1차시 영상
slice_slicing 기능
a := make([]string, 4, 5) > len 4 cap 5 메모리 할당 미리 받기
c := append(a, "y") c는 len 5 cap 5 a z c d y가 됨
c := append(a, ""x","y") 하면 len은 6 cap 은10 (임의로 잡힘 여유분까지 보통2배)
a := []string{"a", "b", "c", "d"} len(a) =>4 cap(a) =>4
as := a[0:2] 0부터 시작해서 2 앞에까지 > a부터 c 앞까지 > a, b
slice 한 as의 값을 바꿔도 원래 잘라온 a도 값이 바뀜> 번지의 주소를 받았기 때문(포인터)
=>as와 a의 주소는 같지만 c는 다름(별도의 메모리를 더 할당받아서)
배열도 slicing 가능
b := [4]int{4, 3, 2, 1}
bs := b[1:3] 1부터 3전까지 니까 1, 2번인 >3과 2 출력
(slice와 배열의 차이 []안에 숫자 있냐 없냐)
s2 := append([]int{}, s1...) 반복문은 귀찮아서 s1의 slice를 s2에 복제함

2주차
func numbers(n1 int, n2 int) => 인수를 2개 받는 함수
numbers(5,10,3)하면 인수가 2개가 넘어서 에러가남 => 가변처리
func numbers(n1 int, n2 ...int) 뒤에 인수는 슬라이스로 동작함
numbers의 인수에 슬라이스는 들어갈 수 없음. int 여러개만
p가 슬라이스면 numbers(p) 는 불가능 primes(p...)는 가능 ...이 슬라이스임을 알려줌
for range를 이용하면 편하게 다룰 수 있음

```
p := []int{111, 19, 1, 17, 15}
idx := 2
p = append(p[:idx], p[idx+1:]...)  => 슬라이스p에서 1을 자름 append이용

for k := idx + 1; k < len(p); k++ {
        p[k-1] = p[k]
 }
 p = p[:len(p)-1]
```

=>위에 슬라이스에서 p를 자르는 것과 같음 for문 이용
sort.Ints(p) 는 인트형 슬라이스인 p를 솔트함