1주차 키워드 정리에 관하여

B2 윤민기

**#0 들어가기에 앞서**

- 개념을 외울 때는 안 까먹게 스토리로 기억할 것

- 컴퓨터는 나의 머릿속에 있는 개념을 구현 해주는 기계임을 잊지 말 것

- 나의 전공 위에 컴퓨터를 얹기 위해 노력할 것

**#1 Computational Thinking 과 인공지능**

**1. 문제 해결이란 무엇인가?**

- input 을 넣었을 때 어떠한 output 이 나올지 예측할 수 있는 알고리즘을 만드는 것.

- 이를 통해 다음에 같은, 혹은 비슷한 문제가 발생해도 해당 알고리즘을 사용, 혹은 참조할 수 있는 것

- 이 과정 속에서 최소의 노력으로 최대의 효과를 내는 것을 의미한다.

**2. Computational Thinking 에 관하여**

- 컴퓨터의 작동 원리에 기반한 사고를 실생활 전반에 적용 시키는 사고 방법론이다.

**- 추상화**

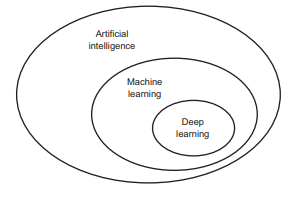
: 일상적으로 추상화란 부정적으로 느껴진다. 사물을 직관적으로 보지 못한다는 느낌마저 든다. 그러나 CT에서 추상화란 자동화를 위한 필수적 과정이다. 객체의 정보, 혹은 동작에서 본질적이고 반복되는 특징을 찾아내는 것, 이를 통해 패턴을 인식하는 과정이 CT에서 말하는 추상화이다.

**- 자동화**

: 추상화를 통해 패턴을 인식했으면 이를 사람의 노력 없이 반복할 수 있도록 구현해야 한다. 한 사람(혹은 한 팀) 의 노력으로 셀 수 없이 많은 사람이 사용할 수 있는 기능, 모듈, 패키지를 구현하는 것, 그것이 자동화다.

**3. Artificial Intelligence & Machine Learning & Deep Learning**

**- Artificial Intelligence**

: AI란 무엇인가? AI에 대한 모두가 동의하는 정의가 없다고 한다. 그 이유는 말 그대로 인공’지능’이기 때문이다. 시대가 흐름에 따라 이전 시대에는 이 정도면 인공지능이라 여겼던 것들이 현대에 와선 더 이상 ‘지능’이 아니다, 라는 선언이 이루어진다는 의미다.

따라서 많은 학자들은 이렇게 말한다.

‘AI’는 아직 까지 실현되지 않은 모든 것이다.

**- Machine Learning**

: 컴퓨터가 명시적으로 프로그래밍 되지 않아도 학습할 수 있도록 하는 연구 분야, 라고 아서 사무엘은 정의했다.

머신러닝은 크게 지도학습, 비지도학습, 강화학습으로 나뉜다.

지도학습은 입력값과 정답 레이블을 함께 주어서 학습하는 방법이다. 이는 분류와 회귀, SVM, KNN 등에서 쓰이며 현재 까지의 기류에서도 머신러닝의 주를 이룬다.

비지도학습은 입력값만 주고 정답 레이블은 따로 주지 않는다. 이는 알파고가 기보 없이 학습한 모델인 알파고 제로에 해당한다. 제로 베이스에서 인간이 기존에 생각지도 못한 결과로 향하는 것 전반이 비지도학습에 해당한다.

강화학습은 지도학습이 정답 레이블을 줬다면, 정답 레이블 대신 보상과 체벌을 주는 방식으로 학습하는 학습법이다.

**- Deep Learning**

: 딥러닝은 이 머신러닝 중에서도 인공신경망을 바탕으로 한 더 세분야된 학습법이다. 기존 인공신경망 방식에서 사전학습법을 추가한 방식을 딥러닝이라고 부른다.

현재 딥러닝이 전국민이 다 아는 용어가 된 것에는 알파고의 영향력이 매우 크다. MLTP 방식에 사전학습법을 추가해 풍부하다 못해 미친듯이 많은 데이터를 주입함으로써 딥러닝이 등장하기 이전에는 상상하지 못했던 결과를 도출할 수 있게 되었다.

