딥러닝 수업

▼ 설치 및 이용방법

아나콘다 & 주피터 노트북

개념

아나콘다 : 가상환경 관리도구

- 관리 : 가상환경에 설치 된 라이브러리 생성, 삭제, 업데이트 가능

가상환경을 만든다=컴퓨터 안에 여러 대의 컴퓨터를 만든다

가상환경을 왜 만드냐?

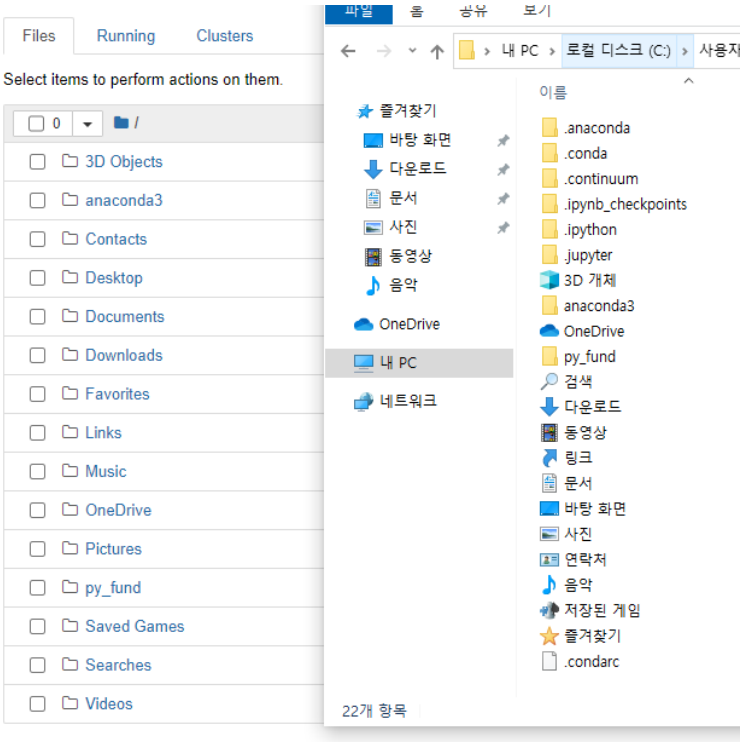
- 라이브러리 간의 버젼 차이로 호환이 안 됨

가상환경이 PC 상의 표현

주피터노트북 URL : http://localhost:8890/notebooks/py_fund/day1.ipynb

주피터노트북 : localhost:8890

아나콘다 가상환경 생성 시 폴더로 만들어진다. 이 때 이름이 localhost:8890



▼ 자료형

▼ 숫자형

1. 숫자형 자료 활용 연산자
    a. 사칙연산

    1+2
    1-2
    1*2
    1/2
    1.0//2.0

    b. 나눗셈 후 몫, 나머지, 거듭제곱
    print(5//3) # 몫
    print(5/3-((5/3) - int(5/3))) # 정수 구하는 것
    print(5%3)# 나머지
    print((5/3)- int(5/3)) # 소수점 구하는 것
    print(5**3) # 거듭제곱

    c. divmod : 몫과 나머지를 나타냄

    q,r = divmod(5,3) # q=1, r=2

2. 숫자형
    a. 정수형

    int(3/2) # 1
    int(1.67) # 1
    '3'+2 #3의 자료구조 문자형이므로 에러
    type('3') #str
    a=int(input()) #3
    #input 이라는 함수를 이용하면 모두 문자를 읽어드린다. 수치 데이터와 같은 경우에는 문자로 인식 되기 때문에 문자를 숫자로 변환해주는 작업을 반드시 해야한다.

    b. 실수형

    float(1) #1.0
    1.23E5 #1.23*10^5
    1.23E-10 #1.23*10^-10

    c. 8진수와 16진수

있다는걸 알아만 두자

▼ 문자형

1. 문자열이란

   문자, 단어 등으로 구성된 문자들의 집합을 의미한다.
   ’life is short’
   ’a’

