

```

#선택정렬1
def selectionsort1(dat):
    size=len(datalist)
    cnt=0
    for i in range(0, size-1): #i=0~size-2
        for j in range(i+1, size): #j=i+1~size-1(1~4)
            if datalist[i]>datalist[j]: #교환 조건
                datalist[i],datalist[j]=datalist[j],datalist[i]
                cnt += 1 #데이터 교환 횟수 증가
        print(i+1, "단계:", datalist)
        print('-'*30)

    print("총 데이터 교환 횟수 :", cnt)

#메인코드
print("[선택정렬1 - 오름차순]")
datalist=[5,4,2,1,3]
print("\n선택정렬1 전:", datalist)
selectionsort1(datalist)
print("\n선택정렬1 후:", datalist)

```

Handwritten notes for selectionsort1:

- 1씩 증가.** (Next increase)
- 50/10** (50/10)
- i=0** (i=0)
- while (i < size) i=0** (while (i < size) i=0)
- i=0** (i=0)
- i=i+1, j=j+1** (i=i+1, j=j+1)

```

#선택정렬2
def selectionsort2(nlist):
    size=len(nlist)
    cnt=0 #데이터 교환 횟수 변수
    for i in range(size-1): #i=0~size-2
        minindex=i
        for j in range(i+1, size): #j=0~size-1
            if nlist[minindex]>nlist[j]: #교환 조건
                minindex=j #단계별 작은 값의 인덱스로 교환
        if minindex!=i:
            nlist[i],nlist[minindex]=nlist[minindex],nlist[i]
            cnt+=1 #데이터 교환 횟수 증가
        print(i+1, "단계:", nlist)
        print("-"*30)

    print("총 데이터 교환 횟수:", cnt)

```

Handwritten notes for selectionsort2:

- data list** (data list)
- size=5** (size=5)
- 0~4, 3** (0~4, 3)
- minindex=1, min=3만큼** (minindex=1, min=3만큼)
- 0 1 2 3 4** (0 1 2 3 4)
- 1 3 2 2 1** (1 3 2 2 1)

```

#메인코드
print("[선택정렬2 - 오름차순]")
datalist=[5,4,2,1,3]
print("선택정렬2 전:", datalist)
selectionsort2(datalist) #선택정렬2 함수 호출
print("선택정렬2 후:", datalist)

```

#삽입정렬

```
def insertion_sort(nlist):  
    size=len(nlist)    size = 5  
    0  
    for i in range(1, size): #데이터 삽입 순서: 2번째데이터~마지막데이터  
        indata=nlist[i] #삽입되는 데이터  
        pos=i #삽입되는 데이터의 위치(index)  
  
        while pos>0 and nlist[pos-1]>indata:  
            nlist[pos]=nlist[pos-1]  
            pos -= 1 #pos=pos-1  
  
        if pos != i: #pos가 i가 아니라면 indata를 위치에 저장  
            nlist[pos]=indata  
  
    print(i, "단계:", nlist)
```

2번씩
이동

```
print(i, "단계:", nlist)  
print("-"*30)
```

#메인코드

```
print("[삽입정렬 - 오름차순]")  
datalist=[5, 2, 1, 3]  
print("\n삽입정렬 전:", datalist)  
insertion_sort(datalist) #삽입정렬 함수 호출  
print("\n삽입정렬 후:", datalist)
```

결과

제출결과

테스트케이스

< 순차탐색 >

```
#함수 정의
def SearchList(k,alist):
    size=len(alist) #리스트의 길이
    for i in range(size): #range(0, size, 1)
        if k==alist[i]:
            return i
    return -1

#메인코드
print('[ 순차 탐색 ]\n')
datalist = [5,4,2,1,3]
print("리스트 데이터 :",datalist)
key=int(input("탐색 키(search key) 입력 : "))
index=SearchList(key,datalist) #순차 탐색 함수 호출

if index==-1:
    print("=> 탐색 실패")
else:
    print("=> 탐색 성공!!!\n=> 탐색 위치(index) =",index)
```

```
#함수 정의
def SearchList(k,alist):
    size=len(alist) #리스트의 길이
    for i in range(size): #range(0, size, 1)
        print("%d단계."%(i+1),end="|")
        if k==alist[i]:
            print("%d와(과) 인덱스 %d번째의 값 %d 비교 => 일치"%(k,i,alist[i]))
            return i
        else:
            print("%d와(과) 인덱스 %d번째의 값 %d 비교 => 불일치"%(k,i,alist[i]))
    return -1
```

```
#메인코드
print('[ 순차 탐색 ]\n')
datalist = [5,4,2,1,3,7,6]
print("리스트 데이터 :",datalist)
key=int(input("\n탐색 키(search key) 입력 : "))

index=SearchList(key,datalist) #순차 탐색 함수 호출
print()
if index==-1:
    print("=> 탐색 실패")
else:
    print("=> %d회에 탐색 성공!!!\n=> 탐색 위치(index) = %d"%(index+1,index))
```

