2020-05-12 오전

컴퓨터 : **자동**으로 계산(데이터 처리)을 수행하는 기기

Input Output

Data ------------🡪 Process ------------🡪 Information

(값) (자동) Data Model

↓

분석

Data와 Information 구분하기

Computer System

* Hardware : CPU, Memory, I/O device
* Software : System S/W : Computer System 관리 ex) O/S (Operating System) 덜 친화적

↕구분되는 기준 : 사용자(End-User)에게 친화적

Application S/W

* 사용자

스크린샷이(가) 표시된 사진

자동 생성된 설명CPU의 기능

1. 연산기능 (산술/논리/관계) – ALU
2. 제어기능 (명령 해석) – CU

CPU 동작

1. ALU 이용 연산
2. I/O 동작 – I/O Device

Register – 임시 기억

Memory (기억 기능) – 프로그램 내장 방식 겸 컴퓨터 → 2진수로만 표현 가능 (0,1)

기억할 수 있는 것

1. Program Code
2. Data

텍스트이(가) 표시된 사진

자동 생성된 설명모든 내용은 Memory에 있어야함

2진수 한자리 : 1 bit

8 bit = 1 byte = 최소 저장 단위

=256가지 표현 가능

영문자 하나 = 1 byte

이제 글자 저장할 때 = 2 byte (=한글)

인코딩 : 사람 언어를 컴퓨터 관점으로 바꾸는 것 Encoding

디코딩 : 컴퓨터 언어를 사람 관점으로 바꾸는 것 Decoding

ASCII Code : 인코딩, 디코딩 할 때 기준이 되는 표 = 1 byte 초창기 영어권에서 사용할 때

Uni Code = 2 byte 전세계 모든 문자를 사용하기 위해 만들어짐

UTF-8 방식 인코딩

인코딩이 인코딩, 디코딩 모두 통칭함

Memory Address = Reference = 위치값

텍스트이(가) 표시된 사진

자동 생성된 설명

위치값 쓰는 용도를 잘 알아야 함

2020-05-12 오후

시계이(가) 표시된 사진

자동 생성된 설명Memory 접근 순서

1. 접근할 위치의 주소 지정 - Address bus
2. [읽기/쓰기] 동작 – Control bus

get set

“”이 붙으면 문자열 ex) “ABC” = 3 byte

숫자 256 이상 표현할 때 = overflow ↔ underflow

Data bus는 대부분 정수의 크기에 맞춰서 만들어짐

Data bus size = 1 word

숫자 : 정수 : 4 byte

한문자 : 1 byte

실수 : 8 byte

저장과 보관의 차이

* 저장 : 언제든지 바로 쓸 수 있다 → Memory에서 많이 쓰임
* 보관 : 당장 바로 쓸 수 없다 → 보조 기억 장치, 보조 Memory에서 많이 쓰임

I/O Device : Memory로 입출력

보조기억장치, 2차 memory 입출력

1. Software

* System Software : Computer 시스템을 관리하는 S/W

O/S (Operation System) : Computer H/W를 제어하는 S/W

System (Language) Library : 다른 프로그램이 사용하는 기능을 제공하는 S/W 묶음

Application

* Application Software : 사용자 관점의 사용성을 갖는 S/W

Application

Language Library

UI (User Interface)

* CUI (Character UI)

Platform

(Stack 구조)

CPU

Memory

System Library

OS – UI/ Kernel

* GUI (Graphic UI)

**통신**

I/O

Device

DLL (Dynamic Link Library)

Library = 묶음

2. Software 개발을 위해 필요한 프로그램

* 1. Editor program

: Source code (Script) 작성 (Text 편집기)

* Text 파일 : ASCII 형식으로 구성된 파일
* Binary 파일 : Binary 형식으로 구성된 파일
  1. Compiler (Compile program) (Interpreter [Interpret program])

: Source code를 Binary 형태로 번역하는 프로그램

* Compiler : 일괄 번역하는 방식 – C, C++, Java, C#
* Interpreter : Code 한 줄 씩 번역하는 방식 – R, Python
* Syntax Error : 번역시 오류 발생, 문법에 맞지 않는다.
  1. Linker (Link program)

: Binary 형태로 바뀐 파일을 실행 가능한 파일로 만드는 프로그램

: 이때 Language Library를 link한다. Link = 붙인다

* Link Error : Language Library에 없는 기능 사용시 발생
  1. Program을 실행하는 프로그램 (Loader)
* \*\*\*\*\*Run-time Error : 프로그램 결과가 원하는 결과가 아닌 모든 경우

