [(1, 4), (2, 5), (3, 6)]
>>> zip(a,c)
 # 元素个数与最短的列表一致 [(1, 4), (2, 5), (3, 6)]
>>> zip(*zipped)
 # 与 zip 相反，*zipped 可理解为解压，返回二维矩阵式 [(1, 2, 3), (4, 5, 6)]
```

```
import pandas as pd

data = pd.read_csv # return data.frame
for index, row in data.iterrows():
# 按行遍历data.frame
```

所以现在排序

-k是指指定第几列作为排序的key。1-based。这里首先按染色体编号排序，然后按第二列exon start的数值大小排序（n指按数字排序，从大到小。r则是逆序）

```
sort -k 1,1 -k 2,2n output.txt > output1.sorted.bed
```

现在合并

-s 只合并同一条链上的。-nms报告被merge的features（现已取消，可看-c -o distinct）

-d 指间距多少的features会被合并

```
bedtools merge -i output1.sorted.bed -s -d 120 -c 4 -o distinct > output.sorted.merged.bed
```

这样输出的时候，strand会在第四列，报告在第五列

```
bedtools merge -i SLC2A2_raw_sorted_name.bed -s -d 120 -c 4 -o distinct > SLC2A2
_raw_sorted_merged.bed
```

bug:

如果从windows拷贝到linux下，可能会让文件格式发生变化。用cat -A XX可以查看

```
pangxiaoyi@pangxiaoyi-VirtualBox:~/Desktop/SLC2A2$ cat -A output.txt
chr3^I171026655^I171026670^ISLC2A2_exon_1^I16^I-^M$
chr3^I171018530^I171018623^ISLC2A2_exon_2^I94^I-^M$
chr3^I170999064^I170999166^ISLC2A2_exon_3^I103^I-^M$
chr3^I171005415^I171005415^ISLC2A2_exon_4^I1^I-^M$
```

即末尾会多一个^M，所以在末尾添加字符时，实际会添加到行首，覆盖一个字符。

所以merge有时行首乱码或者使用-s参数时根本不输出。

因此，对于从win拷贝来的文件，最好执行一个dos2unix，改格式。

```
pangxiaoyi@pangxiaoyi-VirtualBox:~/Desktop/SLC2A2$ dos2unix output.txt
dos2unix: 正在转换文件 output.txt 为Unix格式...
pangxiaoyi@pangxiaoyi-VirtualBox:~/Desktop/SLC2A2$ cat -A output.txt
chr3^I171026655^I171026670^ISLC2A2_exon_1^I16^I-$
chr3^I171018530^I171018623^ISLC2A2_exon_2^I94^I-$
chr3^I170999064^I170999166^ISLC2A2_exon_3^I103^I-$
```

dos2unix可以使用sudo apt-get install dos2unix安装

另外，在linux下用cp/cat并不能改变文件，手动选中，copy内容还能去掉行尾的^M。

^(*￣(oo)￣)^

^I是tab，空格就是空的==

ExtractExonWithinCDS_2.py

改范围，两端发生了变化，output.sorted.merged.bed->target.bed

改格式，target.bed->target.final.bed

ExtractExonWithinCDS_3.py

向外扩一点，切成长度为120的片段

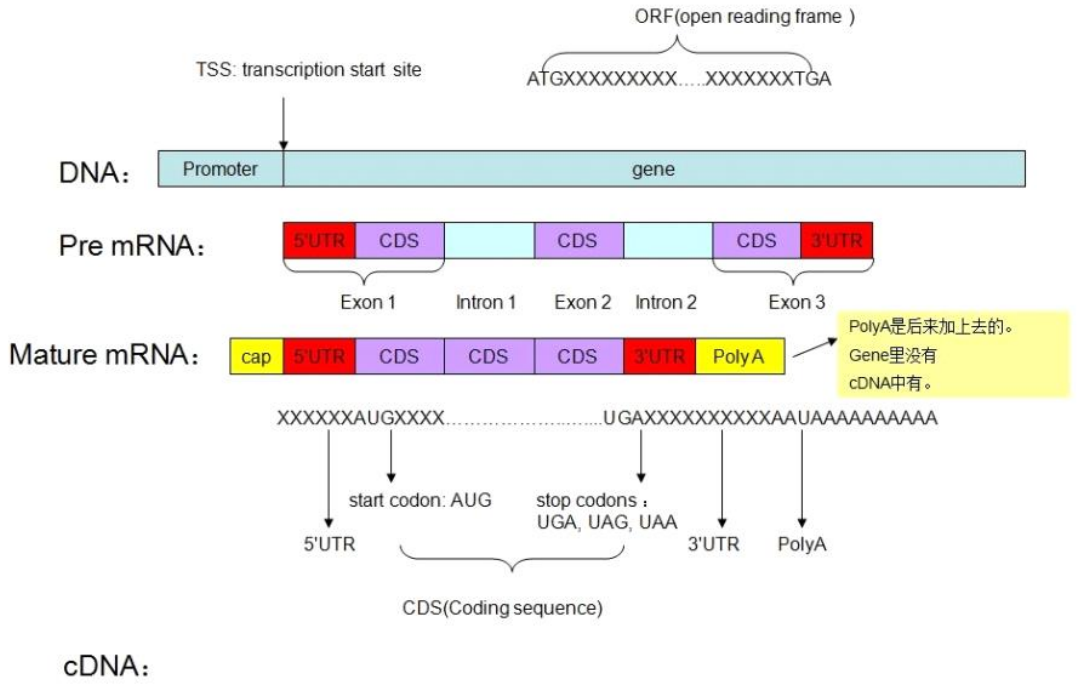
target.final.bed->target.final.expanded.bed

ExtractExonWithinCDS_4.py

重命名，end+1

uniq命令： 删除重复的行 -c 指在每行左边输出次数

awk '{print $1}' 输出文本的第一行



Soft masking indicates masked regions by using lower-case letters. Hard masking (-hardmask option) overwrites masked regions with a wildcard letter, N for nucleotides or X for proteins.

web基础梳理

2018年10月19日 13:18

运行一个简单的本地HTTP服务器：
python -m SimpleHTTPServer
现在可以通过转到localhost:8000查看目录的内容
如果8000被占用，则上面的命令应该再加一个端口号，访问的时候也加。

可以在goole fonts中选择字体，在CSS中使用。

应该在文件名中使用小写和连字符（google会把连字符当作词的分隔符，但不会这样处理下划线）。浏览器、web服务器和编程语言不能一致地处理空格。一些服务器会将文件名里的空格替换为%20。

网站结构：
index.html主页
images 图像
styles 样式表
scripts JS代码

html里路径仍然是/

html的debug
https://validator.w3.org
墙内太慢了，不如自己debug

http是文本化的无状态协议
动态网站是指，一些响应内容只有在被需要的时候才会生发的网站。在一个动态网站上，页面通常是通过将数据库的数据植入到HTML模板中的占位符中而产生的。

django

安装
sudo pip install Django
新建项目
django-admin startproject XXX

HelloWorld: 文件夹，项目的容器。
manage.py: 一个实用的命令行工具，可让你以各种方式与该 Django 项目进行交互。
HelloWorld/__init__.py: 一个空文件，告诉 Python 该目录是一个 Python 包。
HelloWorld/settings.py: 该 Django 项目的设置/配置。
HelloWorld/urls.py: 该 Django 项目的 URL 声明; 一份由 Django 驱动的网站"目录"。

HelloWorld/wsgi.py: 一个 WSGI 兼容的 Web 服务器的入口，以便运行你的项目。

进入主目录下启动服务

python manage.py runserver 0.0.0.0:8000

报错：

解决：

寻找占用端口的进程

ps -aux | grep -i "manage.py"

```
pangxiaoyi@pangxiaoyi-VirtualBox:~/Desktop/probe$ ps -aux | grep -i "manage.py"
pangxia+  3099  0.0  0.7 124124 28908 pts/0    T    09:16   0:00 python manage.p
y runserver 0.0.0.0:8000
pangxia+  3101  0.3  0.8 276036 32720 pts/0    Tl   09:16   0:00 /usr/local/bin/
python manage.py runserver 0.0.0.0:8000
pangxia+  3146  0.0  0.0  21536  1064 pts/0    S+   09:19   0:00 grep --color=au
to -i manage.py
```

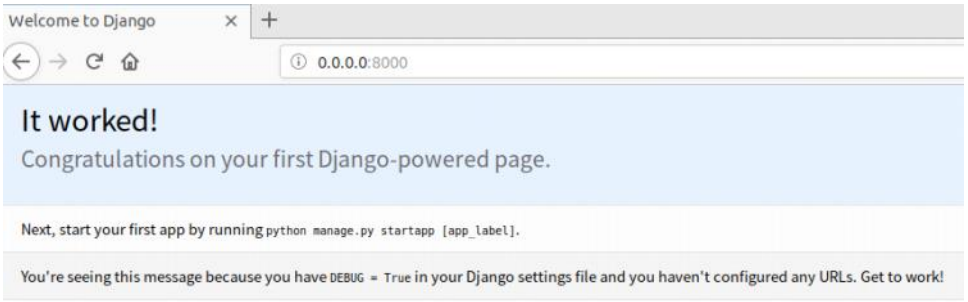
杀死

kill -9 3101

修改settings.py

ALLOWED_HOSTS = ['*']

然后再runserver就可以在0.0.0.0:8000看到



第一个helloworld程序：

新建一个view.py

```
from django.http import HttpResponse
def hello(request):
    return HttpResponse("Hello world ! ")
```

然后修改urls.py，绑定view里的hello函数与url

```
from django.conf.urls import url
from . import view
urlpatterns = [
    url(r'^$', view.hello),
```

]
直接刷新0.0.0.0.8000即可看到结果

url函数
url(regex, view[, kwargs, name])
regex是正则表达式，与之匹配会执行view，kwargs是视图使用的字典类型的参数，name用来反向获取url

使用模板
在settings.py里修改模板的路径
TEMPLATES里'DIRS'改为[BASE_DIR+"/templates",],
并在主目录下建立文件夹templates，在里面放上hello.html，其内容是
<h1>{{ hello }}</h1>
此处变量用双大括号标注。
修改view.py为

```
from django.shortcuts import render
def hello(request):
    context          = {}
    context['hello'] = 'Hello World!'
    return render(request, 'hello.html', context)
```

这样，templates放的是网页模板，里面用到的是view.py里定义的变量，因此实现了模板和数据的分离。
模板标签
if支持嵌套，condition也接受and/or/not

```
{% if condition1 %}
   ... display 1
{% elif condition2 %}
   ... display 2
{% else %}
   ... display 3
{% endif %}
```

for支持嵌套

```
{% for athlete in athlete_list %}
    <h1>{{ athlete.name }}</h1>
    <ul>
    {% for sport in athlete.sports_played %}
        <li>{{ sport }}</li>
    {% endfor %}
    </ul>
{% endfor %}
```

反向迭代

```
{% for athlete in athlete_list reversed %}
```

还有ifnotequal

```
{% ifequal section 'sitenews' %}
```

```
    <h1>Site News</h1>
{% else %}
    <h1>No News Here</h1>
{% endifequal %}

{# 这是一个注释 #}
```

{% include %} 标签允许在模板中包含其它的模板的内容。

过滤器可以在变量被显示前修改它，可以使用管道字符|套接

```
{{ name|lower }} {#将name变量转换为小写#}
{{ my_list|first|upper }} {#将第一个元素转换为大写#}
{{ bio|truncatewords:"30" }} {#显示bio变量的前30个字符#}
length {#返回变量长度#}
addslashes {#添加反斜杠到任何反斜杠、单引号或者双引号前面。#}
{{ pub_date|date:"F j, Y" }} {#按指定的格式字符串参数格式化date或datetime对象#}
```

模板可以继承。新建base.html，block部分是可以被继承和替换的。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>ProbeDesignTest</title>
</head>
<body>
    <h1>Hello World!</h1>
    <p>testing……</p>
    {% block mainbody %}
      <p>original</p>
    {% endblock %}
</body>
</html>
```

修改hello.html，首先声明父模板，然后在与base.html同名的block中修改内容。

```
{%extends "base.html" %}

{% block mainbody %}
<p>继承了 base.html 文件</p>
{% endblock %}
```

模型

在根目录下创建app

django-admin startapp app

然后注册这个应用，在INSTALLED_APPS中添加一行

'app.apps.AppConfig',

查看urls.py

url映射通过urlpatterns管理，url()函数要么将URL式样(URL pattern)关联到特定视图(specific view)，将在模式匹配时显示；要么关联到某个URL式样列表的测试代码。 (第二种情况下，URL式样是目标模型里的“base URL”)

更改为

```
from django.conf.urls import url, include
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
]
urlpatterns += [
    url('', include('app.urls')),
]
# path函数已经不能用了，注意。
```

在probeD文件夹下新建urls.py

```
from django.conf.urls import url
from . import views

urlpatterns = [
]
```

中文注释的话，开头要加

```
# -*- coding: UTF-8 -*-
```

celery 异步处理

优势：主进程退出之后，任务可以在后端的队列中继续进行。

应用场景如用户注册邮箱

pip install celery

```
Successfully installed amqp-2.3.2 billiard-3.5.0.4 celery-4.2.1 kombu-4.2.1 vine-1.1.4
```

```
def post(request):
    result = add.delay(2, 3)
```

- 使用函数名.delay()即可使函数异步执行
- 可以通过result.ready()来判断任务是否完成处理
- 如果任务抛出一个异常，使用result.get(timeout=1)可以重新抛出异常
- 如果任务抛出一个异常，使用result.traceback可以获取原始的回溯信息

来自 <https://my.oschina.net/37Y37/blog/1920149>

任务调用方法：

delay和apply_async

delay比较简单，apply_async可以带很多参数，如queue可以指定队列，因此可以将不同的任务分配到不同的队列

每个task有三种状态：PENDING STARTED SUCCESS

任务状态查询：

result.state # 返回字符串

result.id是id

这里再做一次修改

CELERY_BROKER_URL = 'redis://localhost:6379/0'
CELERY_RESULT_BACKEND = 'redis://localhost:6379/0'

这里的0表示使用的是redis的0号队列。这样可以让同一个任务的实例单独使用同一个队列，这在有多个celery任务时比较有用。

要开这个啊！！！！！

```
celery worker -A probe -l info
```

来自 <https://www.cnblogs.com/pxy7896/p/9930377.html>

python manage.py migrate

JSON

2019年1月30日 17:22

JSON data types:

Sr.No.	Type & Description
1	Number double- precision floating-point format in JavaScript
2	String double-quoted Unicode with backslash escaping
3	Boolean true or false
4	Array an ordered sequence of values
5	Value it can be a string, a number, true or false, null etc
6	Object an unordered collection of key:value pairs
7	Whitespace can be used between any pair of tokens
8	null empty

Number

- It is a double precision floating-point format in JavaScript and it depends on implementation.
- Octal and hexadecimal formats are not used.
- No NaN or Infinity is used in Number.

```
var json-object-name = { string : number_value, .......}
```

String

```
var json-object-name = { string : "string value", .......}
```

```
u four hexadecimal digits
```

Boolean

```
var json-object-name = { string : true/false, .......}
```

Array

- It is an ordered collection of values.
- These are enclosed in square brackets which means that array begins with .[. and ends with .]..
- The values are separated by , (comma).
- Array indexing can be started at 0 or 1.
- Arrays should be used when the key names are sequential integers.

Syntax

```
[ value, .......]
```

Example

Example showing array containing multiple objects –

```
{
```

```
  "books": [
    { "language":"Java" , "edition":"second" },
    { "language":"C++" , "lastName":"fifth" },
    { "language":"C" , "lastName":"third" }
  ]
}
```

来自 <https://www.tutorialspoint.com/json/json_data_types.htm>

notepad++操作指南

2018年10月26日 16:48

去除行尾空格和空白行：按CTRL+F 选择正则表达式 -- 查找目标：\s+$ 替换为空。\s的意思是匹配任何空白字符，包括制表符、空格、换页符；等价于[\f\n\r\t\v]。

去除行首空格：按CTRL+H 选择正则表达式-- 查找目标：^\s+ 替换为空。

去除空白行：

Edit -> line operation -> remove blank line

RepeatMasker

2018年10月30日 11:08

RM是library-based，通过相似性比对来识别重复序列，可以屏蔽序列中转座子重复序列和低复杂度序列（默认将其替换成N）。使用数据库Dfam和Repbase。

The Dfam database is a collection of **Repetitive DNA element** sequence alignments, **hidden Markov models (HMMs)** and **matches lists** for complete Eukaryote genomes.

