1. R 이 실행되는 순서 (알아두면 요긴하게 사용)

사용자가 R을 실행할 경우 아래의 순서로 실행.
각종 환경설정 변경 등을 할 경우에 사용.

1) R을 실행하면 R_HOME/etc/Rprofile.site 파일을 실행.

이 파일은 기본적으로 없고, 관리자가 원할 경우 별도로 생성.
일반적으로 거의 사용하지 않음.

2) R의 작업 디렉토리 안에 있는 .Rprofile 스크립트를 실행.

이 파일에는 사용자가 별도로 원하는 환경을 직접 설정할 수 있는 환경설정 내용을 입력.
예) 작업 디렉터리 설정. 자주 사용하는 라이브러리 자동 로딩.

3) 만약 작업 디렉터리에 .RData 로 저장된 파일이 있다면 해당 파일을 읽어서 작업 공간을 불러온다.

4) 만약 .First 함수가 있다면 실행.

이 함수는 사용자가 프로젝트를 시작할 때, 자동으로 실행되는 초기 설정 코드를 입력.
이 파일보다는 .Rprofile 파일을 더 자주 사용.

5) .First.sys 함수를 실행해서 기본적으로 로딩해야 할 패키지들을 로딩.

이 과정이 끝나면, R 사용 준비 완료.

2. 작업용 기본 디렉터리 - setwd("디렉터리명")

작업 디렉터리는 R로 작업을 할 때, 필요한 데이터들을 미리 가져다 두는 약속된 디렉터리.
작업 후, 나오는 출력물들도 기본적으로 이곳에 저장된다.
특별한 조건은 없고, 디렉터리(폴더)를 생성한 후, setwd( ) 함수로 지정.

```
# 아래와 같이 지정.
setwd("c:\\r_temp")
```

```
> setwd("c:\\r_temp")
```

```
# 현재 설정된 작업 디렉터리 확인.
getwd( )
```

```
> getwd( )
[1] "c:/r_temp"
```

```
# 또는 아래와 같이 지정.
setwd("c:/r_temp")
getwd( )
```

```
> setwd("c:/r_temp")
> getwd( )
[1] "c:/r_temp"
```

작업하기 전에는 반드시 작업할 디렉터리를 지정하고, 작업 시(분석할) 필요한 데이터들을 저장.

3. 화면에 결과를 출력 - print( ) 와 cat( ) 사용

R Console 화면 : 명령어를 입력하는 화면. 다른 이름으로 터미널이라고도 호칭.

print( ) 함수 : R Console 화면에 내용을 출력하는 함수.
print( ) 함수의 괄호 안에 출력하고 싶은 함수나 내용들을 적으면,
R Console 화면에 내용이 출력된다.

```
# print 안에 출력하고 싶은 내용을 입력.
print(1+2)
```
```
> print(1+2)
[1] 3
```

```
# 아래 코드는 print 명령이 생략된 것
1+2
```
```
> 1+2
[1] 3
```

```
# 문자를 출력할 때는 ' ' 을 사용
print('a')
```
```
> print('a')
[1] "a"
```

```
# 아래와 같이 문자만 입력해도 정상적으로 출력
'a'
```
```
> 'a'
[1] "a"
```

```
# 소수점일 경우 총 7 자리로 출력
print(pi)
```
```
> print(pi)
[1] 3.141593
```

```
# digits 로 자리수 지정 가능
print(pi,digits=3)
```
```
> print(pi,digits=3)
[1] 3.14
```

함수나 연산을 수행하면, 자동으로 print( ) 함수가 실행되어 결과를 R Console 화면에 출력.
단, print( ) 함수는 한번에 한가지 데이터만 출력할 수 있다.

```
# 아래와 같이 두가지 데이터를 출력하면 첫번째 데이터만 출력.
print(3,4)
```
```
> print(3,4)
[1] 3
```

```
# 문자의 경우는 에러 발생.
print('a','b')
```
```
> print('a','b')
Error in print.default("a", "b") : invalid 'digits' argument
In addition: Warning message:
In print.default("a", "b") : NAs introduced by coercion
```

```
다음에 오류가 있습니다print.default("a", "b") : 'digits' 인자가 잘못되었습니다
추가정보: 경고메시지:
In print.default("a", "b") : 강제형변환에 의해 생성된 NA 입니다
```

cat( ) 함수 : 여러 개의 데이터를 출력하는 함수.