temp

#함수 정의

```
def SearchList(k,alist): #순차 탐색 함수  
    size=len(alist) #리스트의 길이
```

```
    for i in range(size): #range(0,size,1)  
        if k==alist[i]:  
            return i  
    return -1
```

저장 위치

#메인코드

```
print('[과일 탐색 프로그램]')
```

```
fruits = ["사과", "참외", "배", "수박", "토마토", "딸기", "포도", "바나나", "메론", "귤"]
```

```
while True:
```

```
    key=input("\n찾을 과일 입력 : ")
```

```
    if key=="끝":
```

```
        print("프로그램을 종료합니다.")
```

```
        break
```

```
    index=SearchList(key,fruits)
```

```
    print()
```

```
    if index==-1:
```

```
        print(key+"은(는) 과일 리스트에 없습니다.")
```

```
    else:
```

```
        print(key+"의 저장 위치(index)는 "+str(index)+"입니다.")
```



```

1 pstr = "A STRING SEARCHING EXAMPLE CONSISTING OF STING"
2 pattern = "STING"
3
4 print("주어진 문자열 :", pstr)
5 print("찾고자 하는 패턴 :", pattern)
6 print()
7 ssize = len(pstr) # 주어진 문자열의 길이
8 psize = len(pattern) # 패턴 길이
9
10 for i in range(ssize): # 문자열의 길이까지 반복이 될수 있도록
11     if len(pstr[i:]) < psize: # 남은 문자열의 길이가 패턴 길이보다 작으면 프로그램 종료
12         print("Finish!!!")
13         break
14
15 cnt=0 # 문자열과 패턴의 일치하는 개수 카운트 변수
16
17 for j in range(psize): # 패턴 길이까지 반복
18     print("i=", i, "j=", j, "i+j=", i+j, pstr[i+j], pattern[j], "cnt=", cnt) # 문자열 패턴 매칭 과정
19     if pstr[i+j] == pattern[j]:
20         cnt += 1 # 일치되는 문자열 개수 카운트
21     if cnt == psize: # 일치하는 개수와 패턴 길이가 같으면
22         print("Matched index", i) # 매칭

```

Handwritten notes on the first image:

- $i = 38$ (circled)
- $70 \sim 48$ (written next to line 10)
- $ssize = 49$ (written next to line 7)
- $j = 0 \sim 4$ (written next to line 17)
- $i = 38$ (written next to line 10)
- 5 (written next to line 20)
- $5 = 5$ (written next to line 21)
- $pstr[38+0] == pattern[0]$ (written next to line 19)
- 5 (written next to line 20)
- 5 (written next to line 21)
- 5 (written next to line 22)
- 5 (written next to line 23)
- 5 (written next to line 24)
- 5 (written next to line 25)
- 5 (written next to line 26)
- 5 (written next to line 27)
- 5 (written next to line 28)
- 5 (written next to line 29)
- 5 (written next to line 30)
- 5 (written next to line 31)
- 5 (written next to line 32)
- 5 (written next to line 33)
- 5 (written next to line 34)
- 5 (written next to line 35)
- 5 (written next to line 36)
- 5 (written next to line 37)
- 5 (written next to line 38)
- 5 (written next to line 39)
- 5 (written next to line 40)
- 5 (written next to line 41)
- 5 (written next to line 42)
- 5 (written next to line 43)
- 5 (written next to line 44)
- 5 (written next to line 45)
- 5 (written next to line 46)
- 5 (written next to line 47)
- 5 (written next to line 48)
- 5 (written next to line 49)
- 5 (written next to line 50)
- 5 (written next to line 51)
- 5 (written next to line 52)
- 5 (written next to line 53)
- 5 (written next to line 54)
- 5 (written next to line 55)
- 5 (written next to line 56)
- 5 (written next to line 57)
- 5 (written next to line 58)
- 5 (written next to line 59)
- 5 (written next to line 60)
- 5 (written next to line 61)
- 5 (written next to line 62)
- 5 (written next to line 63)
- 5 (written next to line 64)
- 5 (written next to line 65)
- 5 (written next to line 66)
- 5 (written next to line 67)
- 5 (written next to line 68)
- 5 (written next to line 69)
- 5 (written next to line 70)
- 5 (written next to line 71)
- 5 (written next to line 72)
- 5 (written next to line 73)
- 5 (written next to line 74)
- 5 (written next to line 75)
- 5 (written next to line 76)
- 5 (written next to line 77)
- 5 (written next to line 78)
- 5 (written next to line 79)
- 5 (written next to line 80)
- 5 (written next to line 81)
- 5 (written next to line 82)
- 5 (written next to line 83)
- 5 (written next to line 84)
- 5 (written next to line 85)
- 5 (written next to line 86)
- 5 (written next to line 87)
- 5 (written next to line 88)
- 5 (written next to line 89)
- 5 (written next to line 90)
- 5 (written next to line 91)
- 5 (written next to line 92)
- 5 (written next to line 93)
- 5 (written next to line 94)
- 5 (written next to line 95)
- 5 (written next to line 96)
- 5 (written next to line 97)
- 5 (written next to line 98)
- 5 (written next to line 99)
- 5 (written next to line 100)

```

1 pstr = "A STRING SEARCHING EXAMPLE CONSISTING OF STING"
2 pattern = "STING"
3
4 print("주어진 문자열 :", pstr)
5 print("찾고자 하는 패턴 :", pattern)
6 print()
7 ssize = len(pstr) # 주어진 문자열의 길이
8 psize = len(pattern) # 패턴 길이
9 cnt=0 # 여러개의 패턴에 대한 개수 카운트 변수

```

```

1 i=0 # 초기값
2 while i < ssize: # 문자열의 길이까지 반복
3     temp = pstr[i:]
4     if pattern in temp: # 문자열 패턴이 포함된다면
5         tempindex = temp.index(pattern) # 인덱스 반환
6
7         print("Matched index: ", i+tempindex)
8         cnt += 1
9         i = i+tempindex+1
10    else:
11        break
12 if cnt == 0:
13     print("No index Matched")
14 print("Finish!!!")

```

Handwritten notes on the second image:

- $i=0$ (circled)
- $temp = pstr[i:]$ (circled)
- $tempindex = temp.index(pattern)$ (circled)
- $index$ (circled)

```

def BubbleSort(blist):
    size=len(blist) #리스트의 길이
    cnt=0 #데이터 교환 횟수 카운트
    for i in range(0,size-1): #단계(회전), i=0,1,2,3 / 0~size-2
        for j in range(0,size-i-1): #데이터 교환에 대한 변수
            #i=0, j=0~3(=5-0-2) / i=1, j=0~2(=5-1-2) / i=2, j=0~1 / i=3, j=0
            if blist[j]>blist[j+1]: #데이터 교환 조건
                temp=blist[j]
                blist[j]=blist[j+1]
                blist[j+1]=temp
                #blist[j],blist[j+1]=blist[j+1],blist[j]
                cnt+=1 #데이터 교환 횟수 증가
            print(i+1,"단계:",blist)
            print("-"*30)
        print("총 데이터 교환 횟수 :",cnt)

```

```

#메인코드
datalist=[5,4,2,1,3]
print("[버블정렬 - 오름차순]")
print("\n버블정렬 전 :",datalist)
print()
BubbleSort(datalist)
print("\n버블정렬 후 :",datalist)

```

```

1 menu=["탕수육", "유린기", "팔보채", "유산술", "라조기", "고추잡채", "깐풍기", "깐쇼새우"] #8개 메뉴
2
3 for i in range(0,len(menu)):
4     for j in range(i+1,len(menu)):
5         for k in range(j+1,len(menu)):
6             print(menu[i],menu[j],menu[k])

```

```

#이진 탐색 함수 정의
def BinarySearchASC(sk, slist): #오름차순으로 정렬된 데이터를 이진 탐색 수행
    low=0 ; high=len(slist)-1

    while low<=high:
        mid=int((low+high)/2) #중간값(index) 계산
        if sk<slist[mid]:
            high=mid-1
        elif sk>slist[mid]:
            low=mid+1
        else:
            #sk==slist[mid]
            print("=>위치")
            return mid

    return -1 #탐색 실패