Repbase是由美国遗传信息研究所（GIRI）创建并维护，收录了转座子和其他重复序列及其注释信息。

本地安装RepeatMasker，除了需要RepeatMasker主程序外，还需要TRF（Tandem Repeats Finder）、序列搜索引擎（以RMBlast为例）以及Repbase数据库。

搜索引擎可以安装多个，但是每次只能用一个。

Using RepeatMasker to Identify Repetitive Elements in Genomic Sequences

要屏蔽的区域：low-complexity DNA sequences and interspersed repeats

比对引擎：cross_match WU-BLAST(更快)

如果DNA source没有参考基因组，那么需要用RECON或者RepeatScout建立一个Repbase类型的文件

安装：

http://www.repeatmasker.org/RMDownload.html

sequence search engine

cross_match 要注册啥的，没搞

RMBlast blast的修改版本，此处用了2.2.28版本，需要下载

http://www.repeatmasker.org/RMBlast.html

这里的两个binary，然后解压就可以了

HMMER 下载v3.1b2版本

ABBlast/WUBlast 也要注册啥的，没弄

TRF

下载TRF v4.0.4

Repeat database

下载Dfam和RepBase(要注册下载)

装完之后用./configure配置，修改好path就可以了。

暂时设置RMBlast为default。

最简单的命令

RepeatMasker/RepeatMasker -species human sequence.fasta

最常用：

./RepeatMasker -species human -engine hmmer

除了控制台输出外，还会在同目录下产生几个文件：

输入文件名.cat //不懂

输入文件名.masked // 已屏蔽完的fasta序列

输入文件名.out // 重复区域的统计信息，如类型，位置等

bit	perc	perc	perc	query	position in query		matching	repeat	position in repeat			
score	div.	del.	ins.	sequence	begin	end	(left)	repeat	class/family	begin	end	(left)	ID
216	4.4	6.4	0.0	NC_000017.11:c43125483-43044295	504	706	(80483)	+ AluSx	SINE/Alu	66	281	(31)	1
32	0.0	0.0	0.0	NC_000017.11:c43125483-43044295	707	736	(80453)	+ (CAAAA)n	Simple_repeat	1	30	(0)	2
305	10.2	2.0	0.0	NC_000017.11:c43125483-43044295	1837	2131	(79058)	+ AluSc	SINE/Alu	1	301	(8)	3
230	16.7	1.7	5.5	NC_000017.11:c43125483-43044295	2250	2409	(78780)	C AluSx3	SINE/Alu	(22)	289	128	4

输入文件名.tbl

各种统计信息

阈值设定：

-lib 指定数据库，default是灵长类的

-cutoff 使用-lib时设置阈值，默认225。cutoff 值低的会有错配。

-nolow 不去mask low-complexity DNA or simple repeats

-div sets the divergence level to limit the masking and annotation to a subset of less diverged (younger) repeats.

速度设定：

-q 快

-qq 更快

-s 慢就更灵敏

-pa 如果有多个输入或者输入很大，可以考虑多处理器加速

-w WU-BLAST比cross_match快，但是后者更准确

如果长序列效果不好，可以

修改RepeatMasker中的$maxsize，改大，但是内存需求也会变大

或者切断

如果空间不足，RM不会报错，可能会有貌似正确的结果

如果用了WU-BLAST，最好用-s

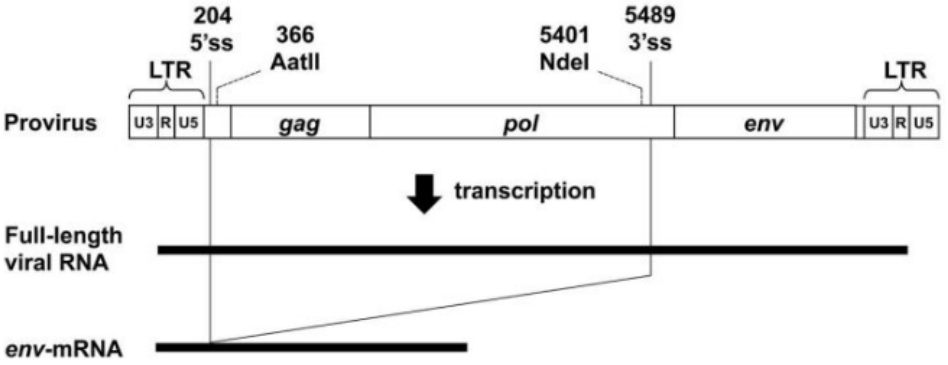
短序列（<2kb）的可能精确度差一点

转座子transposon

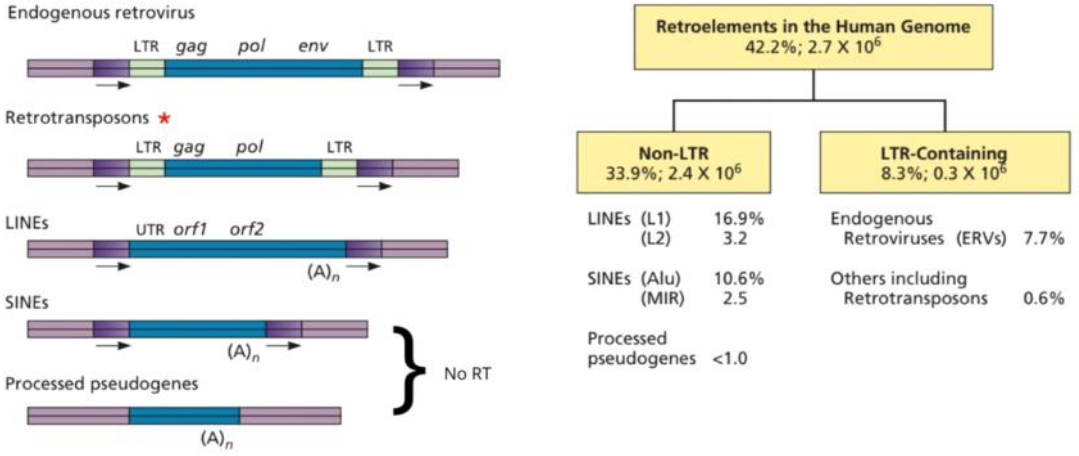
一类DNA序列，它们能够在基因组中转录或逆转录，在内切酶的作用下，在其他基因座上出现。I型转座子即反转录转座子，该型转座子会先被转录为RNA，然后利用逆转录酶将该RNA逆转录为cDNA，然后才被插入到目标位点中。“复制-粘贴”。II型转座子也称不复制转座子，其序列两端是两段直接重复序列（direct repeat, dR），与它们接壤的是反向重复序列（invert repeat, iR），中间是插入序列（insert sequence, IS）。所以II型的中间体就是其本身，“剪切-粘贴”。

假基因是一类本来正常，然后因为突变或转座而*可能*失去原来功能的基因。在环境压力下，某些假基因可以重新被激活，而某些假基因则有着调控基因表达的作用。可总结为“假作真时真亦假”。它们与原来的基因可能很相似，但又可以有很大差异。

人体约有40%的DNA与逆转录病毒有关，其中7.7%的DNA与逆转录病毒非常相似，称之为内源逆转录病毒（endogenous retrovirus, ERV）。

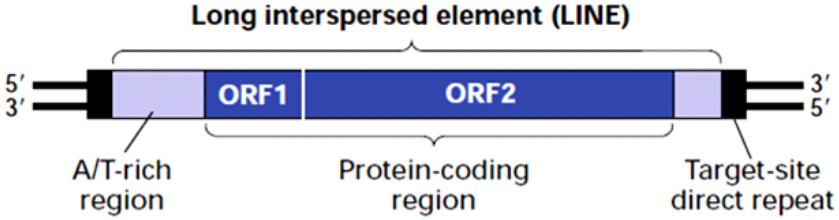


病毒两端有两条相同的序列，LTR(long terminal repear)，LTR不编码蛋白，主要起调控作用。中间三段基因，gag编码了衣壳蛋白等结构蛋白，pol编码了逆转录酶、整合酶、蛋白酶这些病毒复制需要的酶，env编码了病毒包膜的糖蛋白。所有的逆转录病毒都有这三个基因。人类的内源逆转录病毒HERV也有这三段基因和两个LTR，也可以像逆转录病毒一样，逆转录到别处。HERV可能是很久之前感染过人体胚胎，然后逐渐扩增到7.7%的规模，但是已经变异失去了制造病毒颗粒的能力。



逆转录转座子retrotransposon不包含env，可能是逆转录病毒的来源。所有反转录转座子都有一个共同特点，就是在其插入位点上产生短的正向重复序列。它是许多真核生物中数量最大的一类可活动遗传成分。在植物中特别丰富，它们是核DNA的一个主要组成部分。哺乳动物中，几乎有一半的基因组包含转座子或残余转座子。

LINE中有编码与逆转录酶/整合酶相似活性的酶，所以可能也能逆转录；长度6K



SINE中则没有编码逆转录酶，（需要在细胞内已有的酶系统的作用下进行转座）可能是在LINE辅助下进行逆转录和整合的。Alu是属于SINE的。长度约300bp

近年的研究显示，灵长目LTR逆转座子已固定在基因组中，已无转座活性(Lander et al.，2001);灵长目动物基因组中仍有转座活性的元件是non-LTR逆转座子，主要包括长散在重复元件LINE1(long interspersed element 1，L1)、Alu元件、SVA元件等

L1是人类基因组中唯一的自主性逆转座子，其拷贝占17%，但只有极少数有转座活性，其中6

个活性最高的L1拷贝介导了大部分L1转座活动。
Alu元件不能编码逆转录酶，属于非自主转座子，它们利用L1编码ORF2的逆转录酶进行逆转座活动。属于SINE。是灵长类动物基因组中数量最丰富的逆转座子。
典型的SVA元件长约2 kb。SVA逆转座子起源最晚，是人科动物中特有的逆转座子，属于SINE家族中的一员。

逆转座子对基因组结构的影响来源有两种，一是逆转座过程本身，一是其产生的同源序列：

a. 逆转座过程对基因组结构的影响：

1.插入突变
逆转座子对插入位点有选择性
2.侧翼序列转导
转座时，除了对自身进行转录，有时也会将上下游的侧翼序列进行转录。侧翼序列转导可将本来不连锁的基因连接起来，对新基因的形成和基因组的进化都有着重要作用。
3.基因逆转座
基因逆转座(gene retrotranspositon)是指只有基因序列发生逆转座，而不伴随逆转座子的转座过程。有时候，一些mRNA可以采取和Alu、SVA相同的策略，捕获L1的逆转录元件从而逆转录插入到基因组中。复制到新位点的基因来源于mRNA的逆转录，因此并不含有上游调控区域，除非获得新的调控区域，这些基因即成为逆转座的假基因(retropseudogene)
4.DNA双链断裂
5.侧翼序列切除
当L1和Alu插入基因组新位点时，可能会引起邻近基因组序列的缺失。

b. 逆转座子同源序列对基因组结构的影响：

1.DNA双链断裂的修复
2.异常重组

3.微卫星的形成
微卫星(microsatellite)也叫短串联重复序列(short t and em repeat，STR)或简单重复序列，是由几个(多为2~4个)碱基对作为核心单位，串联重复形成的一类DNA序列。

ucsc的repeat数据，其分类如下面链接所示
https://blog.csdn.net/tanzuozhev/article/details/80958785

biomaRt

2018年10月31日 13:57

一般教程会说用bioconductor安装，不过我这边总是提示没有适合我的安装包，所以使用压缩文件手动安装。有两个必须的依赖包要安装，直接在R里装就可以。

sessionInfo()

```
R version 3.4.4 (2018-03-15)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 18.04.1 LTS

Matrix products: default
BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1

locale:
 [1] LC_CTYPE=zh_CN.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=zh_CN.UTF-8        LC_COLLATE=zh_CN.UTF-8
 [5] LC_MONETARY=zh_CN.UTF-8    LC_MESSAGES=zh_CN.UTF-8
 [7] LC_PAPER=zh_CN.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=zh_CN.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] biomaRt_2.37.9

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.18        AnnotationDbi_1.40.0 magrittr_1.5
 [4] BiocGenerics_0.24.0  hms_0.4.2           progress_1.2.0
 [7] IRanges_2.12.0       bit_1.1-14          R6_2.2.2
[10] rlang_0.2.1          httr_1.3.1          stringr_1.3.1
[13] blob_1.1.1           tools_3.4.4         parallel_3.4.4
[16] Biobase_2.38.0       DBI_1.0.0           bit64_0.9-7
[19] digest_0.6.15        assertthat_0.2.0    crayon_1.3.4
[22] S4Vectors_0.16.0     bitops_1.0-6        curl_3.2
[25] RCurl_1.95-4.11      memoise_1.1.0       RSQLite_2.1.1
[28] stringi_1.2.4        compiler_3.4.4      prettyunits_1.0.2
[31] stats4_3.4.4         XML_3.98-1.12       pkgconfig_2.0.1
```

```
d <- useEnsembl(biomart="ensembl")
dat <- useDataset("hsapiens_gene_ensembl",d)
attributes <- listAttributes(dat) // 这里是一个2579*3的data.frame，包括名称，描述和page
filters <- listFilters(dat) // 这里是一个373*2的data.frame，包括名称和描述
attributePages(dat) // 显示page种类
[1] "feature_page" "structure"    "homologs"     "snp"          "snp_somatic"
[6] "sequences"


listAttributes(mart, page,what = c("name","description","page"))
```

searchAttributes(mart, pattern)
// pattern是正则表达式。.*匹配所有。

biomaRt的数据组织：
mart下面是database，database的数据被组织成页面。 每个页面都包含一个属性子集。所以可以通过页面来选取属性。

镜像：

us west	http://uswest.ensembl.org
us east	http://useast.ensembl.org
asia	http://asia.ensembl.org

使用listMarts/listDatasets/UseMart时，都可以加host="XXX"来指定镜像。

使用例子-getBM()

```
value <- c("SLC2A2")
attr <- c("hgnc_symbol","ensembl_transcript_id","chromosome_name","start_position","end_position")
mart = useMart("ensembl", dataset="hsapiens_gene_ensembl")
# attributes是我们希望获取的列，filter是要查的列，value则是实际查的值，即filter的实际值
ids <- getBM(attributes=attr,filters="hgnc_symbol",values=value,mart=mart)
# 如果mart那里直接写ensembl会报错：Error in martCheck(mart) : 找不到对象'ensembl'
# 可能是指定了mart，没有指定dataset的原因
```

一般的使用就是指定数据库，然后getBM查询
`getBM（ ）`只能从一个页面返回一个使用`attributes`的查询。如果要从多个页面组合`attributes`，则需要执行多个查询，然后合并它们。

```
info <- searchAttributes(mart,"sequence")
write.table(info, "search.txt",sep="\t",quote=FALSE)
# 为向量添加元素
attr <- c(attr,"5_utr_start","5_utr_end","3_utr_start","3_utr_end","transcript_start","transcript_end")
ids <- getBM(attributes=attr,filters="hgnc_symbol",values=value,mart=mart)

attr1 <- c("5_utr_start","5_utr_end","3_utr_start","3_utr_end","transcript_start","transcript_end")
target <- ids["ensembl_transcript_id"]
attr1 <- c(attr1, "ensembl_transcript_id")
exonAttr <- c("gene_exon","ensembl_exon_id", "exon_chrom_start", "exon_chrom_end", "rank")
attr2 <- c(attr1, exonAttr)
# filters是列表的时候不需要给values
ids <- getBM(attributes=attr2,filters=target, mart=mart)
write.table(ids, "search.txt", sep="\t", quote=FALSE)
```

attr2 <- c(attr2, c("cds_start", "cds_end"))

查的时发现没有

```
> ids <- getBM(attributes=attr2,filters=target, mart=mart)
Error in scan(file = file, what = what, sep = sep, quote = quote, dec = dec,  :
  3行没有14元素
> attr2
 [1] "5_utr_start"           "5_utr_end"             "3_utr_start"
 [4] "3_utr_end"             "transcript_start"      "transcript_end"
 [7] "ensembl_transcript_id" "gene_exon"             "ensembl_exon_id"
[10] "exon_chrom_start"      "exon_chrom_end"        "rank"
[13] "cds_start"             "cds_end"
```

