Some Examples

read and write array

arrayread.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/17-1/arrayread.py (3.4.4)

File Edit Format Run Options Window Help

```
a=[]
n=int(input("Enter the length of array: "))
for _ in range(n):
    a.append(int(input("enter the data:")))

print(a)

for i in range(n):
    print(a[i], end=' ')
```

```
Enter the length of array: 5
enter the data:7
enter the data:3
enter the data:8
enter the data:12
enter the data:4
[7, 3, 8, 12, 4]
7 3 8 12 4
```

read and write

```
n=int(input("Enter the length of array: "))
a=[None for x in range(n)]
for i in range(n):
    a[i]=(int(input("enter the data:")))

print(a)

for i in range(n):
    print(a[i], end=' ')
```

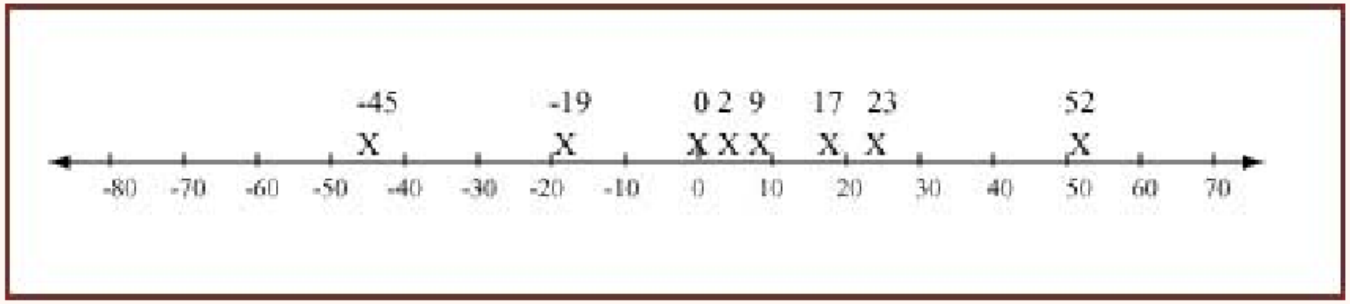
```
Enter the length of array: 5
enter the data:7
enter the data:3
enter the data:8
enter the data:12
enter the data:4
[7, 3, 8, 12, 4]
7 3 8 12 4
```

Minimum and Maximum

The largest integer in a list of integers is the *maximum* value. The maximum may occur several times, but no other integer in the list is larger. Look at the following:

2, 9, 52, 23, -19, 23, 17, -45, 0

The integer 52 is the maximum. If you plot these integers on a number line 52 is the last one on the right:



Similarly, the smallest integer in the list is the *minimum*. The minimum may occur several times. In the above list, the minimum is -45. The minimum is the last number on the left of the number line.

Maximum

max.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/17-1/max.py (3.4.4)

File Edit Format Run Options Window Help

```
def max(ar):
    maxv=ar[0]
    for i in range(len(ar)):
        if ar[i]>maxv :
            maxv=ar[i]
    return maxv

a=[]
n=int(input("Enter the length of array: "))
for _ in range(n):
    a.append(int(input("enter the data:")))

maxvalue=max(a)
print(maxvalue)
```

Maximum

```
Enter the length of array: 9
enter the data:4
enter the data:12
enter the data:3
enter the data:4
enter the data:7
enter the data:1
enter the data:8
enter the data:9
enter the data:3
12
```

Maximum

max.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/17-1/max.py (3.4.4)

File Edit Format Run Options Window Help

```
def max(ar):
    maxv=ar[0]
     for elm in ar[1:] :
          if elm>maxv :
              maxv=elm
     return maxv

a=[]
n=int(input("Enter the length of array: "))
for _ in range(n):
     a.append(int(input("enter the data:")))

maxvalue=max(a)
print(maxvalue)
```

breaking the program

Debugging a program is often difficult. Finding test cases that thouroughly test a program is an important part of this.

Compile and run the program. Once you have it running see if you can "break" it by initializing the array to different values:

- Put the largest element at the beginning.
- Put the largest element at the end.
- Put in more than one copy of the largest element.
- Put in an extremely large element.
- Make all elements the same.
- Make all elements negative.
- Make all elements zero.

Is the correct maximum found in each case? Sometimes a program works for the data a programmer was thinking about when the program was written, but not for all the kinds of data the program is used with.