\* IDE (Interpreted Development Environment, 통합 개발 환경)

: Source 편집, 번역, link, 실행, debuggin을 하나의 프로그램으로 수행

R 프로그램 : Interpreter 포함

R Studio : R용 IDE

Git : 버전 (형상) 관리 S/W → Local 사용

ㄴ 문서에 대한 이력 관리

1. 중앙 집중 방식 : SVN
2. 분산 방식 : Git

Git-hub → Global 사용

Git 사용 방법

3.1 Git 초기화 (init) – git init

3.2 Git으로 관리할 대상 추가 (add) – git add <파일명>

3.3 관리 대상 확정 (commit) – git commit -m “설명”

- Local에서 사용

3.5 Github (remote)와 연결

3.6 Github (remote)에 올리기 (push)

- Remote와 연계

3.8 Github (remote)에서 현재 pc (Local)로 가져오기 (pull)

R Console : Prompt – 명령 대기 상태

2020-05-13 오전

CLI = Command Line Interface

C: = C drive : 부팅 디스크, 계층 구조

(\) : root (Directory)

* Program Files
* Windows
* 사용자

- hong

Git 명령어

**cls = clear the screen 다 지울 때**

**dir = file 목록, 자세히**

**dir /w = file 목록, 이름만 보여줌, 간략하게 대괄호 → 디렉토리 확인할 때**

**cd = change directory, 디렉토리 변경할 때, cd 띄고+변경 디렉토리 이름+엔터**

세 글자 쓰고 tab키 = 이름 자동 완성

**cd \ = 제일 처음**

\ = 역 슬래시

처음 \는 root 다음 \은 구분자

처음에는 \를 쓰지만 그 뒤에 사용할 땐 넣지 않음

Cd \workspace → cd workspaceR (O)

Cd \workspace\workspaceR (O)

드라이브 변경할 때 = 드라이브 이름 + : ex) C: / D:

. 으로 시작하는 file = hidden file

repository = 저장소

**type = 텍스트 파일 내용 보기, 수정 X**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Data Base

↑Field (항목)

←Record (Data)

R 프로그래밍 방식

1. CLI 방식 : 명령을 입력해서 즉각적인 반응을 보고 싶을 때 console 창에 입력
2. Script 방식

Ls = list 명령어

>= prompt = 준비되었다

( ) 괄호가 있으면 함수 (function)

ex) class( 3 ) / print( 3 + 5 ) ⇒ 3, 3+5 = 인수 (Argument), Parameter

함수 : 단위 기능을 수행하는 code 집합

ㄴ 하나의 동작

함수 class : 어떤 값의 타입을 알아내는 함수 ex) class ( 3 ) = numeric ⇒ 숫자

class( “hello” ) ⇒ character ⇒ 문자

2020-05-13 오후

ex) print ( 10 + 5 )

함수 : print ( 10 + 5 )

값 : 10, 5

인수 : 10 + 5

연산자 : +

우선 순위 : Computer에서 실행 순서는 순차적 - 위 → 아래 / 왼쪽 → 오른쪽

덧셈 : +

뺄셈 : -

곱셈 : \*

나눗셈 : /

나머지 : %%

제곱 : ^

함수할때 나오는 에러는 대부분 link error

상수 (Constant = Literal) : 고정된 값, 바뀌지 않는 값

변수 이름은 의미가 있는 이름을 줌

변수의 이름을 바꾸는 것은 값을 바꾸는 것이 아니라 변수 자체를 다시 새로 만드는 것

**Number1 ← 10**

변수명 치환 연산자 값

(L Value) l (R Value)

반드시 변수만 위치 ↓

R Value의 결과를 L Value에 기억시켜라

변수는 기억 공간 / 값은 기억 공간에 기억되는 내용

치환 연산자에는 = 도 있지만 R에서는 사용을 권장하지 않음

가능은 하지만 권장하는 것은 ←

; : 한 줄에 여러 명령을 입력할 때 ;로 하나의 명령을 끝냄

가능하면 한 줄에 명령 하나가 적절 / 가독성이 떨어질 수 있음

Data Type ( 값의 형태 ) , 원시 (원자) Type = Scalar

* 숫자 : 정수, 실수, 산술 연산 가능
* 문자 : 산술 연산이 불가능
* 논리 : TRUE (참) / FALSE (거짓)
* 기타 : NULL, NA