删除元素

http://azaleasays.com/2008/02/06/r-remove-elements-in-a-vectorlist/

attr2 <- attr2[-14]
attr2 <- attr2[-13]
attr2 <- c(attr2, c("start_position","end_position"))
ids <- getBM(attributes=attr2,filters=target, mart=mart)
write.table(ids, "search.txt", sep="\t", quote=FALSE)

换一种写法，更合适貌似

ids <- getBM(attributes=attr2,filters="ensembl_transcript_id",values=target, mart=mart)
attr3 <- c(attr1, "genomic_coding_start","genomic_coding_end", "cds_start","cds_end")

Now final codes

> library(biomaRt)
> listDatasets(mart=ensembl)
Error in is(mart, "Mart") : 找不到对象'ensembl'
> ?listDatasets
> listEnsembl()
          biomart   version
1   ensembl   Ensembl Genes 94
2 ENSEMBL_MART_MOUSE   Mouse strains 94
3   snp Ensembl Variation 94
4   regulation Ensembl Regulation 94
> m <- useEnsembl(biomart="ensembl") # 这里表示引用了一个mart
> listDatasets(mart=m)

查找mart下面的数据集，可以用正则表达式，返回结果会给出名称，描述和版本号

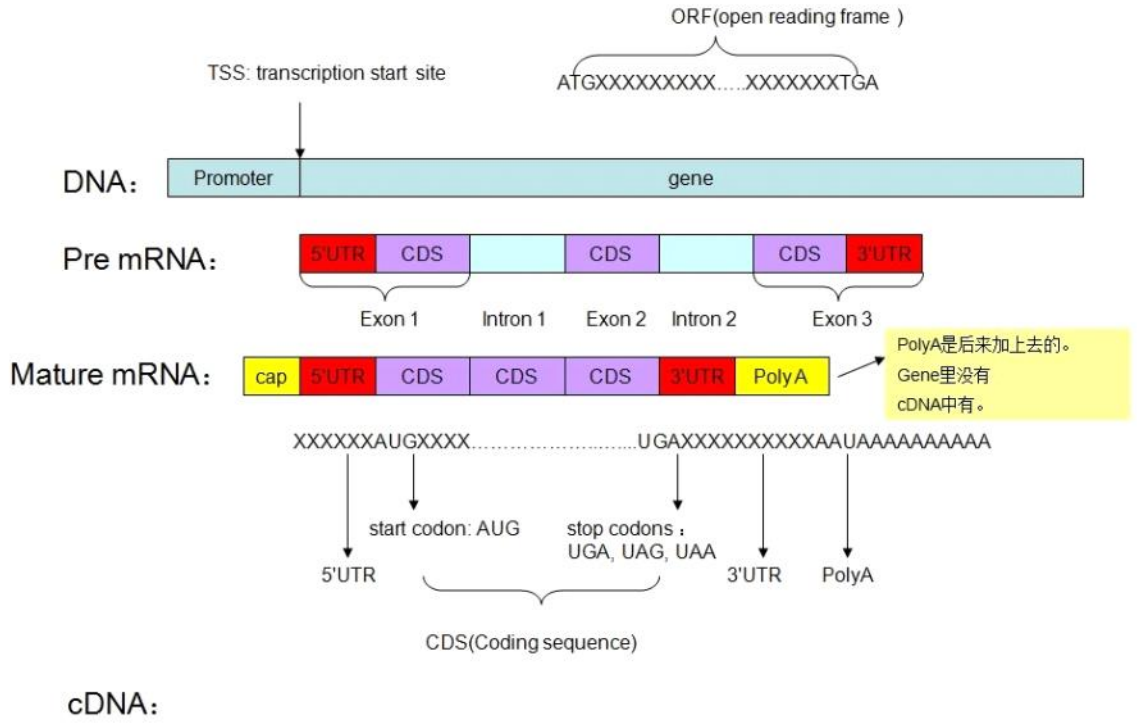
> searchDatasets(mart=m,pattern="sapiens")
              dataset   description   version
54 hsapiens_gene_ensembl Human genes (GRCh38.p12) GRCh38.p12
> searchDatasets(mart=m,pattern="(R|r)at")

HGNC: 对每一个人类基因，提供基因名称和符号（缩写）。唯一的。

commandArgs(),是R自带的参数传递函数，属于位置参数。

args=commandArgs(T)
print (args[1])

Rscript test.R 1



hsapiens_gene_ensembl

GTF/GFF

2018年11月1日 10:27

bedops里可以转换格式gff-bed
convert2bed -i gff < foo.gff3 > test.bed
或者
gff2bed < foo.gff3 > foo.bed
但是要注意，这里对格式要求高，gff中的分隔必须是单个tab，多个空格或多个tab会导致识别失败，没有结果。一定要好好注意：）
ref: https://bedops.readthedocs.io/en/latest/content/reference/file-management/conversion/gff2bed.html

参考文献：
https://www.jianshu.com/p/48b5a0972301
https://blog.csdn.net/sinat_38163598/article/details/72851239
https://zhuanlan.zhihu.com/p/36065699

GTF gene transfer format 对基因进行注释
GFF general feature format 对基因组进行注释
GTF是GFF的2.0版

这两种文件可以方便地相互转化：Cufflinks的gffread
gffread my.gff3 -T -o my.gtf
gffread merged.gtf -o- > merged.gff3

主要区别是第三列和第九列。
GFF：第三列可以用任意名称，第九列：属性名称和属性值用=连接，分号分隔，预定义的属性有ID（每行都要有）和Parent（当前特征是Parent特征的子集）。

```
Contig01  PFAM  gene  501  750  .  +  0  ID=geneA;Name=geneA
Contig01  PFAM  exon  501  650  .  +  2  ID=exonA1;Parent=geneA
Contig01  PFAM  exon  700  750  .  +  2  ID=exonA2;Parent=geneA
```

GTF：第三列是feature type，第九列以gene_id和transcript_id开头，属性名称与属性值用空格分开，每个属性之后要有分号

GFF使用#作为注释符号，文件开头一般会有两行表明版本及创建日期
##gff-version 3
##created 11/11/11

GTF文件举例：以tab分隔的9列，"."表示未知，留空。

seq_id	source	type	start	end	score	strand	phase	attributes
chr12	danRer10_refGene	CDS	25132758	25132785	0	+	0	gene_id "NM_199912"; transcript_id "NM_199914";
一般为chr或者	数据库/注	Gene/cDNA/mRNA/CDS			可能是序列相			最后一个属性之

scanfold编号	释机构	/mRNA/CDS等			似性比对的e-value或者基因预测的p-value			后也要有；

0表示该编码框的第一个密码子第一个碱基位于其5'末端；1表示该编码框的第一个密码子的第一个碱基位于该编码区外；2表示该编码框的第一个密码子的第一、二个碱基位于该编码区外；如果Feature为CDS时，必须指明具体值。

基因注释

2018年11月1日 13:11

参考文献：
https://vip.biotrainee.com/d/48-ncbi-refseq
http://www.genedenovo.com/news/340.html
https://zhuanlan.zhihu.com/p/36084586
https://www.biostars.org/p/16505/

RefSeq Gene注释：1个转录本对应1个RefSeq id，可以翻译成蛋白的转录本，会以NM开头，不能翻译的转录本,会以NR开头（如结构RNA，假基因转子等）

// RefSeq是被review过的，genbank经常收到提交的序列，并且会和EMBL和DDBJ交换数据，所以可能存在数据重复或不准的情况。RefSeq使用官方基因符号，GenBank则未必。
// RefSeq对人的信息注释最为全面
// 使用blast或者entrez搜索后，RefSeq的序列标识里会有ref。
// 如果RefSeq和GenBank有相同的序列，那么二者都会被保存，RefSeq的临时记录被review之后，会增加新的数据，但是仍然可以在entrez中输入各自的accession取得。
// mRNA就采用NM_开头的，基因组用NC_或者AC_开头的。NR_表示不编码的RNA或假基因序列，AF开头的表示克隆序列，BC开头的表示模板序列。还有很多，见链接2

Ensembl注释：ENSG开头的表示Ensembl gene_id，ENST表示Ensembl transcript
UCSC gene注释：ucsc的id，类似uc004cpf

Ensembl和RefSeq的关系：
The Ensembl protein coding gene and transcript set is based on the NCBI RefSeq set (manually curated entries only (NM and NP identifiers), not the predicted set (ie not XP and XM IDs)), along with UniProt proteins from Swiss-Prot and TrEMBL. Added to this is manual curation from the Havana group (at the Wellcome Trust Sanger Institute).

wget

2018年11月1日 15:55

参考文献：
http://www.cnblogs.com/peida/archive/2013/03/18/2965369.html

常用参数
-b 启动后转入后台
可以使用以下的命令查看下载进度
tail -f wget-log
-O 结果写入文件，不然默认会用url最后一个/之后的文字给文件命名
-a 结果追加
-i 下载输入文件中的 URL
-q 安静模式，没有输出
-c 断点续传
--spider不下载任何东西 但是可以测试网站是否可用
--tries=40 wget默认重试20次连接下载文件，如果网络不好/文件大，就增加
-T 设定响应超时的秒数
-w 两次尝试之间间隔的秒数
-random-wait 在下载之间等待0-2*WAIT秒
-limit-rate 限速
--reject=gif 过滤掉图片
-o XX.log 把下载信息放到日志文件而非终端

实例一：
wget -O XX.txt -c -b https://XXXX

实例二：伪装代理名称下载
在firefox里随便打开一个网页，然后ctrl+shift+i调出控制台
在“网络”那个page上随便找一个请求，查看原始头，可以看到User-Agent

所以

wget --no-check-certificate -O seq1.txt --header="user-agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:63.0) Gecko/20100101 Firefox/63.0" https://www.ncbi.nlm.nih.gov/nuccore/NM_000340.1

加--no-check-certificate是因为wget在使用https协议时，默认验证网站的安全证书，但是常常会验证失败导致“无法建立SSL连接”

实例三：下载指定格式的文件

wget -r -A.pdf url

实例四：ftp下载

使用wget匿名ftp下载：

wget ftp-url

使用wget用户名和密码认证的ftp下载

wget --ftp-user=USERNAME --ftp-password=PASSWORD url

实例五：

截取染色体信息

wget -c http://genome.ucsc.edu/cgi-bin/das/hg38/dna?segment=chr17:7676091,7676196

这里的hg38和染色体范围都可以替换，可以用的das有这些：

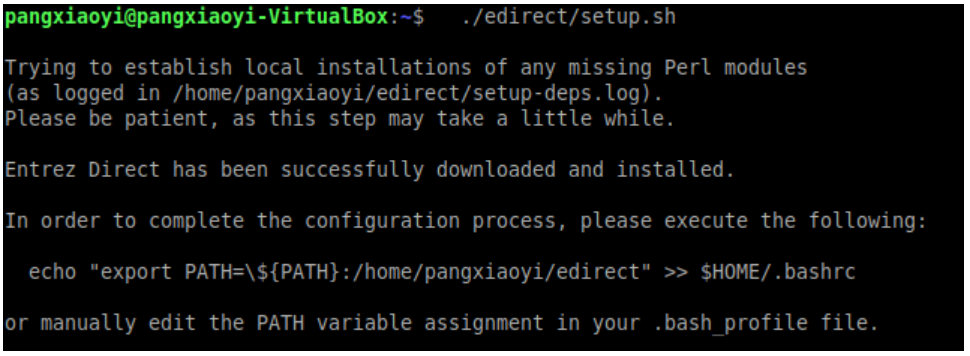
http://genome.ucsc.edu/cgi-bin/das/dsn

Entrez Direct

2018年11月2日 8:53

安装

```
cd ~
/bin/bash
perl -MNet::FTP -e \
'$ftp = new Net::FTP("ftp.ncbi.nlm.nih.gov", Passive => 1);
$ftp->login; $ftp->binary;
$ftp->get("/entrez/entrezdirect/edirect.tar.gz");'
gunzip -c edirect.tar.gz | tar xf -
rm edirect.tar.gz
builtin exit
export PATH=${PATH}:$HOME/edirect >& /dev/null || setenv PATH "${PATH}:
$HOME/edirect"
./edirect/setup.sh
```




```
echo "export PATH=\${PATH}:/home/pangxiaoyi/edirect" >> $HOME/.bashrc
```

JavaScript-tips

2018年11月6日 15:18

paste_input是textarea
document.getElementById("paste_input").style.color = "#A8A8A8";

href="static/example_depth.jpg"

引物设计常用命令

2018年11月12日 8:48

引物的产物也不能重叠，应该分池，防止pcr出大量短序列

从.fa中获取实际序列

bedtools getfasta -fo target_seq.fasta -fi hg38.fa -bed target.bed -s

blast制作数据库

makeblastdb -in hg38.fa -dbtype nucl -parse_seqids -out hg38

blast查询

blastn -query target_seq.fasta -db hg38 -out target_seq_result.txt -task blastn-short -max_target_seqs 1 -evalue 0.5 -outfmt 6

disable all the sequence masking:
blastn -query query.fa -db db.fa -dust no -soft_masking false -outfmt 6

切出引物顺便blast

python /home/pangxiaoyi/Desktop/Primer_Designer_Data_Formatation/pcrtiler_parser.py
cp p.txt /home/pangxiaoyi/app/blast/p.txt >/dev/null 2>&1
blastn -query p.txt -db target_seq -out target_seq_result.txt -task blastn-short -max_target_seqs 1 -evalue 0.001 -outfmt 6

hg38.fa 0-based, 前闭后开

ensembl uses 1-based coordinate system and UCSC uses 0-based...

BWA使用

./bwa index ref.fasta
./bwa mem ref.fasta probe.fasta > probe.sam

MPprimer

python CreateMPprimerInput_pxy.py -i slc2a2.fasta -o slc2a2.p3 -t pxy.template.txt
python MPprimer.py -i slc2a2.p3 -o slc2a2.txt -m 0bp

MFEprimer

target_400.txt是模板,fasta。产出.2bit .db .uni
../IndexDb.sh target_seq.fasta
输入一个引物seq跟模板比
../MFEprimer.py -i primer_seq.txt -d target_seq.fasta -o test.txt
这里的seq最好使用>1-F AGCT格式
python MFEprimerParser.py -i test.txt

MFEprimerV2操作

./IndexDb.sh seq.fasta
./MFEprimer.py -i probes.fasta -d seq.fasta -o result.txt
python MFEprimerParser.py -i result.txt

pooler

./pooler --dg --pools=?,2,sub- --genome=/media/sf_share/hg38.fa --amp-max=300 p1.txt --multiplx=multiplx.p1.txt 2>/dev/null

./pooler --dg --pools=?,2,sub- --genome=/media/sf_share/hg38.fa --amp-max=300 p1.txt 2>/dev/null

```
Do you want to use deltaG? (y/n): y

OK, please enter the deltaG settings:
Temperature: 60

Is that C ? (y/n): y

Magnesium concentration in nM (0 for no correction): 0

Monovalent cation (e.g. sodium) concentration in nM: 50

dNTP concentration in nM (0 for no correction): 0

Shall I count how many pairs have what deltaG range? (y/n): y
```

dG threshold: -7.5

python GetDesignPlan.py -i output.txt -o slc2a2.txt -g /media/sf_share/hg38.fa

探针设计资料

2018年11月19日 8:55

http://www.oebiotech.com/uploadfiles/files/20170214142806_2968.pdf
捕获测序最大的两个优势是可以自由设计目标区域和达到很高的覆盖度。
全基因组测序：信息更全面，数据量大，可能掺杂大量冗余信息。分析比较困难。
扩增子测序：（扩增子是经过人工扩增的DNA片段或RNA片段的扩增产物）就是先富集目标区域的DNA，然后再测序。适合大量样本，特定区域研究，快。
16S/18S/ITS扩增子测序即通过提取环境样品的DNA，选择合适的通用引物扩增16S/18S/ITS的某一或某几个区，使用Illumina测序将目的区域正反向读通，通过检测目的区域的序列变异和丰度，对环境样本物种分类及，丰度，种群结构，系统进化，群落比较等方面信息进行分析的研究方法。
目标序列捕获：
http://blog.sina.cn/dpool/blog/s/blog_64ade28a0101941d.html
一种是对目标片段做PCR；另一种是根据碱基互补配对杂交原理，根据目标基因组序列，设计与之完全互补的探针，将这些探针固定在某些支持物上，然后打断基因组DNA，加上接头（用于测序）后与探针杂交，洗脱未杂交上的DNA，回收目标DNA片段，直接建库测序。
根据杂交时状态不同，目标序列捕获可以分为固相杂交法（探针固定在固体上，如玻璃片。基因芯片是洗脱完，扫描成像获取每个探针的杂交信号；而固相杂交则要再洗脱出于探针杂交的DNA，用于后续测序）和液相杂交法（在溶液中，目标DNA片段和已带有生物素标记探针直接杂交，然后通过生物素亲和素的反应使目标DNA片段锚定在带有亲和素的微珠上。洗去非目标DNA，洗脱后，富集的DNA用于测序。液相杂交与固相杂交相比有两大优势：第一、杂交效率更高；第二、易于操作，时间短，便于自动化操作。使用这种方式的典型产品是安捷伦（Agilent）公司推出的SureSelect目标序列捕获系统）。

如果已知目标区域与参考基因组有较大出入，例如大片段的插入缺失，则不要捕获测序。

4. 应何时选择定制探针捕获？何时选择全外显子捕获？
人类和一些常见物种的全外显子捕获都有预设的探针产品，其中人类全外显子探针已优化过多个版本。如果目标区域很多且均为外显子（几十 M 以上），建议直接选择全外显子捕获，因为全外显子有设计更成熟的探针组和更低的成本，捕获效果更好。对于较小的目的片段，或者也关注外显子以外的区段，建议采用定制探针。

Benefits of Target Enrichment vs. Amplicon Sequencing

Target Enrichment	Amplicon Sequencing
Larger gene content, typically > 50 genes	Smaller gene content, typically < 50 genes
More comprehensive profiling for all variant types	Ideal for analyzing single nucleotide variants and insertions/deletions

	(indels)
More comprehensive method, but with longer hands-on time and turnaround time*	More affordable, easier workflow

* the turnaround time is for library prep assay time (DNA to finished library).

