1이상 10000이하의 수를 랜덤 100회 생성하여, 1의 자리수값으로 그룹을 지어, 많은 숫자가 포함된 그룹부터 TOP3 출력

In [13]:
```
import random
from collections import defaultdict

number_dict = defaultdict(lambda : 0)

for i in range(1,100):
    number = random.randint(1,10000)
    last_number = number % 10
    number_dict[last_number] += 1

print(sorted(number_dict.items(), key = itemgetter(1), reverse = True)[:3])
```

```
[(4, 14), (2, 13), (6, 13)]
```

Callable Object - Function - Decorator

In [49]:
```
def base(i):
    def outer_wrap(fn):
        def inner_wrap(x,y):
            return fn(x,y) + i
        return inner_wrap
    return outer_wrap


@base(10)
def plus(x,y):
    return x + y

plus(1,2)
```

Out[49]: 13

Callable Object - Class - Decorator

In [49]:
```
class base:
    def __init__(self,i):
        self.i = i
    def __call__(self,fn):
        def wrap(*args):
            return fn(*args) + self.i
        return wrap

@base(10)
def plus(x,y):
    return x + y

plus(1,2)
```

Out[49]: 13

In [30]:
```
class Calculator(object):
    def __init__(self, base):
        self.base = base
    def __call__(self, x, y):
        return self.base + x + y
calculator = Calculator(10)
print(calculator(1, 2))
```

```
13
```

무한 팩토리얼

In [26]:
```
def infinite_fact(n):
    x,y = 1,0
    while n+1 > y:
        yield x
        y,x = y+1 , x*(y+1)


for i in infinite_fact(5):
    print (i)
```

```
1
1
2
6
24
120
```

In [42]:
```
def factorial(n):
    result = 1
    for i in range(n,1,-1):
        result *= i
    return result

factorial(10)
```

Out[42]: 3628800

In [44]:
```
def factorial2(n):
    if n <= 1:
        return 1
    else:
        return factorial(n-1)*n

factorial(10)
```

Out[44]: 3628800

무한 랜덤값

In [36]:
```
import random
def infinite_random(n):
    for i in range(n):
        yield random.randint(1,1000)

for i in infinite_random(5):
    print (i)
```

```
31
769
66
506
491
```

UTF-8 Encoded csv

In [31]:
```
print(open('/Users/sunki/work.csv', 'rt', encoding='UTF-8').read())
```

```
이름,국어성적,영어성적
철수0,1,2
철수1,2,3
철수2,3,4
철수3,4,5
철수4,5,6
철수5,6,7
철수6,7,8
철수7,8,9
철수8,9,10
철수9,10,11
철수10,11,12
```

In [34]:
```
columns = ['이름', '국어성적', '영어성적', '수학성적']
score_list = [
    {'이름': '철수', '국어성적': 89, '영어성적': 10, '수학성적': 78},
    {'이름': '철수1', '국어성적': 89, '영어성적': 10, '수학성적': 78},
    {'이름': '철수2', '국어성적': 89, '영어성적': 10, '수학성적': 78},
]

for column in columns:
    print(column, end=',')
print()

for score_dict in score_list:
    print('{이름},{국어성적},{영어성적},{수학성적}'.format(**score_dict))
```

```
이름,국어성적,영어성적,수학성적,
철수,89,10,78
철수1,89,10,78
철수2,89,10,78
```

In [35]:
```
import csv

with open('result.csv', 'wt', encoding='UTF-8') as f:
    writer = csv.writer(f, delimiter='|')
    # csv.reader(f)
    writer.writerow(['이름', '국어성적', '영어성적', '수학성적'])
    writer.writerow(['철수1', '90', '95', '100'])
    writer.writerow(['철수2', '90', '95', '100'])
    # print(writer.writerows)
```

In [36]:
```
!cat result.csv
```

Regular Expression

In [40]:
```
import re

PATTERN = r'^01[016-9]-?\d{3,4}-?\d{4}$'
def phone_validator(number):
    if re.match(PATTERN,number):
        return True
    return False
```