**4. Natural Language Processing**

- NLP라 불리는 자연어 처리 기술은 몇 년 전부터 현재까지 대부분의 미래지향적 IT 기업들의 큰 화두이다. 컴퓨터가 알아들을 수 있도록 만든 코딩이 아닌 정제되지 않은 인간 그대로의 문자와 음성을 컴퓨터가 바로 처리할 수 있도록 만든 기술이다.

**5. Vision Technology**

- VISION 은 NLP 와는 반대로 문자와 음성 대신 컴퓨터가 카메라로 보는 시각의 이미지를 분석해 인간이 ‘보는’ 세상을 컴퓨터가 처리, 인간을 훨씬 상회하는 결과를 도출하게 만드는 기술이다.

**#2 운영체제와 프로그래밍**

**6. Operating System**

- OS는 하드웨어를 관리하기 위해 고안된 시스템 소프트웨어로, 응용 소프트웨어 실행을 위한 환경을 조성한다.

**7. Programming**

- 프로그래밍은 추상 알고리즘을 특정 프로그래밍 언어로 컴퓨터 프로그램에 구현하는 일련의 과정을 의미한다.

**8. ASCII CODE**

- 사람과 컴퓨터 사이의 약속을 의미한다.

- 0부터 127까지 숫자에 영어 대소문자, 특수기호, 숫자 등을 각각 저장하여 호출을 용이하게 만든다.

**9. UNICODE**

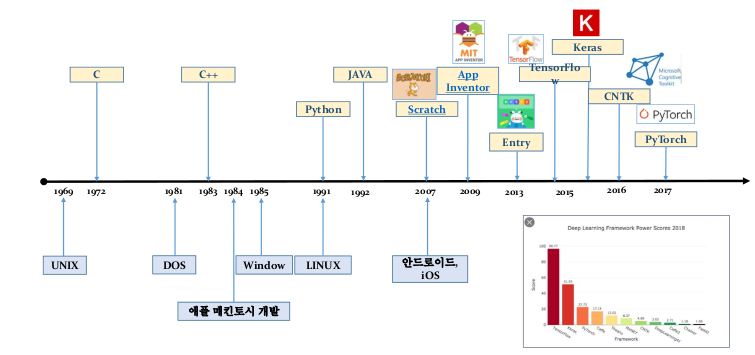
- 글자와 코드가 1:1 mapping 되어 있는 code table 을 의미한다.

**10. Parameter**

- 매개변수는 함수에 넣는 INPUT 값을 의미한다.

- 함수를 선언할 때 입력을 받아야 할 값의 수, 자료형 등을 정의할 때도 파라미터라는 용어를 쓰기도 한다.

**11. OS와 Programming의 역사**



- 하드웨어를 제어하기 위해 최초의 OS UNIX가 등장했다.

- 그러나 1972년 이전 언어로 효과적인 장치 제어에 한계를 느꼈고, UNIX를 보다 잘 사용하기 위해 데니스 리치가 주축이 되어 C 언어가 탄생하였다.

- 1981년 DOS 등장 이전의 컴퓨터는 과도한 가격과 크기로 인해 정부, 대기업에서나 사용하는 특권적 기술이었다. MS 에서 개발한 DOS 의 등장 이후 Personal Computer 의 시대가 열린다.

- 그러나 여전히 CLI 기반의 사용자 친화적이지 않은 UI는 또 다른 진입 장벽을 가져왔고 이를 애플의 매킨토시가 GUI 를 통해 해결한다. 현대적 모니터 화면의 탄생이었다.

- 이에 질세라 WINDOW가 등장하였고 당시에도 있었던 애플의 감수성이 아닌 대중성으로 승부를 던져 PC시장을 장악, 2000년대 중반까지 최고의 위치를 달리게 된다.

- 귀도 반 로섬이 크리스마스에 심심하여 PYTHON을 개발한다. 현재 PYTHON은 언어 사용 1,2위를 다투고 있다.

- SUN 에서 embedded 하드웨어 조작을 위해 JAVA를 개발했다. 본래의 목적은 물론 웹개발에 널리 통용된다.

- PC의 시대를 지나 스마트폰의 바람이 불면서 IOS와 ANDROID가 세계 IT시장을 잠식한다. MS는 당시 회장의 판단 미스로 미래가 없는 기업으로 평가 받게 된다.

- SCRATCH, APP INVENTOR, ENTRY 등 교육용 스크립트 언어 & 싸이트가 등장하게 된다.