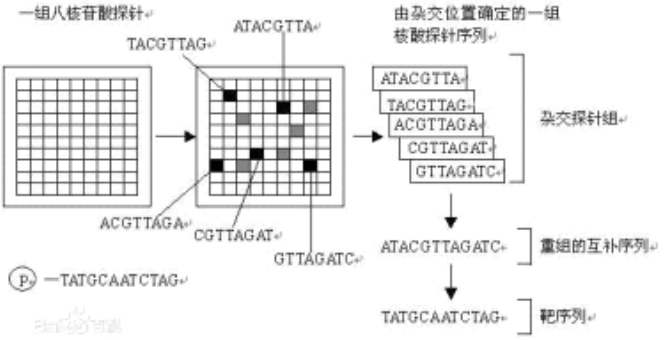
来自 <https://www.illumina.com/techniques/sequencing/dna-sequencing/targeted-resequencing/targeted-panels.html>

UMI

为了克服PCR扩增中引入的人源误差，在进行任何扩增之前，将UMIs（unique molecular indices）并入起始DNA材料中。每一个原始的DNA分子都带有独特的UMIs。在分析数据时，同样的UMIs就是来源于同一条DNA。真实的多样性，所有相同的UMIs连接的DNA中都有同一的多样性位点，若仅有其中的一个或者几个UMIs含有多样性位点，此多样性可判断为假阳性。

基因芯片

基因芯片(gene chip)的原型是 80 年代中期提出的。基因芯片的测序原理是杂交测序方法，即通过与一组已知序列的核酸探针杂交进行核酸序列测定的方法。在一块基片表面固定了序列已知的八核苷酸的探针。当溶液中带有荧光标记的核酸序列 TATGCAATCTAG，与基因芯片上对应位置的核酸探针产生互补匹配时，通过确定荧光强度最强的探针位置，获得一组序列完全互补的探针序列。据此可重组出靶核酸的序列。



分子知识补遗

2018年11月19日 14:47

https://www.wikiwand.com/en/Ultra-conserved_element
Ultra-conserved element
在至少两个物种中都存在的一段DNA区域。UCE常常是noncoding DNA，但是有一些可以转录，产生non-coding RNA。
人类基因组中约有481个。A database collecting genomic information about ultra-conserved elements (UCbase) that share 100% identity among human, mouse and rat is available at http://ucbase.unimore.it.
作用：1.某些与癌症相关：癌细胞中常常有UCE拷贝数目突变。2. 某些UCE靠近transcriptional regulators or developmental genes. 所以可能会起调节作用。

A Universal Probe Set for Targeted Sequencing of 353 Nuclear Genes from Any Flowering Plant Designed Using k-medoids Clustering

bedtools

2018年11月21日 9:36

==========bedops

bedops --ec --everything file

检查bed文件是否合法

subtract

bedtools subtract -a final.withutr.bed -b final.utr.bed -s > test121.result.bed

计算GC

bedtools nuc -fi hg38.fa -bed test.bed -s -seq > test.fasta

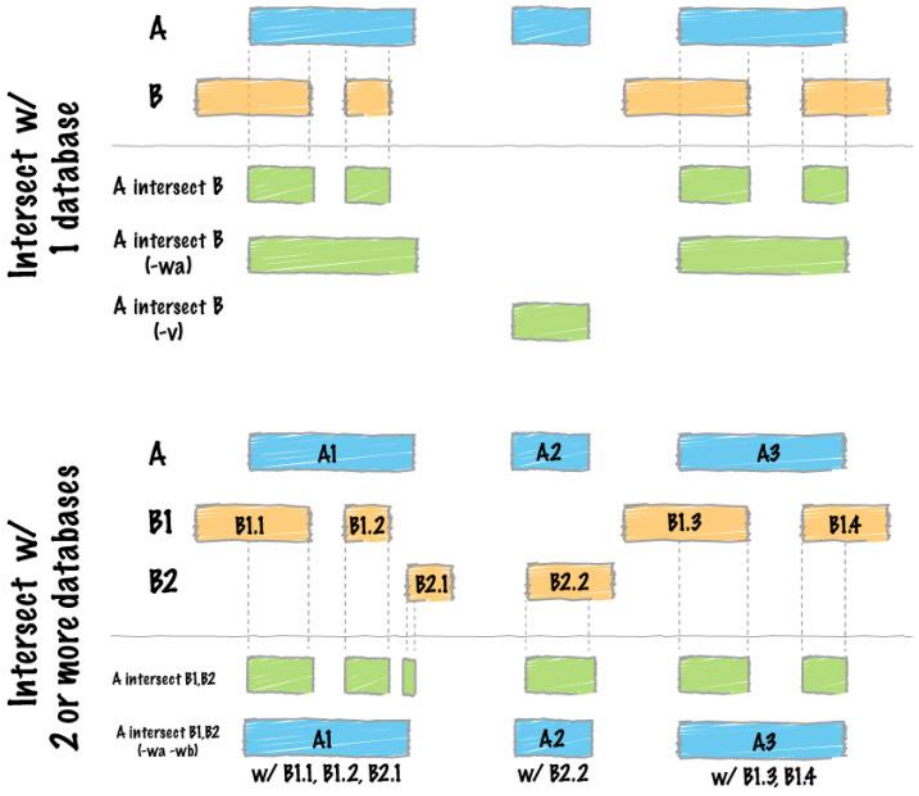
第8列是gc含量。

7_pct_at 8_pct_gc 9_num_A 10_num_C 11_num_G 12_num_T 13_num_N 14_num_oth 15_seq_len

sort -k 1,1 -k 2,2n infile > outfile

intersect

https://bedtools.readthedocs.io/en/latest/content/tools/intersect.html?highlight=intersect



参数：

-wa	Write the original entry in A for each overlap.
-wb	Write the original entry in B for each overlap. Useful for knowing what A overlaps. Restricted by -f and -r. -f Minimum overlap required as a fraction of A. Default is 1E-9 (i.e. 1bp).如-f 0.50就是B的item在A的item上必须map0.5以上才会报

	告。-r if -f is 0.90 and -r is used, this requires that B overlap at least 90% of A and that A also overlaps at least 90% of B
-loj	对于A中所有item都搜索，找不到的会标出.
-wao	统计A对Bovelap的bp数目，没有overlap的标0。-wo则不标没有overlap的。
-v	统计A中没有overlap的item
-s	只查找同一条链上的。-S只查找不同链上的。
-sorted	排完序再加这个会更快
-g	与-sorted一起用。跟一个示例文件，如chr1\tXXX\nchr2\tXXX\nchr3....帮助排序
-header	处理时会忽略header，加这个表示将header原样输出

实例: -wa -wb

```
$ cat A.bed
chr1  10  20
chr1  30  40

$ cat B.bed
chr1  15   20

$ bedtools intersect -a A.bed -b B.bed -wa -wb
chr1  10  20  chr 15  20
```

实例: -wao

```
$ cat A.bed
chr1    10    20
chr1    30    40

$ cat B.bed
chr1    15  20
chr1    18  25

$ bedtools intersect -a A.bed -b B.bed -wao
chr1    10    20    chr1    15  20  5
chr1    10    20    chr1    18  25  2
chr1    30    40    .      -1  -1  0
```

```
bedtools intersect -a probes.strand.sorted.bed -b targets.sorted.bed -s -sorted -wo > intersect.txt
bedtools intersect -a probes.strand.sorted.bed -b targets.sorted.bed -s -sorted -v > not_intersect.txt
```

coverage

https://bedtools.readthedocs.io/en/latest/content/tools/coverage.html?highlight=coverage

参数

-hist	Report a histogram of coverage for each feature in A as well as a summary histogram for _all_ features in A. Output (tab delimited) after each feature in A: 1) depth 2) # bases at depth 3) size of A

	4) % of A at depth 建议六列，strand放最后
-s	按strand计算
-d	会统计A的每个base在B中的覆盖度

实例：-a -b

```
$ cat A.bed
chr1  0   100
chr1  100 200
chr2  0   100

$ cat B.bed
chr1  10  20
chr1  20  30
chr1  30  40
chr1  100 200

$ bedtools coverage -a A.bed -b B.bed
chr1  0   100  3  30  100 0.3000000
chr1  100 200  1  100 100 1.0000000
chr2  0   100  0  0   100 0.0000000
```

这里前三列是A的item，然后是A在B中的overlap数量，然后是Aoverlap的长度，最后是这个overlap占A的比例。没有overlap的也会报告

实例：-hist

```
$ cat A.bed
chr1  0   100 b1  1  +
chr1  100 200 b2  1  -
chr2  0   100 b3  1  +

$ cat B.bed
chr1  10  20  a1  1  -
chr1  20  30  a2  1  -
chr1  30  40  a3  1  -
chr1  100 200 a4  1  +

$ bedtools coverage  -a A.bed -b B.bed -hist
chr1  0   100 b1  1  +  0  70  100  0.7000000
chr1  0   100 b1  1  +  1  30  100  0.3000000
chr1  100 200 b2  1  -  1  100 100  1.0000000
chr2  0   100 b3  1  +  0  100 100  1.0000000
all   0   170 300 0.5666667
all   1   130 300 0.4333333
```

这里是分别统计了覆盖度不同的比例。

merge

这样可以把首尾相连的合起来

```
bedtools -merge -i probes.strand.sorted.bed -s -d 1 > probes.merged.bed
```

getfasta

bedtools getfasta [-fo result.fasta] -fi hg19.fa -bed target.bed
第四列是name，第六列是-/+

getfasta 切序列的时候，在fa文件上是从0开始，前闭后开的。

-name	Use the “name” column in the BED file for the FASTA headers in the output FASTA file.
-tab	Report extract sequences in a tab-delimited format instead of in FASTA format.
-fullHeader	By default, only the word before first space/tab is used
-s	Force strandedness. If the feature occupies the antisense strand, the sequence will be reverse complemented. *Default: strand information is ignored.*
-split	Given BED12 input, extract and concatenate the sequences from the BED “blocks” (e.g., exons)

实例：-name

```
$ cat test.fa
>chr1
AAAAAAAACCCCCCCCCCCCCGCTACTGGGGGGGGGGGGGGGGG

$ cat test.bed
chr1 5 10 myseq

$ bedtools getfasta -fi test.fa -bed test.bed -name
myseq::chr1:5-10        AAACC
```

-tab

```
$ bedtools getfasta -fi test.fa -bed test.bed -tab
chr1:5-10       AAACC
```

-split

此时输入文件，是bed12，例如

```
chr1  164404  173864  ENST00000466557.1       0       -       173864  173864  0       6
387,59,66,216,132,112,  0,1479,3695,4644,8152,9348,
```

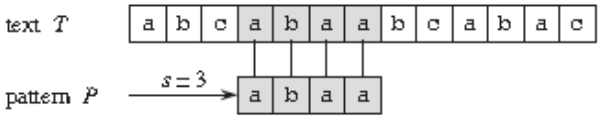
```
# use the UNIX fold command to wrap the FASTA sequence such that each line
# has at most 60 characters
$ bedtools getfasta -fi chr1.fa -bed genes.bed12 -split -name | \
    fold -w 60
```

后缀树

2018年11月27日　　16:12

在《字符串匹配算法》一文中，我们熟悉了字符串匹配问题的形式定义：

- **文本（Text）**是一个长度为 n 的数组 T[1..n]；
- **模式（Pattern）**是一个长度为 m 且 m≤n 的数组 P[1..m]；
- T 和 P 中的元素都属于有限的**字母表 Σ 表**；
- 如果 0≤s≤n-m，并且 T[s+1..s+m] = P[1..m]，即对 1≤j≤m，有 T[s+j] = P[j]，则说模式 P 在文本 T 中出现且位移为 s，且称 s 是一个**有效位移（Valid Shift）**。



比如上图中，目标是找出所有在文本 T = abcabaabcabac 中模式 P = abaa 的所有出现。该模式在此文本中仅出现一次，即在位移 s = 3 处，位移 s = 3 是有效位移。

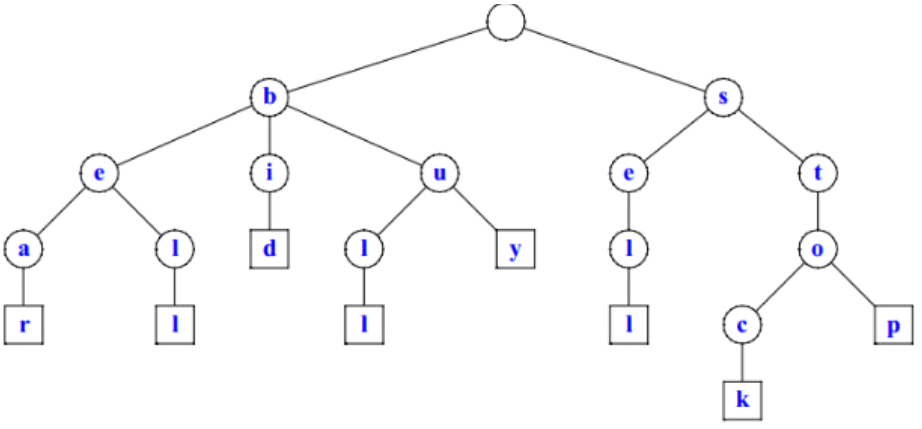
解决字符串匹配问题的常见算法有：

- 朴素的字符串匹配算法（Naive String Matching Algorithm）
- Knuth-Morris-Pratt 字符串匹配算法（即 KMP 算法）
- Boyer-Moore 字符串匹配算法

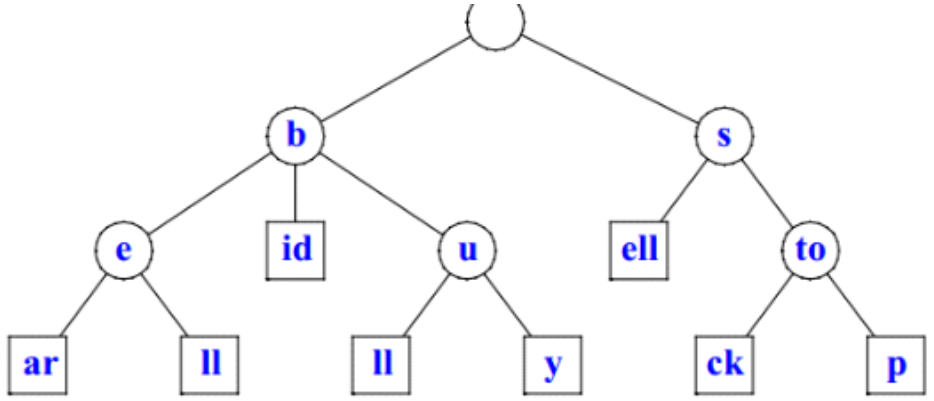
字符匹配算法通常分为preprocessing和matching两个步骤，算法的总运行时间是两者之和。一些匹配算法是对Pattern进行预处理以达到最优复杂度，而后缀树suffix tree则是对Text进行预处理。