NULL = 타입도 없고 값도 없을 때

NA = 잘못된 값

영숫자 = 모양은 숫자지만 문자

C( = 벡터를 생성하는 함수

암시적 형 변환 implicit/ explicit

: 벡터에서는 동일 자료형 값이 입력되야 함에도 불구하고, 타입이 다른 문자와 숫자가 섞여 있을 때 R 임의로 숫자를 문자로 바꿈

Seq( □,□,□ ) : 순서대로 처음 숫자, 마지막 숫자, 반복되는 범위

Rep( □,□ ) : 반복 숫자, 반복하고 싶은 수

Name → hong kim lee

Index → 1 2 3

|  |  |  |
| --- | --- | --- |
| 90 | 85 | 70 |

Score

ㄴ 원소 (요소)

Names 함수로 의미부여 하면 좋음 반드시 X

이름 다음 대괄호 : 벡터의 요소

소괄호 : 함수의 인수

#

# 벡터 요소 한 번에 여러 개 읽기

#

v <- c( 1, 4, 3, 7, 8 ) # 순서대로 첫번째 두번째 – 다섯 번째

v[ c( 1, 3, 5 ) ] # 첫번째 세번째 다섯 번째 수

v[ 1:3 ] # 첫번째에서 세번째 숫자까지

v[ seq( 1, 5, 2 ) ] # 첫번째 숫자에서 다섯 번째 숫자까지 2 간격으로

seq(1, 5, 2) = 1,3,5

v[ -2 ] # -니까 두번째 숫자만 빼고

v[ -c( 3:5 ) ] # -니까 세번째에서 다섯 번째 숫자 빼고

2020-05-14 오전

팩터형 함수 : 값 하나만 불러올 땐 불편함 levels가 같이 나오기 때문

Warning : 상황에 따라 에러가 날 수 있음 경고 무시 가능

2020-05-14 오후

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1열 2열 3열 4열 5열

행( row )

**관측치**

**(observation)**

**값**

**(data)**

1행

2행

3행

4행

5행

2행 3열

= cell

열 ( column )

**변수 ( variable )**

Obs. = observation

Matrix 데이터의 모양의 같아야 한다 동일 자료형

Data frame 변수의 모양이 다를 수도 있다 자료 다양

구조는 같다

R Data 저장

* 변수
* Vector
* List
* Matrix
* Data frame

Data type

* 숫자
* 문자
* 논리
* Factor
* 기타 NULL, NA

2020-05-15 오전

2020-05-15 오후

2020-05-18 오전

분석 절차

1. 문제 정의
2. 자료 수집
3. 자료 전처리
4. **자료 탐색 (EDA)**

Push 올릴 때

Push 받을 때

JDK (Java Development Kit)

JRE (Java Run-time Environment)

2020-05-18 오후

XML ( extended Markup Language )

JSON ( Java Script Object Notation ) – 요즘 많이 쓰임 구조가 더 간결함

2020-05-18 오후

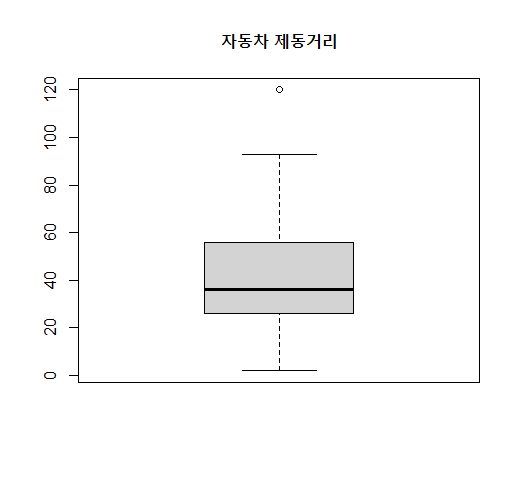
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

최소값 1사분위수 2사분위수 3사분위수 최대값

25% 50% 75%

= 중앙값

데이터 몰려 있음



특이값

2사분위(중앙값)

최소값

3사분위

1사분위

2020-05-19 오전

2020-05-19 오후

변수(독립) ┓ ┏ 절편

Y = wx + b ⇒ 회귀식 ( 직선의 방정식 )

↓ ㄴ 기울기

변수(종속)

Big Data 분석 과정

1. 문제 정의
2. Data 수집
3. Data 전처리
4. **탐색적 Data 분석( EDA )**

-----------------------------------------

1. Data 분석( Data Model 구축 ) ⇒ 머신러닝 / 딥러닝
2. 결과

2020-05-20 오전

1. 문제 정의
2. 자료 수집
3. Data 전처리균
4. EDA
5. Data 분석 / Modeling
6. 보고서