단, 복잡한 데이터인 행렬, 리스트 같은 형태의 데이터는 출력할 수 없다.

```
# 숫자와 문자 여러 개를 한꺼번에 출력.
cat(1,':','a','\n',2,':','b')
```

```
> cat(1,':','a','\n',2,':','b')
1 : a
 2 : b
```

여러 개의 출력 명령을 연속적으로 실행할 경우, 세미콜론(;)을 사용.

```
# 각 명령을 세미콜론으로 구분.
1;2;3
```

```
> 1;2;3
[1] 1
[1] 2
[1] 3
```

```
# 세미콜론을 기준으로 순서대로 실행.
1+2 ; 2*3 ; 4/2
```

```
> 1+2 ; 2*3 ; 4/2
[1] 3
[1] 6
[1] 2
```

4. R 에게 줄 수 있는 자료들

R은 컴퓨터에서 어떤 특정한 작업(주로 분석작업)을 하는 프로그램.
어떤 작업을 하기 원할 경우, R에게 자료를 주고 "어떤 일을 해라" 라고 명령.

R에게 자료를 줄 때,
그 자료가 숫자인지 문자인지 날짜인지, 아니면 다른 특별한 형태인지를 알려주어야 한다.

R에서 사용 가능한 자료(데이터)의 종류와 그 자료 형(데이터 형)은 정해져 있다.

1. 숫자형과 주요 산술연산자

숫자형 : 일상적으로 사용하는 숫자들.
산술연산자 : 숫자를 사용하여 연산할 때 사용하는 연산자.

주요 연산자

기호	의미	사용 예 -> 결과
+	더하기	5+6 -> 11
-	빼기	5-4 -> 1
*	곱하기	5*6 -> 30
/	나누기(실수 가능)	4/2 -> 2
%/%	정수 나누기	위와 동일
%%	나머지 구하기	5%%4 -> 1
^ , **	승수 구하기	3^2 -> 9 , 3^3 -> 27

```
# 연산자의 우선순위에 따라 뒤의 * (곱하기)부터 연산
1+2*3
```
```
> 1+2*3
[1] 7
```

```
# 만약 더하기부터 하려면 왼쪽처럼 괄호를 사용.
(1+2)*3
```
```
> (1+2)*3
[1] 9
```

숫자형의 경우, 길이가 길 경우, 아래와 같은 형식으로 사용 가능.

```
# 0 이 4개 까지는 그대로 출력.
10000
```
```
> 10000
[1] 10000
```

```
# 0 이 5개부터는 1e+05로 출력.
100000
```
```
> 100000
[1] 1e+05
```

```
# 0이 6 개라서 결과에 1e+06 으로 출력.
1000000
```
```
> 1000000
[1] 1e+06
```

```
#  1 * 10^2 이라는 뜻이므로 100 이 출력.
1e2
```
```
> 1e2
[1] 100
```

```
# 3 * 10^2 아는 뜻이라서 3 *100 의 결과인 300 이 출력.
3e2
```
```
> 3e2
[1] 300
```

```
# -1 은 소숫점 1 자리로 출력하라는 뜻.
3e-1
```
```
> 3e-1
[1] 0.3
```

```
# -2 는 소수점 2 자리로 출력하라는 뜻.
3e-2
```
```
> 3e-2
[1] 0.03
```

as.numeric( ) : 문자형 숫자를 연산 가능한 숫자로 변경하는 함수.

```
# 숫자처럼 보여도 사실은 문자.
'1' + '2'
```
```
> '1' + '2'
Error in "1" + "2" : non-numeric argument to binary operator
```

```
다음에 오류가 있습니다"1" + "2" : 이항연산자에 수치가 아닌 인수입니다
```

```
# 숫자로 강제로 변환후, 연산.
as.numeric('1') + as.numeric('2')
```
```
> as.numeric('1') + as.numeric('2')
[1] 3
```

2. 문자형

R에서는 단일 문자와 여러 문자로 구성된 문자열을 구분하지 않고,
전부 문자열 형태로 인식. (" " 또는 ' ' 을 사용)

```
# 문자라서 홑따옴표로 감싸고 출력
'First'
```
```
> 'First'
[1] "First"
```

```
# 쌍따옴표로 감싸도 된다.
"Second"
```
```
> "Second"
[1] "Second"
```

```
# 그냥 사용하면 변수 이름으로 인식이 되어서 에러가 발생.
First
```
```
> First
Error: object 'First' not found
```

```
에러: 객체 'First'를 찾을 수 없습니다
```

class( ) : 데이터 형을 반환 시켜주는 함수.

```
# 문자형 데이터를 검사.
class('1')
```
```
> class('1')
[1] "character"
```

```
# 숫자형 데이터를 검사.
class(1)
```
```
> class(1)
[1] "numeric"
```

3. TRUE / FALSE 값 (논리값)

데이터들을 비교해서 참일 경우, TRUE / 거짓일 경우, FALSE를 반환.