字典树Trie：

将每个词的字符按顺序添加到树的节点上，这样从root开始遍历，就可以判断词是否在Trie中。举例：根据集合 {bear, bell, bid, bull, buy, sell, stock, stop} 所构建的 Trie 树：



由于ar、ll、id这种并没有其他子节点，所以可以合并节点，变成compressed trie



根据Text生成所有后缀的集合，将每个后缀作为一个单独的关键词，构建一棵compressed trie => 这就是suffix tree

隐式后缀树：一些后缀可能包含在其他后缀之中，如果不写出被包含的这些，则是implicit。如果在每个后缀的结尾加上一个特殊字符，如$或者#，则可以使得后缀保持唯一性。

性质：

1. 从根到树叶的路径与**S的后缀一一对应。即每条路径惟一代表了S的一个后缀；**

2. 每条边都代表一个非空的字符串；

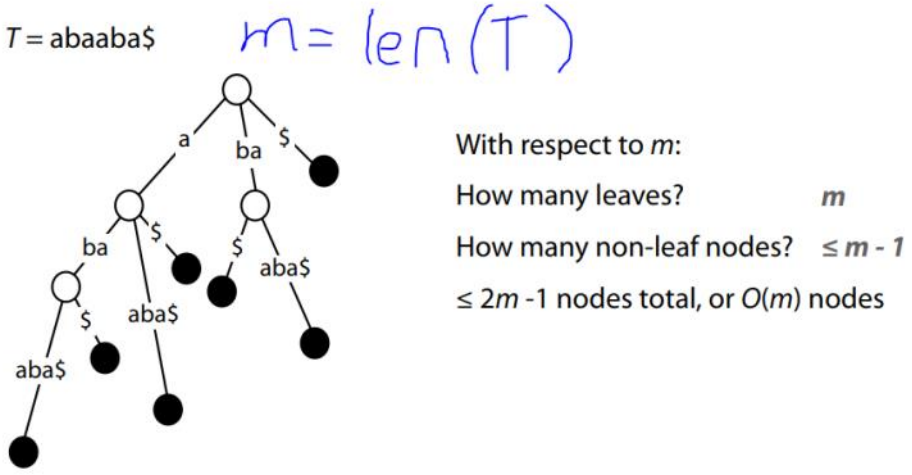
3. 所有内部节点（根节点除外）都有至少两个子节点。

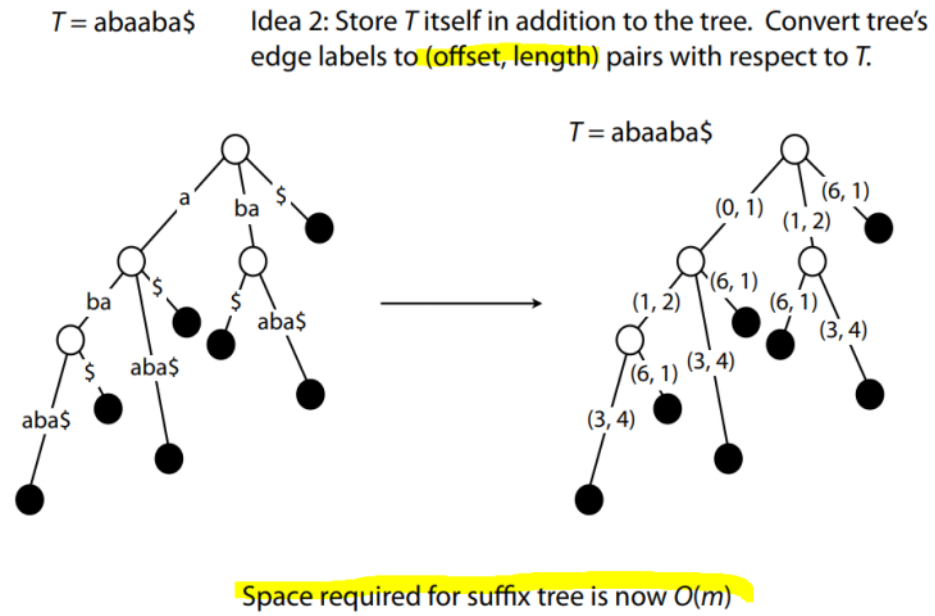
这里找longest common substring，找到最深的非叶节点且含#和$时，到它的path就是LCS

后缀树的应用

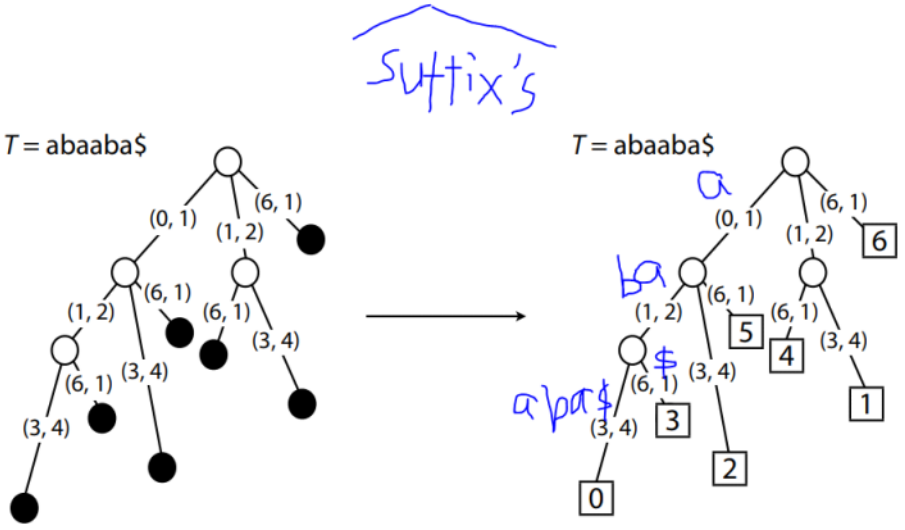
- **查找字符串 Pattern 是否在于字符串 Text 中**
  - 方案：用 Text 构造 后缀树 ，按在 Trie 中搜索字串的方法搜索 Pattern 即可。若 Pattern 在 Text 中，则 Pattern 必然是 Text 的某个后缀的前缀。
- **计算指定字符串 Pattern 在字符串 Text 中的出现次数**
  - 方案：用 Text+'$' 构造 后缀树 ，搜索 Pattern 所在节点下的叶节点数目即为重复次数。如果 Pattern 在 Text 中重复了 c 次，则 Text 应有 c 个后缀以 Pattern 为前缀。
- **查找字符串 Text 中的最长重复子串**
  - 方案：用 Text+'$' 构造 后缀树 ，搜索 Pattern 所在节点下的最深的非叶节点。从 root 到该节点所经历过的字符串就是最长重复子串。
- **查找两个字符串 Text1 和 Text2 的最长公共部分**
  - 方案：连接 Text1+'#' + Text2+'$' 形成新的字符串并构造 后缀树 ，找到最深的非叶节点，且该节点的叶节点既有 '#' 也有 '$'。
- **查找给定字符串 Text 里的最长回文**
  - 回文指："abcdefgfed" 中对称的字符串 "defgfed"。
  - 回文半径指：回文 "defgfed" 的回文半径 "defg" 长度为 4，半径中心为字母 "g"。
  - 方案：将 Text 整体反转形成新的字符串 Text2，例如 "abcdefgfed" => "defgfedcba"。连接 Text+'#' + Text2+'$' 形成新的字符串并构造 后缀树 ，然后将问题转变为查找 Text 和 Text1 的最长公共部分。

http://www.cs.jhu.edu/~langmea/resources/lecture_notes/suffix_trees.pdf





Suffix tree: leaves hold offsets



Node depth: 从root到这个node经过的edge数量

label depth: total length of edge labels for edges on path from root to node

python实现：O(m2) time, O(m) space

或者用Ukkonen's algorithm，O(m) time and space

```
In [2]: stree = SuffixTree('there would have been a time for such a word')

In [3]: stree.hasSubstring('nope')
Out[3]: False

In [4]: stree.hasSubstring('would have been')
Out[4]: True

In [5]: stree.hasSubstring('such a word')
Out[5]: True

In [6]: stree.hasSuffix('would have been')
Out[6]: False

In [7]: stree.hasSuffix('such a word')
Out[7]: True
```

```
class SuffixTree(object):

    class Node(object):
        def __init__(self, lab):
            self.lab = lab # label on path leading to this node
            self.out = {}  # outgoing edges; maps characters to nodes

    def __init__(self, s):
        """ Make suffix tree, without suffix links, from s in quadratic time
            and linear space """
        s += '$'
        self.root = self.Node(None)
        self.root.out[s[0]] = self.Node(s) # trie for just longest suf
        # add the rest of the suffixes, from longest to shortest
        for i in xrange(1, len(s)):
            # start at root; we' ll walk down as far as we can go
            cur = self.root
            j = i
            while j < len(s):
                if s[j] in cur.out:
                    child = cur.out[s[j]]
                    lab = child.lab
                    # Walk along edge until we exhaust edge label or
                    # until we mismatch
                    k = j+1
                    while k-j < len(lab) and s[k] == lab[k-j]:
                        k += 1
                    if k-j == len(lab):
```

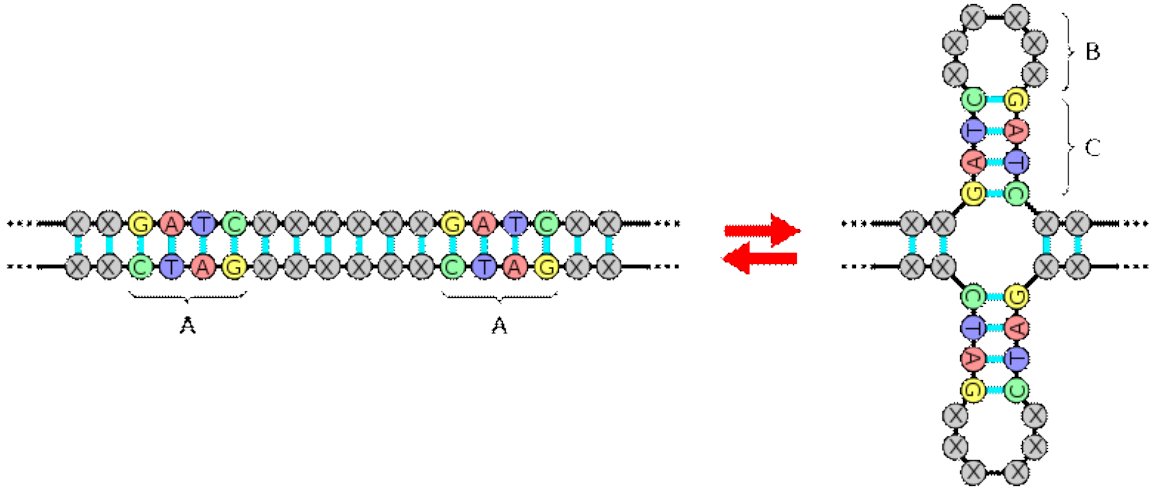
```
                cur = child # we exhausted the edge
                j = k
            else:
                # we fell off in middle of edge
                cExist, cNew = lab[k-j], s[k]
                # create "mid": new node bisecting edge
                mid = self.Node(lab[:k-j])
                mid.out[cNew] = self.Node(s[k:])
                # original child becomes mid's child
                mid.out[cExist] = child
                # original child's label is curtailed
                child.lab = lab[k-j:]
                # mid becomes new child of original parent
                cur.out[s[j]] = mid
        else:
            # Fell off tree at a node: make new edge hanging off it
            cur.out[s[j]] = self.Node(s[j:])

def followPath(self, s):
    """ Follow path given by s.  If we fall off tree, return None.  If we
        finish mid-edge, return (node, offset) where 'node' is child and
        'offset' is label offset.  If we finish on a node, return (node,
        None). """
    cur = self.root
    i = 0
    while i < len(s):
        c = s[i]
        if c not in cur.out:
            return (None, None) # fell off at a node
        child = cur.out[s[i]]
        lab = child.lab
        j = i+1
        while j-i < len(lab) and j < len(s) and s[j] == lab[j-i]:
            j += 1
        if j-i == len(lab):
            cur = child # exhausted edge
            i = j
        elif j == len(s):
            return (child, j-i) # exhausted query string in middle of edge
        else:
            return (None, None) # fell off in the middle of the edge
    return (cur, None) # exhausted query string at internal node
```

```
    def hasSubstring(self, s):
        """ Return true iff s appears as a substring """
        node, off = self.followPath(s)
        return node is not None

    def hasSuffix(self, s):
        """ Return true iff s is a suffix """
        node, off = self.followPath(s)
        if node is None:
            return False # fell off the tree
        if off is None:
            # finished on top of a node
            return '$' in node.out
        else:
            # finished at offset 'off' within an edge leading to 'node'
            return node.lab[off] == '$'
```

来自 <http://nbviewer.jupyter.org/gist/BenLangmead/6665861>

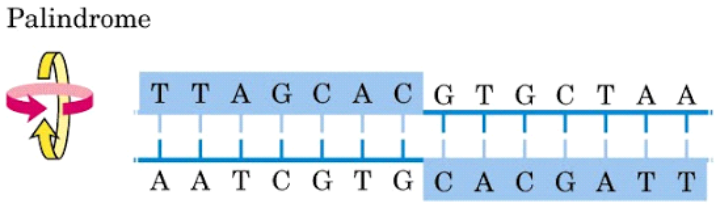


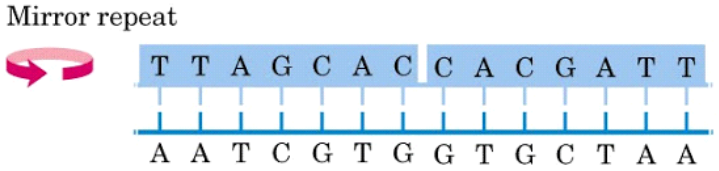
1. DNA序列有三種重複方式

A. **Inverted repeat**：兩股5端到3端的序列相同，前後的DNA序列互補造成迴文(Palindrome)序列的產生。可與特定的限制酶作用

同股上有inverted repeat的DNA互相配對，形成hairpin（髮夾）型，與另一股已配對的DNA形成十字形cruciform

功用：可做為蛋白質辨識DNA，或與DNA結合的信號





資料來源：[南伊利諾伊大學醫學院在斯普林菲爾德]

B. **Mirror repeat**：在箭頭中間放個鏡子使兩邊的序列呈鏡像排列

C. **Direct repeat**：序列重複出現

所以将互补配对链与原链连接，查找回文即可。

来自 <https://smallcollation.blogspot.com/2013/08/dnarepeat-sequences-of-dna.html#gsc.tab=0>

repeat_and_hairpin_opt.py codes-reading

repeatDataFrame是position取[1,length]的结果。

提取里面长度大于minWindowSize的片段，排序，就是repeat_multiArray_new。再去掉beginPosition是1或者endPosition是length的，就是repeat_list。

计时：

```
import timeit
start1 = timeit.default_timer()
print("time used: " + str(timeit.default_timer() - start1))
```

合并集合：

```
def merge(times):
    saved = list(times[0])
    for st, en in sorted([sorted(t) for t in times]):
        if st <= saved[1]:
            saved[1] = max(saved[1], en)
        else:
            yield tuple(saved)
            saved[0] = st
            saved[1] = en
    yield tuple(saved)
```

times=[[1,2],[2,3],[1,4]]这样

python 3.x
print(args, file=f1)
python 2.x
print >> f1, args

```
>>> import pandas as pd
>>> d = {'col1': [1, 2], 'col2': [3, 4], 'col3':['1st','2nd']}
>>> df = pd.DataFrame(data=d)
```

shape是数组，给出维度信息

```
>>> df.shape
(2, 3)
>>> df
   col1  col2 col3
0     1     3  1st
1     2     4  2nd
```

pandas

2018年10月19日 11:23

Python tricks

正则表达式

```
tmp_dict.append(item[re.search(r'^\d*',item).end():])
s = '6637197'
re.search(r'^\d+',s).span()
```