```

Handwritten notes and diagram:

- Handwritten "0~" at the top.
- Handwritten "33 key" and "numlist" with arrows pointing to the function parameters.
- A diagram showing a horizontal array of boxes representing an array. Arrows labeled "low", "mid", and "high" point to different positions in the array. A bracket indicates the range from "low" to "high".

```

print('[이진 탐색(오름차순정렬)]\n')
numlist=[43,97,13,96,84,51,64,25,72,14,93,33,6,95,53]
print("원본 데이터:", numlist)
numlist.sort() #오름차순 정렬
print("정렬 데이터:", numlist)
cnt=0 #탐색 횟수 카운트 변수
key=int(input("\n탐색 키(search key) 입력:"))
result = BinarySearchASC(key, numlist) #이진 탐색 함수 호출
print()

if result==-1:
    print("=>탐색 실패")
else:
    print("=>%d번에 탐색 성공!!!"%cnt)
    print("=>탐색 위치(index)=%d"%result)

```

Handwritten notes and annotations:

- Handwritten "33" next to the input key.
- Handwritten "4 33" next to the function call.
- Handwritten "33" next to the result variable.
- Handwritten "4 33" next to the result variable.
- Handwritten "33" next to the result variable.

```

#이진 탐색 함수 정의
def BSearchASC(sk, slist, low, high): #재귀함수
    cnt = 0
    if low > high:
        return -1 #재귀함수 호출 종료
    mid = int((low + high) / 2) #중간인덱스 계산
    cnt += 1

    if sk == slist[mid]:
        print("key=%d, slist[%d]=%d => 일치"%(sk, mid, slist[mid]))
        return mid
    elif sk > slist[mid]:
        low = mid + 1
        print("key=%d, slist[%d]=%d => 불일치"%(sk, mid, slist[mid]))
    else:
        high = mid - 1
        print("key=%d, slist[%d]=%d => 불일치"%(sk, mid, slist[mid]))

    return BSearchASC(sk, slist, low, high)

```

```

print('[이진 탐색(오름차순정렬)]\n')
numlist = [43, 97, 13, 96, 84, 51, 64, 25, 72, 14, 93, 33, 6, 95, 53]
print("원본 데이터:", numlist)
numlist.sort() #오름차순 정렬
print("정렬 데이터:", numlist)
cnt = 0 #탐색 횟수 카운트 변수
start = 0 #시작 인덱스
end = len(numlist) - 1 #마지막 인덱스
key = int(input("\n탐색 키(search key) 입력:"))
result = BSearchASC(key, numlist, start, end) #이진 탐색 함수 호출
print()

```

```

if result == -1:
    print("=>탐색 실패")
else:
    print("=>%d번에 탐색 성공!!!" % cnt)
    print("=>탐색 위치(index)=%d" % result)

```


#함수 작성

def BinarySearchDESC(sk, slist):#오름차순으로 정렬된 데이터를 이진 탐색 수행

global cnt

low=0 ; high=len(slist)-1

while low<=high:

mid=int((low+high)/2) #중간값(index) 계산

cnt+=1 #탐색 횟수 증가

if sk>slist[mid]:

high=mid-1

elif sk<slist[mid]:

low=mid+1

else: #sk==slist[mid]

return mid

return -1 #탐색 실패

print(['과일 찾기 프로그램'])

flist= ["사과", "참외", "배", "수박", "토마토", "딸기", "포도", "바나나", "메론", "귤"]

flist.sort(reverse=True) #내림차순 정렬 - 이진 탐색의 선행작업

print(['과일 찾기 프로그램'])

flist= ["사과", "참외", "배", "수박", "토마토", "딸기", "포도", "바나나", "메론", "귤"]

flist.sort(reverse=True) #내림차순 정렬 - 이진 탐색의 선행작업

while True:

key=input('찾을 과일 입력 : ')

cnt=0 #탐색 횟수 초기화

if key=='끝':

print("프로그램을 종료합니다.")

break

result=BinarySearchDESC(key,flist) #함수 호출

if result==-1:

print(key+"은(는) 과일 리스트에 없습니다.")

else:

print(key+"를 "+str(cnt)+'회 탐색으로 찾았습니다!!!')

print(key+'의 저장 위치(index)는 '+str(result)+'입니다.')