Minimum

max.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/17-1/max.py (3.4.4)

File Edit Format Run Options Window Help

```
def minimum(ar):
    min=ar[0]
    for elm in ar[1:] :
        if elm<min :
            min=elm
    return min


a=[]
n=int(input("Enter the length of array: "))
for _ in range(n):
    a.append(int(input("enter the data:")))

minvalue=minimum(a)
print(minvalue)
```

Summing

sum.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/17-1/sum.py (3.4.4)

File Edit Format Run Options Window Help

```
def sumution(ar):
    total=ar[0]
    for elm in ar[1:] :
        total = total +elm

    return total


a=[]
n=int(input("Enter the length of array: "))
for _ in range(n):
    a.append(float(input("enter the data:")))

sum=sumution(a)
print("The total is: ", sum)
```

Summing

```
Enter the length of array: 7
enter the data:-47.39
enter the data:24.96
enter the data:-1.02
enter the data:3.45
enter the data:14.21
enter the data:32.6
enter the data:19.42
The total is:  46.230000000000004
```

Summing

sum.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/17-1/sum.py (3.4.4)

File Edit Format Run Options Window Help

```
# a_0 + S a_i / i

def sumution(a):
    s=a[0]
    for i in range(1, len(a)):
        s = s +a[i] / i

    return s


a=[]
n=int(input("Enter the length of array: "))
for _ in range(n):
    a.append(float(input("enter the data:")))

sum=sumution(a)
print("The total is: ", sum)
```

Summing

sum.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/17-1/sum.py (3.4.4)

File Edit Format Run Options Window Help

```
# S a_i * b_i

def sig(a,b):
    s=0
    for i in range(len(a)):
        s = s +a[i] * b[i]

    return s


a=[]
b=[]
n=int(input("Enter the length of array: "))
for _ in range(n):
    a.append(float(input("enter the data for first array:")))
for _ in range(n):
    b.append(float(input("enter the data for second array:")))

sum=sig(a,b)
print("The total is: ", sum)
```

All sum

allsum.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/17-1/allsum.py (3.4.4)

File Edit Format Run Options Window Help

```
# a_i = a_0 + a_1 + a_2 + ... + a_i

def alls(a):
    b=[x for x in a]
    for i in range(1, len(a)):
        for j in range(i-1, -1, -1):
            a[i] = a[i] + b[j]
    return


a=[]
n=int(input("Enter the length of array: "))
for _ in range(n):
    a.append(float(input("Enter the data: ")))

alls(a)
print("The result is: ", a)
```

All sum

allsum.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/17-1/allsum.py (3.4.4)

File Edit Format Run Options Window Help

```
# a_i = a_0 + a_1 + a_2 + ... + a_i

def alls(a):
    for i in range(1, len(a)):
        a[i] = a[i] + a[i-1]

    return


a=[]
n=int(input("Enter the length of array: "))
for _ in range(n):
    a.append(float(input("Enter the data: ")))

alls(a)
print("The result is: ", a)
```