컴퓨터 : bit를 사용하여 동작되는 기기.

bit : 0과 1, 두 가지 경우의 수만 존재.

bit 값 중에서 0은 거짓, 0 이외의 값은 모두 참.

두 개 이상의 값을 연산했을 경우, 참과 거짓을 찾지 위해 여러 가지 연산자가 존재.

& : 양쪽의 데이터가 모두 참일 경우에만 결과가 참.

& 기호를 곱하기로 간주. (1 x 1 = 1, 1 x 0 = 0)

| : 두 값 중에서 한가지만 참이어도 결과는 참.

| 기호는 더하기로 간주. (1 + 0 = 1, 1 + 1 = 1)

! : 해당 데이터가 아닌 것이라는 의미를 보유. NOT 라고 호칭.

```
# 3 곱하기 0 의 뜻으로 거짓(FALSE).
3 & 0
```
```
> 3 & 0
[1] FALSE
```

```
# 3 곱하기 1 의 뜻으로 참 곱하기 참이라서 참(TRUE).
3 & 1
```
```
> 3 & 1
[1] TRUE
```

```
# 참 곱하기 참이라서 결과도 참(TRUE).
3 & 2
```
```
> 3 & 2
[1] TRUE
```

```
# 참 더하기 거짓 이라서 결과는 참(TRUE).
3 | 0
```
```
> 3 | 0
[1] TRUE
```

```
# 참 더하기 참 이라서 결과는 참(TRUE).
3 | 1
```

```
> 3 | 1
[1] TRUE
```

```
# 거짓이 아닌 것이라서 참(TRUE).
!0
```

```
> !0
[1] TRUE
```

```
# 참이 아닌 것이라서 거짓(FALSE).
!1
```

```
> !1
[1] FALSE
```

```
# 참이 아닌 것이라서 거짓(FALSE).
!3
```

```
> !3
[1] FALSE
```

4. NA 형 & NULL 형

NA : 잘못된 값이 들어 올 경우 (Not Applicable , Not Available, 결측값).
값이 있어도 정해진 범위 안에 있는 값이 아니라서 사용할 수 없는 경우.
NA 값이 포함되어 있을 경우 연상르 수행하면 결과 값은 NA.

is.na(변수명) : 변수의 값이 NA 인지를 확인하는 함수.

NULL : 값이 없을 경우.
값이 정해지지 않아서 얼마인지 모른다는 의미.

예) 사람의 나이를 이야기할 때, 0~150 사이가 가능한 나이라고 가정.
변수에 할당된 값이 1000 일 경우, NA
변수에 할당된 값이 없을 경우, NULL

```
# NA 가 그대로 출력.
cat(1,NA,2)
```

```
> cat(1,NA,2)
1 NA 2
```

```
# NULL 값이 제거되고 출력.
cat(1,NULL,2)
```

```
> cat(1,NULL,2)
1 2
```

```
# NA 를 더하니까 결과가 NA 로 출력.
sum(1,NA,2)
```

```
> sum(1,NA,2)
[1] NA
```

```
# NULL 값은 제외해버리고, 나머지 값만 더해서 결과가 출력.
sum(1,NULL,2)
```

```
> sum(1,NULL,2)
[1] 3
```

R에서

NULL 값은 자동으로 제거하고 연산하지만,

NA값은 연산에 그대로 반영되기 때문에 값은 수동으로 제거해야한다.

na.rm : NA 값을 수동으로 제거할 수 있는 파라미터. **na.rm=T** (NA 값을 Remove)

```
# NA 값이 연산 결과를 틀리게 만들었다.
sum(1,2,NA)
```

```
> sum(1,2,NA)
[1] NA
```

```
#na.rm=T 로 NA 값을 제거하고 계산.
sum(1,2,NA,na.rm=T)
```

```
> sum(1,2,NA,na.rm=T)
[1] 3
```