输出

```
(0, 7)
```

查找list中某元素的所有位置：

```
[idx for idx, e in enumerate(s) if e==2]
```

```
def foo(a, b, c, **args):
    print "a = %s" % (a,)
    print "b = %s" % (b,)
    print "c = %s" % (c,)
    print args

foo(a="testa", d="excess", c="testc", b="testb", k="another_excess")
```

输出：

```
a = testa
b = testb
c = testc
{'k': 'another_excess', 'd': 'excess'}
```

```
def foo(a, b, c, *args):
    print "a = %s" % (a,)
    print "b = %s" % (b,)
    print "c = %s" % (c,)
    print args

foo("testa", "testb", "testc", "excess", "another_excess")
```

输出：

```
a = testa
b = testb
c = testc
('excess', 'another_excess')
```

也就是说，*identifier是按tuple接收多余的参数的，**identifier是按字典接收参数的

计时

在 Unix 系统中，建议使用 time.time()，在 Windows 系统中，建议使用 time.clock()。

```
import time
start = time.clock()
... do something
elapsed = (time.clock() - start)
```

又或者

```
start = time.time()
... do something
elapsed = (time.time() - start)
```

获取当前目录下某后缀的文件个数：

```
DIR = os.getcwd()
seq_count = sum(".json" in name for name in os.listdir(DIR) if os.path.isfile(os.path.join(DIR, name)))
print seq_count
```

把pycharm放到桌面（其他软件类似）

```
sudo leafpad /usr/share/applications/Pycharm.desktop
```

模仿里面其他的，写：）

然后把图标放在桌面就好

string

```
s.replace("ha", "wo")
```

str.isalnum() 所有字符都是数字或者字母

str.isalpha() 所有字符都是字母

str.isdigit() 所有字符都是数字

str.islower() 所有字符都是小写

str.isupper() 所有字符都是大写

str.istitle() 所有单词都是首字母大写，像标题

str.isspace() 所有字符都是空白字符、\t、\n、\r

转小写：str.lower()返回小写字符串

判断是否为浮点数

```
import re
float_number = str(input( “Please input the number:” ))
```

调用正则

```
value = re.compile(r'^[-+]?[0-9]+\.[0-9]+$')
result = value.match(float_number)
if result:
    print "Number is a float."
else:
    print "Number is not a float."
```

作者：杜鲁门

来源：CSDN

原文：https://blog.csdn.net/bug_moving/article/details/52886412

tuple

```
p = 1,2
type(p) == type(())
# p是元组类型
```

defaultdict

```
from collections import defaultdict

gene_dict = defaultdict(int)
# 这样再使用不存在的key取出来的值就是0了
```

全局变量

如果要改变其值，改之前要写global a，然后后面就可以引用了

NumPy

它的arange类似range，但是对于列表操作更快

```
from numpy import *
```

这样就不用每次都带前缀了：）

```
array
array([1,2,3])
array([[1,2],[3,4]]) # 要用[]包起来。按行
>>> b = array([[1,2],[3,4], [5,6]])
>>> b.size
6
>>> b.shape
(3, 2)
x = arange(start, stop, step) #[start, stop)
# 第三个参数指明个数。都是等宽的。包含stop
>>> c = linspace(1,3,4)
>>> c
array([1.        , 1.66666667, 2.33333333, 3.        ])
```

特殊的array

```
>>> zeros((3,3))
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])

>>> diag([1,2,3])
array([[1, 0, 0],
       [0, 2, 0],
       [0, 0, 3]])

>>> random.rand(3,3)
array([[0.93884711, 0.55048793, 0.1110919 ],
       [0.61443117, 0.56528858, 0.33408106],
       [0.82741974, 0.87371625, 0.15731454]])
```

M[1] 或M[1,:]取行

M[:,1]第一列

M[1,:] = 0 # 整行取0

Pandas tricks

DataFrame写出到txt

```
out1 = open('hairpinTest.txt','w')
with pd.option_context('display.max_rows', None, 'display.max_columns', None):
    print(hairpinDataFrame,file=out1)
out1.close()
```

DataFrame写出到xlsx

```
writer = pd.ExcelWriter('hairpinInfo.xlsx')
hairpinDataFrame.to_excel(writer, sheet_name='hairpins')
writer.save()
```

```
import pandas as pd
```

df = pd.read_table('probes.blast.txt', sep=',') # 当然read_table默认分割符是\t

df.shape # 返回维度

df.columns=['A','b','c'] # 给出列名

df[df['perc'].isin([df['perc'].min()])] # 查找某列最小值在的列

```
df['val'].max()
```

df[df['val']==222].count() # 会统计各列的数目

```
temp = df[df['val']==222]
temp.columns=['qid','sid','perc_ident','len','mismatch','gap','q_start','q_end','s_start','s_end','e_val','bit_
score']
temp[(temp['mismatch']==0)&(temp['gap']==0)&(temp['perc_ident']==100.0) & (temp['len']==120) &
(temp['q_start']==1) & (temp['q_end']==120)].count()
```

// 7847， 也就是所有行都满足

确定是否有重复行

```
import pandas as pd
df = pd.read_table(os.getcwd() + "/temp/GS_xgen-pan-cancer-targets.bed", sep='\t')
df.columns = ['chr','start','end','name','len','strand']
result=df.name.value_counts()
result[result > 1]
>>> len(result[result > 1])
0
```

read_csv实例

```
probes_all_df = pd.read_csv('filter1st.txt', keep_default_na = False, names = ['qid', \
                'sid','perc_ident','len','mismatch','gap','q_start','q_end','s_start', \
                's_end','e_val','bit_score'], sep = "\t", converters = {'start': str, 'end': str})
```

dropna()
DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
默认：如果本行有一个NA，则丢掉本行
drop_duplicates()
DataFrame.**drop_duplicates**(*subset=None, keep='first', inplace=False*) [sourc

Return DataFrame with duplicate rows removed, optionally only considering certain columns

Parameters:	**subset** : *column label or sequence of labels, optional* Only consider certain columns for identifying duplicates, by default use all of the columns **keep** : *{'first', 'last', False}, default 'first'* • `first` : Drop duplicates except for the first occurrence. • `last` : Drop duplicates except for the last occurrence. • False : Drop all duplicates. **inplace** : *boolean, default False* Whether to drop duplicates in place or to return a copy
Returns:	**deduplicated** : *DataFrame*

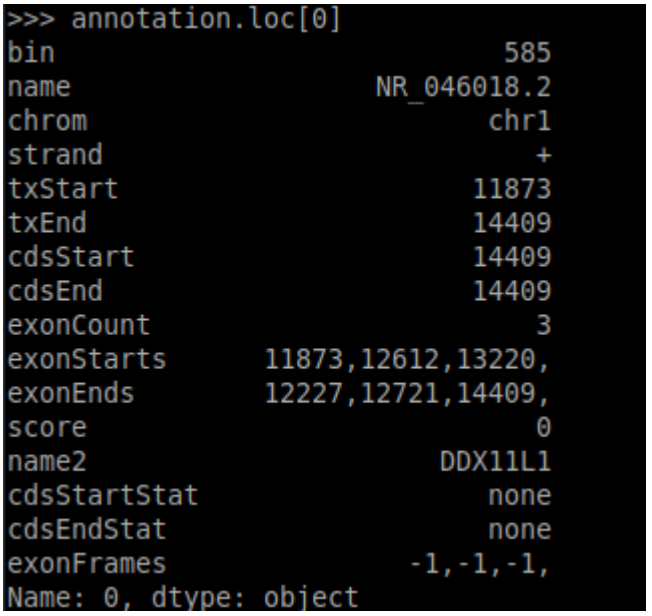
```
>>> df_single_level_cols
     weight height
cat       0      1
dog       2      3
>>> df_single_level_cols.stack()
cat  weight    0
     height    1
dog  weight    2
     height    3
dtype: int64
```

读csv
# sep must be RIGHT
probes_all_df = pd.read_csv(result, keep_default_na=False, header=None,
names="query_title,hit_id,hsps".split(","), sep=",")
#最简单的读入，默认会将第一行作为列名
annotation = pd.read_csv(os.getcwd() + "/../data/hg38annotation_UCSC_NCBIRefSeq.txt", sep= "\t")
读取一行：
1.直接切
annotation[0:1]

```
>>> annotation[0:1]
   bin         name chrom     ...     cdsStartStat  cdsEndStat  exonFrames
0  585  NR_046018.2  chr1     ...             none        none   -1,-1,-1,

[1 rows x 16 columns]
```

不可以用annotation[0]，因为这样会报错。同时，这里会显示不全。
2.用loc
annotation.loc[0]



如果annotation.loc[0:1]，这样切出的是0和1两行，这个与直接切片不同。对于不连续的list，可以写annotation.loc[[1,3]]这样切出行1和3

3.用iloc，特别是删除或筛选数据后，索引号与原来不一致时

loc[]与iloc[]方法之间还有一个巨大的差别，那就是loc[]里的参数是对应的索引值即可，所以参数可以是整数，也可以是字符串。而iloc[]里的参数表示的是第几行的数据，所以只能是整数

Bokeh 这个框架，比起 D3.js，它的可视化选项相对较少。因此，目前来看 Bokeh 无法挑战 D3.js 的霸主地位。而且 Bokeh 过于依赖 python 的数值计算库，并非一个纯前端的框架，使得它的使用范围也小于 D3.js。而在纯 python 的数值计算领域，也已经有 matplotlib 这种提供了与 Matlab 一模一样的接口的数据可视化库，Bokeh 的适用场景也并不多。

但是，它非常适合嵌入 Flask 或者 Django 的程序中，非常好用，速度也很快。

作者：柴柴土

链接：https://www.jianshu.com/p/8b4f17950777

來源：简书

破解-背景知识

2018年12月11日 17:26

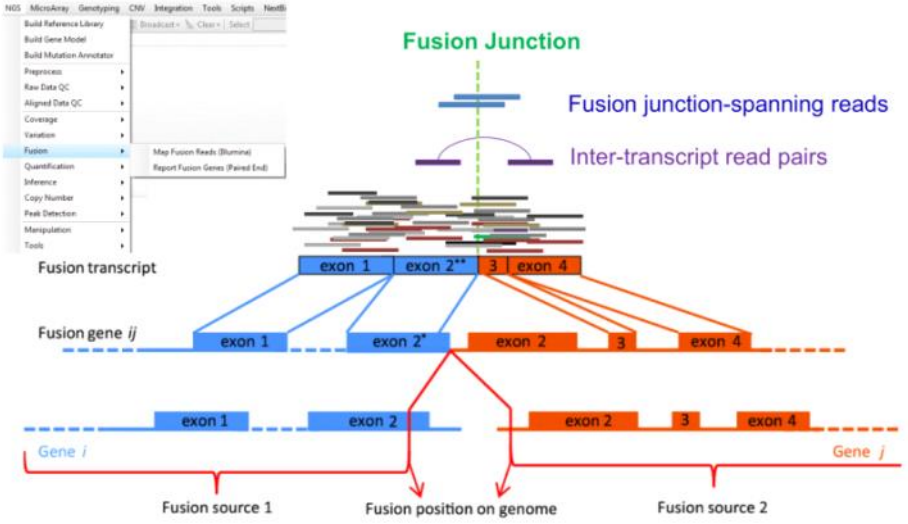
融合基因指两个或多个基因联合起来，一起转录形成一个转录本。
只有少数是因为染色体易位等原因，在DNA水平就联合在一起了，大多数只是在转录时共同转录，所以常利用RNA-seq来研究：只要检测到一个转录本来源于不同的基因，即可识别。
软件：
tophat-fusion, soap-fusion, fusionmap, chimerascan

来自 <https://www.cnblogs.com/xudongliang/p/7596855.html>

嵌合RNA（chimeric RNA）由两个或两个以上基因的外显子组成，并具有编码新蛋白改变细胞表型的潜力。
形成原因：反式剪接tran-splicing：不同基因的外显子剪接后相互连接。
嵌合RNA及其产物通常是导致肿瘤发生的原因之一，不过在正常细胞中也可能有低水平表达。
可视化工具：bioconductor包的chimeraviz

融合发现工具
deFuse、EricScript、InFusion、JAFFA、FusionCatcher、FusionMap、PRADA、SOAPfuse、STAR-FUSION

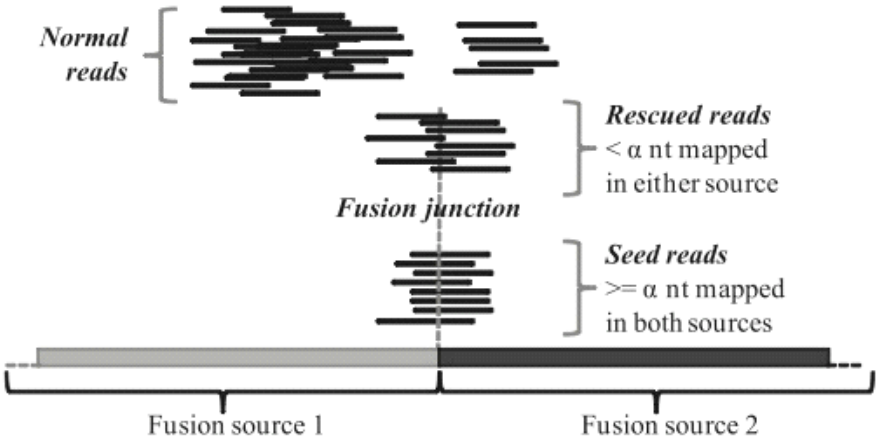
FusionMap（好像过时了）



通过两种方式来检测融合基因：

1) 对于没有mapping 上的基因组的unmapped reads, 通过识别 Fusion junction-spanning reads 来识别融合基因；这部分reads 在mapping的时候由于插入缺失的限制，没有能够mapping 上任何一个基因；

2）对于mapping 上基因组的reads, 通过识别 Inter-transcript read pairs 来识别融合基因，这部分reads 的R1端和R2端分别mapping 到不同的基因



在fusionmap 中，假定融合基因由2个基因组成，对于没能比对上基因组的Fusion Junction-spanning reads, 又分为两类：设定一个阈值，如果这条reads 在两个基因中比对上的长度都大于阈值，就属于seed reads; 如果在任意一个基因中比对上的长度小于阈值，就属于Rescued reads;

known disease associations, as identified by the HGMD, OMIM, and ClinVar databases.