2. 문자열을 어떻게 만들고 사용할까?

   a. 문자열 안에 작은 따옴표 포함

      ‘Life\’s is short’ #백슬래시 이용해서 문자열에 포함 시킴

   b. 여러 줄인 문자열을 변수에 대입하고 싶을 때

      ‘Life is short\nYou need python’ #\n 사용해서 줄바꿈
      ’’’Life is short
      You need python’’’ # ‘’’ ~~~’’’

3. 문자열 연산하기

   a. 문자열 더해서 연결하기

      a=’python’
      b=’is’
      c=’good’
      a+b+c #’python is good’

   b. 문자열 곱하기

      ‘-’*50 # ——

   c. 문자열 길이 구하기

      a = ‘life is short’
      len(a) #17 자주사용함

4. 문자열 인덱싱과 슬라이싱

   a. 문자열 인덱싱

      a=’python programming is fun’
      a[1]=y
      a[-1]=n

   b. 문자열 슬라이싱

a[0]+a[1] +a[2] #pyt
a[0:3] #pyt
a[-3:0] #fun

c. 문자열 변경

s='claudy' # a를 u로 수정하고 싶다.
s[2]='o' # 에러발생 문자는 특정위치 변경 불가.
위의 글자를 하나의 상수처럼 생각한다.
s='claudy'
s.replace('a','o')
s[:2] +'o'+s[3:]

5. 문자열 포매팅

a. 문자열 포매팅 따라하기

'오늘은 파이썬 코딩을 x시간 했다.' # x를 변수로 잡는 것

b. 문자열 포맷코드

t=3
day = '이틀'
'오늘은 파이썬 코딩을 %d시간 했다.' %3 #숫자바로대입
'그리고 %s간 머리가 아팠다.'%day #변수로잡고 문자열 대입

c. 포맷코드 숫자와 사용하기

'오늘은 파이썬 코딩을 %d시간 했다. 그래서 %s간 머리가 아팠다.'%(t,day)

d. 문자열 포매팅
[포매팅 연산자 %d와 %를 같이 쓸 때는 %%를 쓴다]
"Error is %d**%%**." % 98

e. num=3
day=’five’
print('i ate %d apples. so i was sick or %s days'%(3,'five'))
print('i ate %d apples. so I was sick for %s days'%(num,day))
print('i ate {0} apples. so i was sick or {1} days'.format(num,day))
#format(인덱스)
print('i ate {n} apples. so i was sick or {d} days'.format(n=2,d='six'))

f. 정렬

```
msg='hello'
print('{0}'.format(msg))
print('{0}'.format('hello'))
print('{0:<10}'.format('hello')) #왼쪽정렬
print('{0:>10}'.format('hello')) #오른쪽정렬
print('{0:^10}'.format('hello')) #가운데 정렬
print('{0:@^10}'.format('hello')) #가운데 정렬 이후 공백 부분을 다른 문자로 채워넣을 수 있다.
print('{0:%>10}'.format('hello')) #오른쪽정렬
```

6. 문자열 관련 함수들
   a. count

```
#count() : 문자열에서 특정 문자의 개수를 셀 때 사용
'data'.count('a')
#문자열.count('특정문자')
m='data'
m.count('a')
'data'.count('')
```

   b. find

```
#find():
m.find('a') #1
m.find('z') #-1 발견되지 않는 경우
```

   c. index

```
#index():
m.index('a') #정상 검색의 경우에는 find 함수와 같다.
m.index('z') #에러 발생
```

   d. join

```
리스트 입력, join함수 , 문자열 출력
x='a,b,c'
r=x.split(',')
r
#split <-> join

#join
''.join(r) #abc
','.join(r) #a,b,c
```

```
r2=",".join(r) #
r2
```

e. upper/lower/capitalize

```
#capitalize,upper,lower
"abc".capitalize() #Abc
"abc".upper() #ABC
'ABC'.lower() #abc
```