In [41]:
```
phone_validator('01039598599')
```

Out[41]: True

숫자 1글자 : [0123456789] 또는 [0-9] 또는 \d 알파벳 소문자 1글자 : [abcdefghijklmnopqrstuvwxyz] 혹은 [a-z] 알파벳 대문자 1글자 : [ABCDEFGHIJKLMNOPQRSTUVWXYZ] 혹은 [A-Z] 알파벳 대/소문자 1글자 : [a-zA-Z] 화이트 스페이스 : [\t\n\r\f\v] 혹은 \s 16진수 1글자 : [0-9a-fA-F] 문자열의 시작을 표시 : ^ 문자열의 끝을 표시 : $ 한글 1글자 : "ㄱ-힣"

re.match : 문자열 전체매칭 re.search : 문자열 부분매칭 re.sub : 지정 패턴의 문자열을 다른 문자열로 변경

숫자 0회 또는 1회 : \d? 숫자 0회 이상 : \d* 숫자 1회 이상 : \d+ 숫자 2글자 : \d{2} 숫자 3글자 이상, 5글자 이하 : \d{3,5}

List , Dict , Set

[표현식 for 변수 in 순회가능한객체 if 필터링조건] { Key:표현식 for 변수 in 순회가능한객체 if 필터링조건 } { 표현식 for 변수 in 순회가능한객체 if 필터링조건 }

In [59]:
```
[i**2 for i in range(10) if i % 2 == 0]
```

Out[59]: [0, 4, 16, 36, 64]

In [21]:
```
numbers1 = [1, 3, 5, 7]
numbers2 = [2, 4, 6, 8]
print(numbers1 + numbers2)
print([i + j for (i, j) in zip(numbers1, numbers2)])
```

```
[1, 3, 5, 7, 2, 4, 6, 8]
[3, 7, 11, 15]
```

In [24]:
```
{j: j**2 for j in range(10) if j % 2 == 0}
```

Out[24]:
```
{0: 0, 2: 4, 4: 16, 6: 36, 8: 64}
```

In [26]:
```
dict_values = { 'blue': 10, 'yellow': 3, 'red': 7 }
print('magenta' in dict_values)
for (key, value) in dict_values.items():
    print(key, value)
```

```
False
red 7
yellow 3
blue 10
```

In [58]:
```
{ i%5 for i in range(100) if i % 2 == 0}
```

Out[58]:
```
{0, 1, 2, 3, 4}
```

In [20]:
```
set_numbers1 = { 1, 3, 4, 5, 1, 4, 3, 1 }
set_numbers2 = { 11, 13, 14, 15, 11, 14, 13, 11,1 }
print(set_numbers1)
print(set_numbers1 - set_numbers2) # 차집합
print(set_numbers1 | set_numbers2) # 합집합
print(set_numbers1 & set_numbers2) # 교집합
print(set_numbers1 ^ set_numbers2) # 여집합
print(5 in set_numbers1)
```

```
{1, 3, 4, 5}
{3, 4, 5}
{1, 3, 4, 5, 11, 13, 14, 15}
{1}
{3, 4, 5, 11, 13, 14, 15}
True
```

In [65]:
```
tuple( i%5 for i in range(30) if i % 2 == 0 )
```

Out[65]:
```
(0, 2, 4, 1, 3, 0, 2, 4, 1, 3, 0, 2, 4, 1, 3)
```

Filter, Sort, Map

In [71]:
```
list(filter(lambda x: x<5 , range(10)))
```

Out[71]:
```
[0, 1, 2, 3, 4]
```

In [36]:
```
mylist = list(range(20))
mylist2 = sorted(mylist, key=lambda x: x%3)

print (mylist2)
print (mylist)

mylist.sort(key = lambda x: x%3)

print (mylist)
```

```
[0, 3, 6, 9, 12, 15, 18, 1, 4, 7, 10, 13, 16, 19, 2, 5, 8, 11, 14,
17]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19]
[0, 3, 6, 9, 12, 15, 18, 1, 4, 7, 10, 13, 16, 19, 2, 5, 8, 11, 14,
17]
```

In [72]:
```
list(map(lambda x: x**2, range(10) ))
```

Out[72]:
```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [5]:
```
sum(filter(lambda i: i%2==0, range(100000000))) #list
```

Out[5]:
```
2499999950000000
```

In [6]:
```
sum(i for i in range(100000000) if i % 2 == 0) #generator
```

Out[6]:
```
2499999950000000
```

Pytest

func.py

def func(x): return x + 2

def base_add(x, y): '인자의 합에 10을 더한값을 리턴' return 10 + x + y

test_func.py

from func import func, base_add

def test_func(): assert func(1) == 3

def test_base_add1(): assert base_add(1, 2) == 113

def test_base_add2(): assert base_add(10, 2) == 122

shell -> cd 경로 -> python -m pytest