来自 <http://space.internet.genscript.com/display/IBRBIA/181212_Leo_toXiaoyi_aboutTargetDesigner>
HGMD
The Human Gene Mutation Database

OMIM
Online Mendelian Inheritance in Man
Online Catalog of Human Genes and Genetic Disorders

ClinVar
NCBI的，genomic variation and its relationship to human health

冬姐的设计需求

2018年12月26日 16:12

1. 如何把结果打印到指定表格之中
2. 输入文件希望放在含中文字符的路径中
3. 非法数据也提取出来，仅供检查

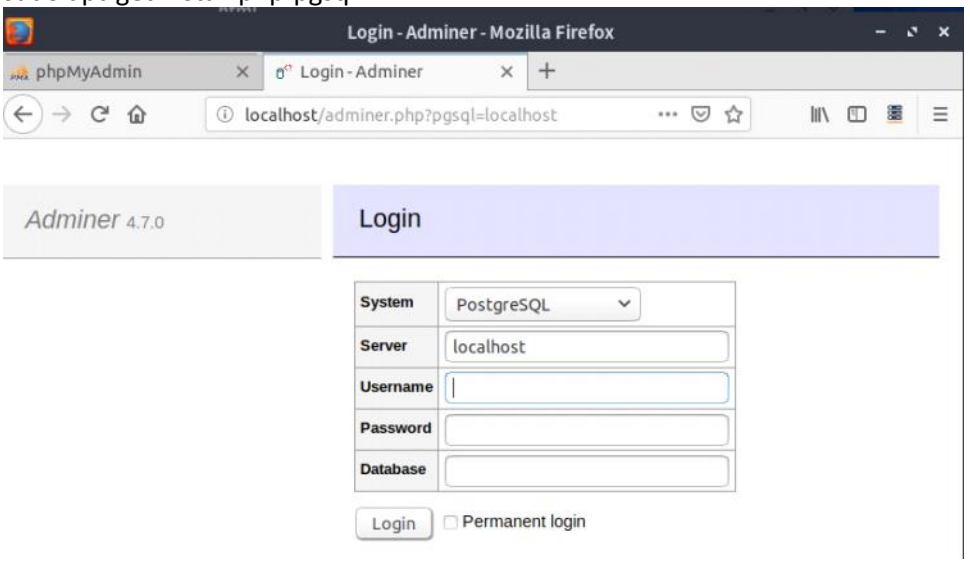
==========
打印到word

PHP&adminer

2019年1月16日 15:41

ubuntu安装php环境
sudo apt-get install php
此时我安装到的是php7.2，所以验证是否安装成功要用php7.2 -v
sudo apt-get isntall libapache2-mod-php
安装phpmyadmin时选apache2，可能还要输入mysql的密码，这里用的是pxy7896
sudo apt-get install phpmyadmin
创建phpMyAdmin快捷方式：
sudo ln -s /usr/share/phpmyadmin /var/www/html
启用Apache mod_rewrite模块：
sudo a2enmod rewrite
重启服务：
service apache2 restart
测试：浏览器访问：
http://localhost/phpmyadmin
用户名root+密码pxy7896即可访问mysql数据库

使用一个叫adminer-4.7.0-en.php的工具查看postgresql数据库（图形化界面）
将这个脚本拷贝到/var/www/html下改名adminer.php
然后在浏览器输入localhost/adminer.php
输入用户名密码即可。不输入数据库名称的话，可以进去再选择。
此时会报错，说没有pgsql、pdoXXXX等，所以要
sudo apt-get install php-pgsql



这个同样可以进入mysql里，只要system选好了就可以了。

IGV使用

2019年1月28日 17:01

条件：java 1.8，不支持1.9，内存>2G
安装：
windows下，下载，解压，双击运行
linux下，下载，解压，sh运行
view-> preference 勾选show soft-clipped bases 即可展示错误的reads(mottled)
show center line之后ctrl+s即可按base排序，方便归类
import genomes是导入参考序列
import file则可以导入bed文件、 bam文件。bam文件需要bai文件作为其索引文件，这个可以由igv生成。

DL4j

2019年2月18日 8:32

安装：

1. Java >= 1.7
2. Apache Maven

(Maven is a dependency management and automated build tool for Java projects.)
官网下载解压，然后把bin的路径放到环境变量的PATH
mvn -v

```
C:\Users\dell>mvn -v
Apache Maven 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-18T02:33:1
4+08:00)
Maven home: D:\Program Files\maven\apache-maven-3.5.4\bin\..
Java version: 1.8.0_121, vendor: Oracle Corporation, runtime: D:\Program Files\J
ava\jdk1.8.0_121\jre
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 7", version: "6.1", arch: "amd64", family: "windows"
```

3. Update git

git clone git://git.kernel.org/pub/scm/git/git.git

4. IntelliJ community (IDE)

2018年9月19日 11:27

>hg19_dna range=chr3:170714137-170744768 5'pad=0 3'pad=0 strand=- repeatMasking=none

bedtools getfasta -fo t.fasta -fi hg38.fa -bed t.bed

makeblastdb -in target_seq.fasta -dbtype nucl -parse_seqids -out target_seq
blastn -query test/p.txt -db target_seq -out target_seq_result.txt -task blastn-short -max_target_seqs 1 -evalue 0.01 -outfmt 6

mRNA join(303..324,8357..8449,8525..8573,12249..12511,
16898..17022,19717..19832,20875..21037,21508..21695,
24300..24404,27814..27915,28584..28787,28877..30639)

来自 <https://www.ncbi.nlm.nih.gov/nuccore/NC_000003.12?report=genbank&from=170996341&to=171026979&strand=true>

MPprimer

./CreateMPprimerInput.py -i long.fasta -o test.p3 -t template.txt
../MPprimer.py -i target1.p3 -o MPprimer_result1.mp -m 0bp
../MPprimer.py -i target2.p3 -o MPprimer_result2.mp -m 0bp

./MPprimer.py -i test.p3 -o test.mp -m 0bp

python CreateMPprimerInput_pxy.py -i slc2a2.fasta -o slc2a2.p3 -t pxy.template.txt
python MPprimer.py -i slc2a2.p3 -o slc2a2.txt -m 0bp

MFEprimer
target_400.txt是模板,fasta。产出.2bit .db .uni
../IndexDb.sh target_seq.fasta
输入一个引物seq跟模板比
../MFEprimer.py -i primer_seq.txt -d target_seq.fasta -o test.txt
这里的seq最好使用>1-F AGCT格式
python MFEprimerParser.py -i test.txt

tcsh
$PRIMUX_DIR/run_Primux -d Results -f target_seq.fasta -P options_primers -p options_probes -g 1 -c 1 | & tee log.stdout

./msre-htprimer --refseq-gtf-file ref_input/hg38_annotation/hg38_refseqgenes.gtf --restriction-enzyme-file test_data/type-II-enzymes.txt --primer3-param-file test0925/primer3-param-file.txt --gene-distance 5000 --primer-pair-to-return 1 --primer-type msre --genome-name Human --genome-assembly hg38 --genome-target gene --min-perfect-match 15 --output-dir test0925/msre_output_dir --target-file test0925/target_msre.bed --genome-fasta-file ref_input/hg38_fasta/hg38.fa

2018年7月27日 13:20

makeblastdb -in brca1_ref.txt -dbtype nucl -parse_seqids -out test/brca1_assembly
blastn -query test.fasta -db brca1_assembly -out seq.blast

blastn -query test.fasta -db brca1_assembly -outfmt 6

rm *.class
javac *.java -Xlint:unchecked

touch MANIFEST.MF // 添加如下语句到该文件中，然后保存
Manifest-Version: 1.0
Main-Class: com.linky.Server // 里面包含 main 入口函数

jar -cvfm server.jar MANIFEST.MF PCRTiler
java -jar server.jar

../MFEprimer.py -i msre_seq.txt -d brca2.rna -o msre_MFEprimer_result.txt

必须标成1-F, 1-R成对儿才行，不然有时候会显示no primer pairs founded
./pooler --dg --suggest-pools --pools=2 --genome=./ --amp-max=200 primux_seq.txt

./pooler --dg --genome=wang.2bit --amp-max=100 primer_18.fasta

Degenerate base means more than one base possibility at a particular position, this is usually the case when a DNA sequence is derived from amino acid sequence with codon based sequence.

R=A+G
Y=C+T
M=A+C
K=G+T
S=G+C
W=A+T
H=A+T+C
B=G+T+C
D=G+A+T
V=G+A+C
N=A+C+G+T

简并度：
5- NAG S G NGC DTTA NCABK-3
简并度=4 × 2 × 4 × 3 × 4 × 3 ×2=2304（所代表的碱基数量相乘）

简并引物的设计 • 从蛋白到核酸， 请注意：
– 尽量选择简并度低的氨基酸区域为引物设计区
– 充分注意物种对密码子的偏好性

– 引物不要终止于简并碱基，对大多数氨基酸残基来 说，意味着引物的 3末端不要位于密码子的第三位。

– 在简并度低的位置，可用次黄嘌呤(dI)代替简并碱基。

- fna = FastA format file containing Nucleotide sequence (DNA)
- gbff = Genbank Genome file containing genome sequence and annotation
- gff = general feature format containing genomic regions, the "genes, transcripts, etc"
- faa = FastA format file containing Amino-acid sequence (Protein, peptide)
- gpff = Genbank Protein file containing protein sequence and annotation

  See https://www.ncbi.nlm.nih.gov/genome/doc/ftpfaq/ for more explanation.

AGCACCGACCAGCAGACATCCGAGCACACCTGGGCA
TCCGAGCACACCTGGGCA
AGGGCACGAGCTATGGCA
Sequence ID: NG_011513.1**Length:** 216617**Number of Matches:** 1
Related Information
Range 1: 208811 to 208828

来自 <https://blast.ncbi.nlm.nih.gov/Blast.cgi#alnHdr_224809353>

Range 1: 212268 to 212285

来自 <https://blast.ncbi.nlm.nih.gov/Blast.cgi>

TGCCCCACGGCTACGAGA
GCGGGCAAAGTTGGCCCA

EGFR
ERBB2
IFNAR1
LAMA3
SLC2A1

EGFR
>f
GGGGTGCAGGAGAGGAGA
>r
TTTGCGGCAGACCAGGCA

hg38_knownGene_uc003tqh.4_range=chr7:55018034-55157951

英语表达补缺

2018年8月10日 11:34

1. 当使用such as时，读者默认后面接着的是不完整的列举，所以不要加and so on 或etc
2. 如果要全部列举，改用namely或者i.e.然后加逗号
3. for example (e. g.)表示泛泛地举几个例子,并没有囊括所有的实例,其中就已经包含“等等”,如果再加etc. 或and so on,就画蛇添足了。
4. etc.。是et cetera的缩写,意思是“等等”,相当于“and so on”。可用来列举事物,若要列举人,则需用et al. 或用and others

NB:〔用于提醒读者留意某则信息〕

邮件常用

Thank you for contacting us.
Thank you for your prompt reply.
Thank you for getting back to me.
Thank you for providing the requested information.
Thank you for all your assistance.

Thank you for your kind cooperation.
Thank you for your attention to this matter.
Thank you for your understanding/consideration.
Thank you again for everything you've done.

Perl Tutorial

2018年7月24日 10:30

基础

$perl -v
$perl -e <perl code> # Unix/Linux
eg: $perl -e 'print "Hello World\n"'
IDE: Padre or eclipse+epic

第一个例子：

#每条语句之后要加；
解释器路径
#!/usr/bin/perl
print "Hello, world\n";
写完之后要添加可执行权限，然后运行：
$chmod 0755 hello.pl
$ ./hello.pl

#括号只用来提示优先级，可以没有
print("Hello, world\n");

=begin和=cut之间是多行注释，会被忽略
=begin
喵喵喵
=cut

引号内的空白不会被忽略，引号外面的空白忽略。

双引号内能转义字符和变量，单引号照常输出。

perl标识符由英文字母或下划线开头

Here文档

Here文档又称作heredoc、hereis、here-字串或here-脚本，是一种在命令行shell（如sh、csh、ksh、bash、PowerShell和zsh）和程序语言（像Perl、PHP、Python和Ruby）里定义一个字串的方法。

使用概述：
1.必须后接分号，否则编译通不过。
2.END可以用任意其它字符代替，只需保证结束标识与开始标识一致。（这里使用的是EOF）
3.结束标识必须顶格独自占一行
4.开始标识可以不带引号号或带单双引号，不带引号与带双引号

效果一致，解释内嵌的变量和转义符号，带单引号则不解释。
5.当内容需要内嵌引号（单引号或双引号）时，不需要加转义符。

```
#!/usr/bin/perl

$a = 10;
$var = <<"EOF";
这是一个 Here 文档实例，使用“双引号”。
可以在这输如字符串和变量。
例如：a = $a
EOF
print "$var\n";

#这里会原样输出
$var = <<'EOF';
这是一个 Here 文档实例，使用单引号。
例如：a = $a
EOF
print "$var\n";
```

Perl数据类型

1.标量

```
$myFirst = "123";
# 八进制
$first = 047;
# 十六进制
$second = 0x1f;
$third = $first + $second;
# 输出39 + 31 = 70
print "$first + $second = $third\n"
```

浮点寄存器通常不能精确地存贮浮点数，从而产生误差，在运算和比较中要特别注意。指数的范围通常为-309到+308。

```
$value = 9.01e+21 - 9.01e+21 + 0.01; #输出0.01
$value = 9.01e+21 + 0.01 - 9.01e+21; #输出0
```

2.数组

```
# 用@定义
@arr = (1, 2, 3);
# 用 $数组名[下标] 操作
Print "\$arr[0] = $arr[0]\n";
@arr = qw/23 "是" 数组/; #使用qw//定义数组
print "$arr[-1]\n"; # 反向读取，输出的是数组
Print "$arr[-2]\n"; # 输出的是"是"。原样。
```

```
@var_20 = (1..20); # [1,20] closed interval and 0-based
print "@var_20\n"; # 输出整个数组

@array = (1,2,3);
$array[50] = 4;
# 返回的是数组物理大小而非元素个数
$size = @array;
$max_index = $#array;
print "数组大小:  $size\n"; # 51
print "最大索引: $max_index\n"; # 50

# push在数组结尾添加一个元素
push(@sites, "baidu");
print "2. \@sites  = @sites\n";
#unshift在数组开头添加一个元素
unshift(@sites, "weibo");
print "3. \@sites  = @sites\n";
#pop删除数组末尾的元素
pop(@sites);
print "4. \@sites  = @sites\n";
#shift移除数组开头的元素
shift(@sites);
print "5. \@sites  = @sites\n";

#[]切割数组，连续的可以用[3..5]
@sites2 = @sites[3,4,5];

# splice替换数组元素
splice(@arr, offset[, length [, list]])
# offset是起始index，length是长度，list是用来替换的列表

# split将字符串转化为数组。支持正则表达式
$var_string = "www-runoob-com";
@string = split('-', $var_string);

# 数组转为字符串
$string1 = join( '-', @string );

数组排序用sort,默认ascii升序

# 数组嵌套
@numbers = (@odd, @even);
```

3.哈希

```
# 用%定义
%h=('a'=>1, 'b'=>2);
```

```
%data = ('google', 45, 'runoob', 30, 'taobao', '喵');
$data{'amazon'} = 'amazon.com';#直接索引来添加值
%data = (-google=>'google.com', -runoob=>'runoob.com');
# 用-代替引号
$val = $data{-google};
%data = (1=>123,2=>321);
print "$data{1}\n";

# 用 $哈希名{key} 操作，输出 $data{'taobao'} = 喵
print "\$data{'taobao'} = $data{'taobao'}\n";

%data = (-taobao => 45, -google => 30, -runoob => 40);
 # 将hash提取到array
@array = @data{-taobao, -runoob};
# 返回所有的key和value
@names = keys %data;
@urls = values %data;
exists 函数来判断key是否存在
# 删除哈希中的元素
delete $data{'taobao'};
```

变量

使用 use strict 语句让所有变量需要强制声明类型。

perl根据上下文决定变量类型：

```
@names = ('google', 'runoob', 'taobao');

@copy = @names;  # 复制数组
$size = @names;  # 数组赋值给标量，返回数组元素个数


$str = "hello"."world"; # 字符串连接
print "$str";

$num = 4 * 5;
$mix = $str . $num;
print "$mix"; #会先计算结果，再转为字符串贴上 helloworld20

# 注意这些只是特殊字符串。类似def，双引号中不转义
print "文件名 ". __FILE__ . "\n";
print "行号 " . __LINE__ ."\n";
print "包名 " . __PACKAGE__ ."\n";
```

v字符串

一个以 v 开头,后面跟着一个或多个用句点分隔的整数,会被当作一个字串文本。

```
$smile = v65.110.66;
print "$smile\n"; # ascii码解码，即AnB
```

条件语句

```
If(){}elseif(){}else{}

unless(boolean_expression 1){
  # 在布尔表达式 boolean_expression 1 为 false 执行
}
elsif( boolean_expression 2){
  # 在布尔表达式 boolean_expression 2 为 true 执行
}
elsif( boolean_expression 3){
  # 在布尔表达式 boolean_expression 3 为 true 执行
}
else{
  # 没有条件匹配时执行
}

use Switch;
 switch(argument){
   case 1          { print "数字 1" }
   case "a"        { print "字符串 a" }
   case [1..10,42]  { print "数字在列表中" }
   case (\@array)   { print "数字在数组中" }
   case /\w+/      { print "正则匹配模式" }
   case qr/\w+/    { print "正则匹配模式" }
   case (\%hash)    { print "哈希" }
   case (\&sub)    { print "子进程" }
   else           { print "不匹配之前的条件" }
}
```

循环语句

```
# 当条件为 false 时执行循环。 当条件为 true 时，程序流将继续执行紧接着循环的下一条语句。
until(condition)
{
   statement(s);
}

@list = (2, 12, 36, 42, 51);
 # 执行foreach 循环列表
foreach $a (@list){
    print "a 的值为: $a\n";
}
```

```
do
{
   statement(s);
}while( condition );
```

continue 语句可用在 while 和 foreach 循环中，在条件语句再次判断前执行。

next 语句

停止执行从next语句的下一语句开始到循环体结束标识符之间的语句，转去执行continue语句块，然后再返回到循环体的起始处开始执行下一次循环。

last：类似break。last语句之后的语句不再执行，continue语句块也不再执行。

redo 语句直接转到循环体的第一行开始重复执行本次循环，redo语句之后的语句不再执行，continue语句块也不再执行。

go to label/expr/name

运算符

字符串比较：lt gt le ge eq ne cmp(左边的串大于右边的返回1，相等返回0，小于返回-1)