f. lstrip/rstrip/strip

```
#lstrip, rstrip, strip
a='         test     exam '
a.lstrip() # 왼쪽
a.rstrip() # 오른쪽
a.strip() # 양쪽
'hi  '=='hi'
'hi  '.rstrip()=='hi'
```

g. replace

```
#.replace(old,new)
msg='Python is too easy'
msg.replace('easy','difficult')
msg.replace(' ','') # 공백 모두 제거
msg #원본은 변경 안됨 -> 변수 안에 있는 것은 변경 되지 않는다.
msg=msg.replace('easy','difficult')
msg
```

h. split

문자열 입력, split함수 , 리스트 출력
리스트 입력, join함수 , 문자열 출력

```
"10 20" # 공백 문자로 분리 된 문자열
"10 20 30".split() # 공백문자로 분리 된 문자열
'010-1234-5689'.split('-')

함수의 생김새
#출력값=함수명(입력값)  보편적인 함수를 사용하는 방법
#적용대상.함수명(입력값)  split 함수가 이렇게 생겼다.

phone=input('휴대폰 번호를 입력하세요 : 010-1234-5678')
```

```
x=phone.split('-')
print('통신사 :',x[0])
print('국번호 :',x[1])
print('전화번호 :',x[2])

d='20220819cloudy'
print('연도 :',d[:4])
print('월 :',d[5:6])
print('일 :',d[6:8])
print('날씨 :',d[8:])
```

map

#map함수 : split 된 문자열 각각에 대해 특정 함수를 일괄적으로 적용

```
x=input('수 2개를 입력하세요:').split()
print(x) # 여기서 리스트로 자동으로 받는건 뭐임?
int(x[0])
int(x[1])
int(x[0])-int(x[1])
```

#map(적용하고자하는 함수명, 대상)

```
a,b=map(int,x)

a,b=map(int, input('수 2개를 입력하세요').split())
print(a-b)
```

▼ 리스트(collections)

scores=[90,80,100,90] #[]:리스트

[1,2,'a','b','hello',[100,200],{'a':1,'b:2'}] # 리스트 안 문자, 숫자, 리스트, 딕셔너리 모든 자료구조형이 다 올 수 있다.

x=[10,20,['a','b','c',['hello','hi']],30]#x는 리스트, 요소개수 : 4(0~3번 index)

```
x[2][3][1] #hi
x[0:2] # 10, 20
```

리스트 연산

```
a=[1,2]
b=[3,4]
a+b #[1,2,3,4]
```

리스트 생성

list() # 리스트 생성해주는 함수

#연속 된 숫자를 생성함수 0~19로 초기화 된 리스트 생성
range(5)
range(5,10)
range(10,20) #10~19까지
list(range(10,20)) # range 객체를 리스트 함수에 넣으면 됨
list(range(10,20,2)) # 2씩 증가함
list(range(20,10,-1)) # 1씩 감소함

a=list(range(5,51,5))

a[1] #10 추출
a[2] #20 추출
a[:10:2] #2칸씩 띄어서 추출하고 싶을때
a[5::2] # 5번째 인덱스에서 끝까지 2씩 증가 출력
a[6:2:-2] #5번째 인덱스에서부터 2번째 인덱스까지 역순으로 출력

#한 단어씩 출력해라
x='data'
x
x[0]
x[1]
x[2]
x[3]

list(x)