- 구글의 TENSORFLOW를 시작으로 MIT의 KERAS, MS의 CNTK, NLP 처리에 유리한 Pytorch가 등장한다. 바야흐로 AI OPEN SOURCE FRAMEWORK & LIBRARY 가 보편화 되었다.

- 그렇다면 미래의 기술, 가치는 어디에 있을까? 이를 끊임없이 생각해보는 것 역시 이번 POSCO BIGDATA ACADEMY 의 큰 목적 중 하나라 할 수 있겠다.

**#3 왜 파이썬인가?**

**12. Python 의 특징**

- python 은 뼛속부터 객체인 객체지향 언어이다.

- C가 주는 강력한 능력을 보다 편리하게 사용하기 위해 사용되었다.

- 백엔드, 인공지능, 빅데이터 가릴 것 없이 모두 효과적이면서 빠르게 처리하며

각종 라이브러리를 포용하기 쉽다는 장점 때문에 점점 더 주목 받고 있다.

**13. Python 에서의 Variable**

- 변수는 일반적으로 데이터를 저장하는 공간을 의미한다.

- C의 경우 compile time에 static 하게 자료 공간 요청을 확실하게 한다.

- python 에서의 변수는 보다 구체적으로 객체를 가리키는 id 값을 담고 있는 저장 공간을 의미한다.

- 데이터타입 선언이 필요가 없으며 Dynamic typing을 이용하기 때문에, 런타임에 구문을 통해 data type 설정한다. 이 과정 속에서 선언 배정이 동시에 발생한다.

- 이 때 변수의 데이터 형식은 값을 넣는 순간마다 변경될 수 있는 유연한 구조를 갖는다.

**14. Shallow copy vs Deep copy**

- shallow copy 는 변수를 복사할 때 눈에 보이는 것만 늘어났을 뿐 주소가 같다.

a = [1, 2, 3, 4, 5], b = a, b = [1, 2, 3, 4] 시, a 역시 [1, 2, 3, 4]가 된다.

- deep copy 는 변수를 새로 선언했을 때 주소 역시 새로 생기는 경우를 의미한다. 이 때 완전히 독립된 변수로서 각각의 변화가 서로에 아무런 영향을 끼치지 않는다.

**15. input and output**

- c의 printf() (f 는 format을 의미한다) → python의 input()

- c의 scanf() (f 는 format을 의미한다) → python 의 output()

**16. operator overloading** (연산자의 재정의)

- 언어에 built-in 된 자료형 뿐만 아니라 직접 만든 class 에서도 원하는 연산자 기능을 구현하기 위해 operator overloading 통해 덧셈, 뺄셈, or, and 등의 연산자를 구현할 수 있다.

- 이때 호출을 위한 함수는 \_\_or\_\_, \_\_and\_\_ 등등이 있다.

**17. Data Structure**

- stack 구조: LIFO 형태의 자료구조이며 맨 끝의 항목이 가장 먼저 나간다.

- queue 구조:FIFO 형태의 자료구조이며 맨 앞의 항목이 가장 먼저 나간다.

**18. 구글과 네이버의 검색 알고리즘 우선순위**

- 핵심은 정렬이다.

- 구글의 경우 page-link ranking algorithm

- 네이버의 경우 광고 위주 algorithm

**19. 함수**

- 함수의 architecture 를 설계하는 것 → 가장 중요하다

- 함수는 왜 만들까?바로 dividing conquer(분할 정복)를 위해서다.

**20. recursive function calling**

- 함수 내부에서 자기 자신을 계속 호출하는 방법을 의미한다.

- 변수를 조금씩 변경하면서 연속적으로 반복된 연산을 할 때 유용하다.

- 하노이탑의 전설이 그 대표적인 예시다.

**21. OOP**

- 객체지향 프로그래밍이란 뜻으로, 기존 C 의 절차형 프로그래밍이 재사용이 불편하고 사람의 실제 세계와 컴퓨터의 언어가 너무 동떨어져 있다는 지적에서 시작했다.

- 실제 세계의 구조에 맞춰 코드를 구현하는 방식을 의미하는 거대한 프로그래밍 패러다임이다.

**22. Class, Object**

- Python 은 다 객체다!

- class 로부터 생성된 것을 object 라 부른다.

- function 만 있어도 객체고, variable 만 있어도 객체다

- constructor: instance를 생성하면 자동으로 호출되는 함수

constructor 는 \_\_init\_\_() 라는 이름을 갖는다.