位运算符：& | ^异或 ~ << >>

逻辑运算符：and && or || not

引号运算：q{}为字符串添加单引号，qq{}双引号，qx{}反引号

```
# 使用 unix 的 date 命令执行
$t = qx{date};
print "qx{date} = $t\n";
$c = "-" x 3; # 字符串重复
```

时间日期

1、time() 函数：返回从1970年1月1日起累计的秒数

2、localtime() 函数：获取本地时区时间

3、gmtime() 函数： 获取格林威治时间

引用

一个标量类型可以指向变量、数组、哈希、子程序等，只要在被指向的变量名前加\即可。

```
$coderef = \&handler; # 函数引用
$globref = \*foo; # GLOB句柄引用
$aref= [ 1,"foo",undef,13 ]; #匿名数组的引用，可嵌套
my $aref = [
        [1, 2, 3],
```

```
    [4, 5, 6],
    [7, 8, 9],
]
$href= { APR =>4, AUG =>8 }; #匿名哈希引用
$coderef = sub { print "Runoob!\n" };#匿名子程序引用
```

取消引用可以根据不同的类型使用 $, @ 或 % 来取消

$$r	@$aref	%$href

```
ref($r) # 返回引用类型
&$coderef(%hash) # 使用引用调用函数。其实还是解引用了
```

子程序

参数使用特殊数组@_按引用方式调用，在子程序内部index，要用$_[0]这种。

```
# 定义求平均值函数
sub Average{
  # 确定length
  $n = scalar(@_);
  $sum = 0;
  # 循环数组
  foreach $item (@_){
    $sum += $item;
  }
  $average = $sum / $n;
  print '传入的参数为 : ',"@_\n";          # 打印整个数组
  print "第一个参数值为 : $_[0]\n";        # 打印第一个参数
  print "传入参数的平均值为 : $average\n"; # 打印平均值
}

# 调用函数
Average(10, 20, 30);
```

需要传入标量和数组参数时，需要把列表放在最后一个参数上

如果没有使用 return 语句，则子程序的最后一行语句将作为返回值。

my 操作符用于创建词法作用域变量，通过 my 创建的变量，存活于声明开始的地方，直到闭合作用域的结尾。
闭合作用域指的可以是一对花括号中的区域，可以是一个文件，也可以是一个 if, while, for, foreach, eval字符串。

我们可以使用 local 为全局变量提供临时的值，在退出作用域后将原来的值还回去。

local 定义的变量不存在于主程序中，但存在于该子程序和该子程序调用的子程序中。定义时可以给其赋值。

```
# 全局变量
$string = "Hello, World!";

sub PrintRunoob{
   # PrintHello 函数私有变量
   local $string;
   $string = "Hello, Runoob!";
   # 子程序调用的子程序
   PrintMe();
   print "PrintRunoob 函数内字符串值：$string\n";
}
```

state操作符功能类似于C里面的static修饰符，state仅能创建闭合作用域为子程序内部的变量。state可以声明标量、数组、哈希。但在声明数组和哈希时，不能对其初始化。

```
use feature 'state';
 sub PrintCount{
   state $count = 0; # 初始化变量
   print "counter 值为：$count\n";
   $count++;
}
```

XML Tutorial

2018年7月31日 10:46

XML被设计用来传输和存储数据，焦点是数据的内容。独立于软件和硬件。
XML没有预定义的标签，自定义。
XML 的优势之一，就是可以在不中断应用程序的情况下进行扩展。

XML声明

```
<?xml version="1.0" encoding="UTF-8"?>
<note> # 根元素
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

所有的 XML 元素都必须有一个结束标签
XML 标签对大小写敏感，XML 的属性值必须加引号（单双都可，单引号可以内嵌双引号，也可以使用实体引用）
文档中的连续空格不会被合并。
XML以LF存储换行。
特殊字符使用实体引用来避免歧义

&lt;	<
&gt;	>
&amp;	&
&apos;	'
&quot;	"

注释

```
<!-- This is a comment -->
```

标签最好用单词_单词形式命名，简短有描述性。

尽量避免使用属性，如果信息看起来像数据，使用元素。属性不能包含多个值/不能包含树结构/不容易扩展。
仅使用属性来提供与数据本身无关的信息（元数据。比如note的id，没有数据意义，只是为了标识，则应该作为属性）。

验证XML文档：

a. XML DTD

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
```

```
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Note.dtd

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

b. XML Schema

```
<xs:element name="note">

<xs:complexType>
<xs:sequence>
<xs:element name="to" type="xs:string"/>
<xs:element name="from" type="xs:string"/>
<xs:element name="heading" type="xs:string"/>
<xs:element name="body" type="xs:string"/>
</xs:sequence>
</xs:complexType>

</xs:element>
```

XSLT 是在浏览器显示 XML 文件之前，先把它转换为 HTML

DTD在XML源文件中，则只写DOCTYPE声明即可。

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</note>
```

如果DTD是额外文件

Vi/Vim

2018年8月6日　16:12

三种模式：

命令模式command mode：用户刚启动即进入，按i切换到输入模式，x删除当前光标所在处的字符，：切换到底线命令模式。

数字+箭头（或者hjkl左下上右）=多次移动

ENTER，回车键，换行

DEL，删除键，删除光标后一个字符

HOME/END，移动光标到行首/行尾

Page Up/Page Down，上/下翻页

G　移动到这个档案的最后一行(常用)

nG　n 为数字。移动到这个档案的第 n 行。

n<Enter>　n 为数字。光标向下移动 n 行(常用)

输入模式Insert mode：可以使用以下按键：

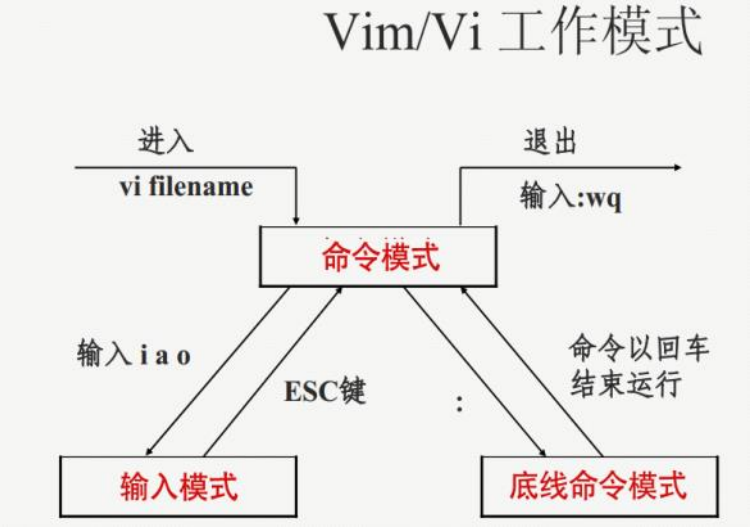
字符按键以及Shift组合，输入字符

Insert，切换光标为输入/替换模式，光标将变成竖线/下划线

ESC，退出输入模式，切换到命令模式

底线模式last line mode：

:wq



Lua

2018年8月24日 8:57

单行注释：--开始

多行注释：--[[--]]

- **变量和流程控制**

num = 42 --所有变量是double

字符串单双引号皆可

u = [[Double brackets
start and end
multi-line strings.]]

t = nil -- Undefines t; Lua has garbage collection.

没定义的变量就是nil，不会报错。

foo = anUnknownVar --此时foo = nil

aBoolValue = false

-- 只有 nil and false 是 falsy; 0 和 '' 是true!

-- 'or' and 'and' are short-circuited.

-- This is similar to the a?b:c operator in C/js:

ans = aBoolValue and 'yes' or 'no' --> 'no'

变量默认都是global的，使用local关键字才能变局部变量。

变量命名用驼峰式：除第一个单词外，其他单词首字母大写。

do/end标记，没有++和+=

```
while num < 50 do
    num = num + 1
end
```

-- ~=是不等于，==是等于

```
if num > 40 then
  print('over 40')
elseif s ~= 'walternate' then
  io.write('not over 40\n') -- defaults to stdout.
else
  thisIsGlobal = 5
  local line = io.read() -- reads next stdin line.
  print('Winter is coming, ' .. line) -- 字符串连接用..
end
```

```
[1,100]
karSum = 0
for i = 1, 100 do
   karSum = karSum + i
end
```

```
begin, end[, step]
fredSum = 0
for j = 100, 1, -1 do fredSum = fredSum + j end

repeat
   print('the way of the future')
   num = num - 1
until num == 0
```

- **函数**

function/end闭合

这里有一个匿名函数

```
function adder(x)
   return function (y) return x + y end
end
```

调用

```
a1 = adder(9)
print(a1(16)) --结果是25

-- Returns, func calls, and assignments all work
-- with lists that may be mismatched in length.
-- Unmatched receivers are nil;
-- unmatched senders are discarded.
x, y, z = 1, 2, 3, 4 --4是取不到的
function bar(a, b, c)
   print(a, b, c)
   return 4, 8, 15, 16, 23, 42
end
x, y = bar('zaphod')
```

会打印zaphod nil nil，值会变为x=4, y=8

函数可以用变量表示，可以使用local关键字

下面两个同义

```
local function g(x) return math.sin(x) end
local g; g  = function (x) return math.sin(x) end
-- Trig funcs work in radians, by the way.
-- Calls with one string param don't need parens:
print 'hello'  -- Works fine.
```

- **Tables**

tables 是lua唯一复杂的数据结构。hash-lookup dict

```
t = {key1 = 'val1', key2 = false}
print(t.key1) --打印值
t.newKey = {} --添加键值对
t.key2 = nil --删掉key2这个键值对

-- Literal notation for any (non-nil) value as key:
```

```
u = {['@!#'] = 'qbert', [{}] = 1729, [6.28] = 'tau'}
print(u[6.28])  -- prints "tau"
a = u['@!#'] -- 引用
-- A one-table-param function call needs no parens:
function h(x) print(x.key1) end
h{key1 = 'Sonmi~451'}  -- Prints 'Sonmi~451'.

for key, val in pairs(u) do  -- Table iteration.
  print(key, val)
end

 _G is a special table of all globals.

list不是单独的数据结构，是从1开始index的table
v = {'val1', 'val2', 1.21, 'gg'}
for i = 1 , #v do
   print(v[i])
end

metatable: 对table复用metamethod函数
defaultFavs = {animal = 'gru', food = 'donuts'}
myFavs = {food = 'pizza'}
setmetatable(myFavs, {__index = defaultFavs})
eatenBy = myFavs.animal  -- works! thanks, metatable

metamethod:
__add(a, b) __sub __mul __div __mod __pow __unm这个是变为相反数
__concat连接两个字符串 __len __eq __lt __le __index(a, b)是for a.b
__newindex(a, b, c)是for a.b=c  __call(a, ...)是for a(...)

类和模块的参考：
----------------------------------------------------
-- 3.2 Class-like tables and inheritance.
----------------------------------------------------

-- Classes aren't built in; there are different ways
-- to make them using tables and metatables.

-- Explanation for this example is below it.

Dog = {}                                  -- 1.

function Dog:new()                        -- 2.
  newObj = {sound = 'woof'}               -- 3.
  self.__index = self                     -- 4.
  return setmetatable(newObj, self)       -- 5.
end

function Dog:makeSound()                  -- 6.
  print('I say ' .. self.sound)
```

```
end

mrDog = Dog:new()                       -- 7.
mrDog:makeSound()  -- 'I say woof'      -- 8.

-- 1. Dog acts like a class; it's really a table.
-- 2. function tablename:fn(...) is the same as
--    function tablename.fn(self, ...)
--    The : just adds a first arg called self.
--    Read 7 & 8 below for how self gets its value.
-- 3. newObj will be an instance of class Dog.
-- 4. self = the class being instantiated. Often
--    self = Dog, but inheritance can change it.
--    newObj gets self's functions when we set both
--    newObj's metatable and self's __index to self.
-- 5. Reminder: setmetatable returns its first arg.
-- 6. The : works as in 2, but this time we expect
--    self to be an instance instead of a class.
-- 7. Same as Dog.new(Dog), so self = Dog in new().
-- 8. Same as mrDog.makeSound(mrDog); self = mrDog.

----------------------------------------------------

-- Inheritance example:

LoudDog = Dog:new()                     -- 1.

function LoudDog:makeSound()
  s = self.sound .. ' '                 -- 2.
  print(s .. s .. s)
end

seymour = LoudDog:new()                 -- 3.
seymour:makeSound()  -- 'woof woof woof'   -- 4.

-- 1. LoudDog gets Dog's methods and variables.
-- 2. self has a 'sound' key from new(), see 3.
-- 3. Same as LoudDog.new(LoudDog), and converted to
--    Dog.new(LoudDog) as LoudDog has no 'new' key,
--    but does have __index = Dog on its metatable.
--    Result: seymour's metatable is LoudDog, and
--    LoudDog.__index = LoudDog. So seymour.key will
--    = seymour.key, LoudDog.key, Dog.key, whichever
--    table is the first with the given key.
-- 4. The 'makeSound' key is found in LoudDog; this
--    is the same as LoudDog.makeSound(seymour).

-- If needed, a subclass's new() is like the base's:
function LoudDog:new()
  newObj = {}
  -- set up newObj
  self.__index = self
  return setmetatable(newObj, self)
```

```
end

----------------------------------------------------
-- 4. Modules.
----------------------------------------------------


--[[ I'm commenting out this section so the rest of
--   this script remains runnable.
-- Suppose the file mod.lua looks like this:
local M = {}

local function sayMyName()
  print('Hrunkner')
end

function M.sayHello()
  print('Why hello there')
  sayMyName()
end

return M

-- Another file can use mod.lua's functionality:
local mod = require('mod')  -- Run the file mod.lua.

-- require is the standard way to include modules.
-- require acts like:     (if not cached; see below)
local mod = (function ()
  <contents of mod.lua>
end)()
-- It's like mod.lua is a function body, so that
-- locals inside mod.lua are invisible outside it.

-- This works because mod here = M in mod.lua:
mod.sayHello()  -- Says hello to Hrunkner.

-- This is wrong; sayMyName only exists in mod.lua:
mod.sayMyName()  -- error

-- require's return values are cached so a file is
-- run at most once, even when require'd many times.

-- Suppose mod2.lua contains "print('Hi!')".
local a = require('mod2')  -- Prints Hi!
local b = require('mod2')  -- Doesn't print; a=b.

-- dofile is like require without caching:
dofile('mod2.lua')  --> Hi!
dofile('mod2.lua')  --> Hi! (runs it again)

-- loadfile loads a lua file but doesn't run it yet.
f = loadfile('mod2.lua')  -- Call f() to run it.
```

```
-- loadstring is loadfile for strings.
g = loadstring('print(343)')  -- Returns a function.
g()  -- Prints out 343; nothing printed before now.

--]]
```

PCRTiler CodeReading

2018年10月16日 10:46

subclass extends superclass
Java不允许多继承。继承只能继承一个类，但implements可以实现多个接口。
class A extends B implements C,D,E {} （class 子类名 extends 父类名 implenments 接口名）
Interface
1.实现一个接口就要实现该接口的所有方法（抽象类除外）
2.接口中的方法都是抽象的。
3.多个无关的类可以实现同一个接口，一个类可以实现多个无关的接口。

######栗子######
A a = new B(); 这条语句，实际上有三个过程：
(1) A a;
将a声明为父类对象，只是一个引用，未分配空间
(2) B temp = new B();
通过B类的构造函数建立了一个B类对象的实例，也就是初始化
(3) a = (A)temp;
将子类对象temp转换未父类对象并赋给a，这就是上传(upcast)，是安全的。
经过以上3个过程，a就彻底成为了一个A类的实例。
即只能访问A类的属性，使用B类的方法。
但是对于AB两类中同名的方法，a.f()调用的仍然是B类的方法。
######END######

static修饰属性强调它只有一个，final修饰属性表明是一个常数（创建后不能再修改），
static final修饰属性则指一旦给值就不可修改，并可以通过类名访问。
static final也可以修饰方法，表示该方法不能重写，可以在不new对象的情况下调用。

Properties是继承于Hashtable的，属性集。