5. Factor 형

여러 번 중복으로 나오는 데이터들을 각각 모아서 대표 값으로 출력해주는 데이터 형

c( ) : combine value 라는 의미로 여러 개의 값을 한꺼번에 처리할 때 사용되는 함수.

```
# Factor 형
setwd("c:\\r_temp")
txt1 <- read.csv("factor_test.txt")
txt1
```

```
> setwd("c:\\r_temp")
> txt1 <- read.csv("factor_test.txt")
> txt1
   no       name blood sex location
1   1     서진수     O  남       서울
2   2     홍길동     A  남       부산
3   3     유관순     O  여       부산
4   4     전우치     B  남       전남
5   5     강감찬    AB  남       강원
6   6   신사임당     A  여       강원
7   7   퇴계이황     B  남       충청
8   8   율곡이이     O  남       전북
9   9 근초고대왕     B  남       제주
10 10   뺑덕어멈     B  여       서울
```

```
factor1 <- factor(txt1$blood)
factor1
```

```
> factor1 <- factor(txt1$blood)
> factor1
 [1] O  A  O  B  AB A  B  O  B  B
Levels: A AB B O
```

```
summary(factor1)
```

```
> summary(factor1)
 A AB  B  O
 2  1  4  3
```

```
gender_F <- factor(txt1$gender)
gender_F
```

```
> gender_F <- factor(txt1$gender)
> gender_F
 [1] 남 남 여 남 남 여 남 남 남 여
Levels: 남 여
```

```
summary(gender_F)
```

```
> summary(gender_F)
남 여
 7  3
```

stringsAsFactors=FALSE : 데이터를 Factor 형태로 바꾸지 않고, 그대로 사용하고 싶을 경우.

6. 날짜와 시간

1) 기존 방법으로 날짜와 시간 제어.

```
# 대소문자 주의! 날짜만 보여주는 함수.
Sys.Date()
```

```
> Sys.Date()
[1] "2017-08-25"
```

```
# 대소문자 주의! 날짜와 시간까지 보여주는 함수.
Sys.time()
```

```
> Sys.time()
[1] "2017-08-25 17:35:06 KST"
```

```
# 미국식 날짜와 시간을 보여주는 함수
date( )
```

```
> date( )
[1] "Fri Aug 25 17:35:07 2017"
```

```
# 홑따옴표나 겹따옴표를 사용.
as.Date('2014-11-01')
as.Date("2014/11/01")
```

```
> as.Date('2014-11-01')
[1] "2014-11-01"
> as.Date("2014/11/01")
[1] "2014-11-01"
```

```
# 의도하지 않은 날짜.
as.Date("01-11-2014")
```

```
> as.Date("01-11-2014")
[1] "0001-11-20"
```

```
# 날짜 형태를 지정.
as.Date("01-11-2014",format="%d-%m-%Y")
```

```
> as.Date("01-11-2014",format="%d-%m-%Y")
[1] "2014-11-01"
```

형 식	의 미
%d	일자를 숫자로 인식합니다.
%m	월 을 숫자로 인식합니다.
%b	월 을 영어 약어로 인식합니다.
%B	월 을 전체 이름으로 인식합니다.
%y	년도를 숫자 두 자리로 인식합니다.
%Y	년도를 숫자 네 자리로 인식합니다.

```
# 날짜 형태를 지정.
as.Date("01-11-2014",format="%d-%m-%Y")
```
```
> as.Date("01-11-2014",format="%d-%m-%Y")
[1] "2014-11-01"
```

```
as.Date("2014년 11월 1일",format="%Y년 %m월  %d일")
```
```
> as.Date("2014년 11월 1일",format="%Y년 %m월  %d일")
[1] "2014-11-01"
```

```
# 년도가 대문자 Y 사용.
as.Date("01112014",format="%d%m%Y")
```
```
> as.Date("01112014",format="%d%m%Y")
[1] "2014-11-01"
```

```
# 년도가 소문자 y 사용
as.Date("011114",format="%d%m%y")
```
```
> as.Date("011114",format="%d%m%y")
[1] "2014-11-01"
```

```
#  기준 일자를 주고  몇 일 후 찾기
# 주어진 날짜기준으로 10일 후의 날짜
as.Date(10,origin="2014-11-10")
```
```
> as.Date(10,origin="2014-11-10")
[1] "2014-11-20"
```

```
# 주어진 날짜 기준으로 10일 이전 날짜
as.Date(-10,origin="2014-11-10")
```
```
> as.Date(-10,origin="2014-11-10")
[1] "2014-10-31"
```

```
# 날짜 연산 하기
# 날짜처럼 보이지만 문자입니다.
"2014-11-30" - "2014-11-01"
```
```
> "2014-11-30" - "2014-11-01"
Error in "2014-11-30" - "2014-11-01" :
  non-numeric argument to binary operator
```

```
# 문자로 확인.
class("2014-11-30")
```
```
> class("2014-11-30")
[1] "character"
```

```r
# 날짜로 변경해서 계산.
as.Date("2014-11-30") - as.Date("2014-11-01")
```

```
> as.Date("2014-11-30") - as.Date("2014-11-01")
Time difference of 29 days
```

```r
as.Date("2014-11-01") + 5
```

```
> as.Date("2014-11-01") + 5
[1] "2014-11-06"
```