- Object: Data + Function 을 분리할 수 없이 하나로 처리하는 동작

→ 컴퓨터를 사람의 관점으로 변환한 것이다.

- Instance: 객체로부터 실체가 된 것이다.

- instance 객체는 오직 한가지 연산만 지원한다.(Dot 연산)

- class 객체는 두 가지 연산을 지원한다.(Dot 연산, () 연산)

- method 개체는 일반 함수 객체와 달리 클래스에 종속되어 있는 함수를 지칭한다.

- 객체: 객체는 실생활에 존재하는 실제적인 물건 또는 개념을 의미한다.

- 속성: 속성은 객체가 가지고 있는 변수다.

- 행동: 행동은 객체가 실제로 작동할 수 있는 함수, 메서드를 의미한다.

**- Struct vs Class** : 둘 모두 User-defined data type 을 의미한다.

- instance variable vs class variable

: class variable 은 모든 각각 instance화된 변수들이 공유하는 값이 필요할 때 선언해 준다. 즉 class 를 통해 만들어진 instance 변수들이 공유하는 방이라고 볼 수 있다. instance 는 생성자 안에서 self로 접근하고, class 는 클래스 이름으로 접근한다.

SOLID: 객체지향 언어에서 추구해야할 5가지 원칙이 있다.

이를 SOLID 원칙이라고 부르는데,

S: Single Responsibility Principle, 소프트웨어의 클래스, 함수는 단 하나의 책임만 가져야 한다.

O: Open-Closed Principle, 기존의 코드를 변경하지 않고 기능을 수정하거나 추가할 수 있도록 설계해야 한다.

L: Liskov Substitution Principle, 자식 클래스는 부모 클래스에서 가능한 행위를 수행할 수 있어야 한다.

I: Interface Segregation Principle, 한 클래스는 자신이 사용하지 않는 인터페이스는 구현하지 말아야 한다.

D: Dependency Inversion Principle, 의존 관계를 맺을 때, 변화하기 쉬운 것보단 변화하기 어려운 것에 의존해야 한다.

**23. Inheritance**

- 기존 class의 필드와 메소드를 그대로 물려받는 새로운 class를 만드는 것이다.

- 재사용성을 비약적으로 높여주는 방법 중 하나다.

**24. Method Overriding**

- 중요!

- 자식 클래스가 부모 클래스의 메서드를 특정한 형태로 다시 구현하는 것을 의미한다.

**25. Operator Overloading**

- 연산자 중복: 사용자 정의 객체에서 필요한 연산자를 내장 타입과 형태와 동작이 유사하도록 재정의

- \_\_add\_\_(), \_\_sub\_\_(), \_\_mul()\_\_ 등을 사용한다.

- 교환법칙 성립하고 싶다면 \_\_radd\_\_(), \_\_rsub\_\_(), \_\_rmul\_\_() 등을 사용한다.

**26. Polymorphism**

- 설계도를 그리는 것이 정말 중요하구나!

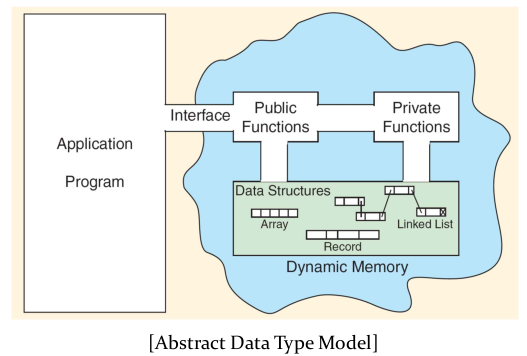
- inheritance 관계 내의 다른 class들의 instance 들이 같은 멤버호출(같은 방법)에 대해 각각 다르게 반응하도록 하는 기능을 의미한다.

- Method overriding, Operator Overloading 도 polymorphism 을 지원하는 중요 기술이다.

**27. 추상 데이터 타입( Abstract Data Type)**

- Dynamic memory allocation vs Static memory allocation

: run time 에서 memory 할당하는 방식 vs compile time 에서 memory 할당하는 방식의 비교이다.

- 관계있는 애들끼리 묶어서 encapsulation (information hiding)을 한다.

Dynamic Memory에 기반을 둔 공개 함수, 비공개 함수가 어플리케이션 사이에 인터페이스를 두고,

데이터와 함수를 주고받는 관계를 추상화한 모델을 의미한다.