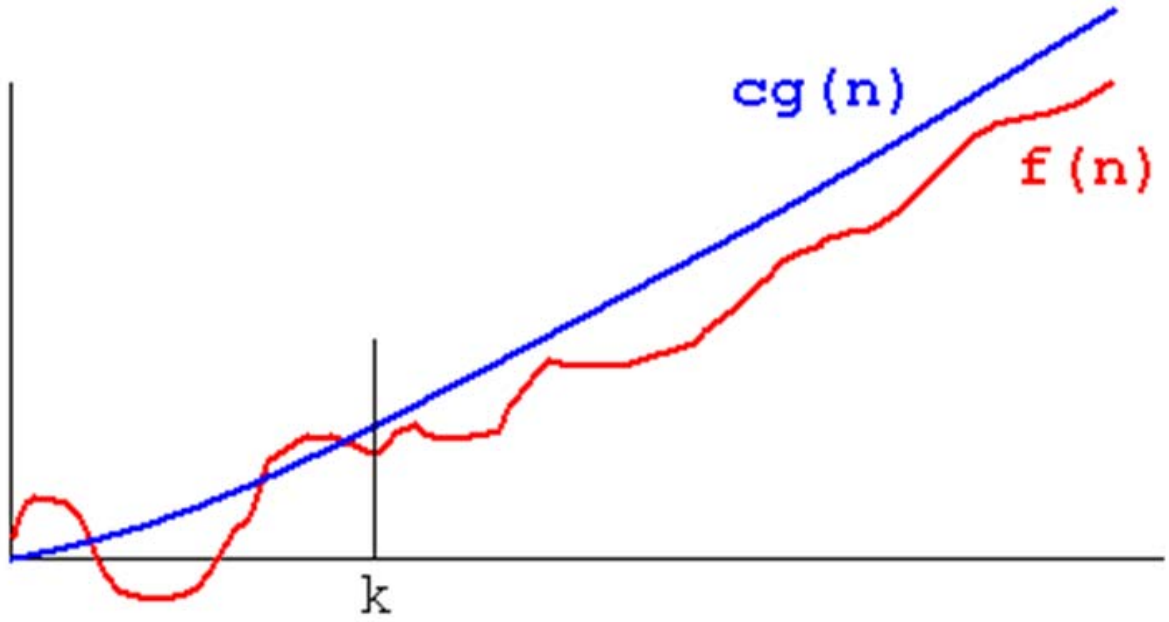
big-O notation

Definition: A theoretical measure of the execution of an *algorithm*, usually the time or memory needed, given the problem size n, which is usually the number of items. Informally, saying some equation $f(n) = O(g(n))$ means it is less than some constant multiple of $g(n)$. The notation is read, "f of n is big oh of g of n".

Formal Definition: $f(n) = O(g(n))$ means there are positive constants c and k, such that $0 \leq f(n) \leq cg(n)$ for all $n \geq k$. The values of c and k must be fixed for the function f and must not depend on n.

big-O notation

Formal Definition: $f(n) = O(g(n))$ means there are positive constants c and k, such that $0 \le f(n) \le cg(n)$ for all $n \ge k$. The values of c and k must be fixed for the function f and must not depend on n.



$f(n)=3n + 4$, $g(n)=n$

$f(n)=O(n)$, $3n+4=O(n)$; $n>3$, $3n+4 <= 4n$

n	3n+4	4n
1	7	4
2	10	8
3	13	12
4	16	16
5	19	20

$n^2 + 3n + 4$ is $O(n^2)$, since $n^2 + 3n + 4 <= 2n^2$ for all $n > 3$

n	*$n^2 + 3n + 4$*	*$2n^2$*
1	8	2
2	14	8
3	22	18
4	32	32
5	44	50

n	*$n^2 + 3n + 4$*	*$3n^2$*
1	8	2
2	14	12
3	22	27
4	32	48
5	44	75

$n^2 + 3n + 4$ is $O(n^2)$, since $n^2 + 3n + 4 <= 3n^2$ for all $n > 2$

Here is a list of classes of functions that are commonly encountered when analyzing algorithms. The slower growing functions are listed first. c is some arbitrary constant.

notation	**name**
$O(1)$	constant
$O(\log(n))$	logarithmic
$O((\log(n))^c)$	polylogarithmic
$O(n)$	linear
$O(n^2)$	quadratic
$O(n^c)$	polynomial
$O(c^n)$	exponential

Order

sum.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/17-1/sum.py (3.4.4)

File Edit Format Run Options Window Help

```
def sumution(ar):
    total=ar[0]
    for elm in ar[1:] :
        total = total +elm
    print(total)
    return total


a=[]
n=int(input("Enter the length of array: "))
for _ in range(n):
    a.append(float(input("enter the data:")))

sum=sumution(a)
print("The total is: ", sum)
```

All sum order

allsum.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/17-1/allsum.py (3.4.4)

File Edit Format Run Options Window Help

```
# a_i = a_0 + a_1 + a_2 + ... + a_i

def alls(a):
    b=[x for x in a]
    for i in range(1, len(a)):
        for j in range(i-1, -1, -1):
            a[i] = a[i] + b[j]
    return


a=[]
n=int(input("Enter the length of array: "))
for _ in range(n):
    a.append(float(input("Enter the data: ")))

alls(a)
print("The result is: ", a)
```

All sum order

allsum.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/17-1/allsum.py (3.4.4)

File Edit Format Run Options Window Help

```
# a_i = a_1 + a_2 + ... + a_i

def alls(a):
    for i in range(1, len(a)):
        a[i] = a[i] + a[i-1]

    return


a=[]
n=int(input("Enter the length of array: "))
for _ in range(n):
    a.append(float(input("Enter the data: ")))

alls(a)
print("The result is: ", a)
```