리스트 수정

a=[10,20,30]
a[1]=15

리스트 추가

append()
a=[10,20,30]
#a[3]=40 #a리스트에 인덱스의 범위가 벗어남

#a+[40] 뒤에 리스트를 붙일 수 있음
a.append(40)#리스트 마지막에 데이터를 추가하는 함수
#replace처럼 함수를 변경한 결과를 재차 변수에 대입해야 하는 함수도 있고
#append처럼 바로 추가되는 함수가 있다.
#a.append(50,60) 에러 발생 append는 한개만 추가 가능하다
#a.append([40,50]) #이런식으로 추가는 가능함
a

a.extend([40,50,60]) #여러개를 추가할 수 있음 다만 자료형을 리스트 처리를 해야 함
a

리스트 정렬

a=[30,10,20,40]
#정렬(sort):정한 순서에 따라 데이터를 나열하는 것
#오름차순 정렬/내림차순 정렬
#오름차순 : ㄱ~ㅎ, ㅏ~ㅣ,0~9,A~Z,a~z
#초성/중성/종성
#각~힣
a.sort()
a

a=['k','c','a','b']
a.sort()
a

a=[10,20]
a.append(30)
a

a.sort(reverse=True) #내림차순
a

리스트 제거

a
a.remove(35)
#만약 동일 데이터가 있을 시 앞에 데이터가 삭제됨
#변수에 재차 대입 없어도 바로 변수에 바로 적용

a.pop(3) #리스트 인덱스 안을 제거할 수 있음
a

리스트 검색

a=[10,20,30]
a.index(20)

▼ 튜플(collections)

scores=(90,80,100,90) #():튜플

t1=()
t1
t2=(1)
t2
type(t2) #int
t3=(1,) #쉼표를 넣어야지 튜플이 된다.
type(t3) #tuple

t4=(1,2)
t4
type(t4)

튜플 인덱싱

t5=(1,2,(4,5))
t5[0] # 튜플에서도| 인덱스는 대괄호 사용

튜플 삭제/변경 불가

del t5[1] #튜플 내 요소는 삭제되지 않는다.
t5[1] = 2 #튜플은 불가능

▼ 딕셔너리(collections)

딕셔너리 ={키:값,키1:값1,키2:값2}

데이터에를 표현을 할 때 단순한 숫자들로만 저장하는 것보다 처리에 대해 조금 더 용의
#딕셔너리와 저장했을 때 리스트와 저장했을 때의 차이
mycar={'cc':'2000','drv':4,'price':2000}
mycar=[2000,4,2000]

scores={'s1':90,'s2':80,'s3':100,'s4':90} #{}:딕셔너리

data={'name':'kim','age':20,'addr':'seoul'}
data
#키 중복 허용, 키 중복 시에는 마지막 키,값이 저장
#값은 중복 허용

딕셔너리 값 불러오기

mycar={'cc':'2000','drv':4,'price':2000,'model':{'se':2000,'re':3000}}
mycar['model'] #모델 키, 값을 불러오기
mycar['model']['se'] #딕셔너리 안에 딕셔너리 값 불러오기

#딕셔너리 생성하기
my_dict={}
dict() # 비어 있는 딕셔너리 생성
mycar=dict(cc='2000',drv=4,price=2000) #약간 차이가 있음 안쓸듯
#보통 생성보다는 파일을 불러오는게 더 많음

#딕셔너리 키값 확인
'cc' in mycar
'aa' in mycar

#딕셔너리 키의 개수 확인

len(critics)

#리스트는 튜플, 문자열, range함수 : 연속적인 값들이 저장(index로 저장가능)
#딕셔너리는 비연속적인 자료 : index 데이터 참조 불가능

in, not in
#데이터 in 리스트(문자열, 튜플, range)
#시퀀스형 자료 : 리스트, 문자열, 튜플, range
#인덱스, 슬라이싱이 가능하다.
#len 함수 이용 가능
30 in (10,20,30)
30 in range(1,30)

▼ 불

Boolean : True, False

3>1 #비교연산자 -> Boolean type
3==3
3!=3
'python'=="Python"

==, != 값 자체가 같은지 비교한다.

5 == 5
5 == 5.0

#is, is not은 객체를 비교
5 is 5
5 is 5.0 #type이 서로 다르다.

논리연산자

#논리연산자 : and, or, not
and 연산자 : 피연산자가 모두 참 -> 참, 나머지는 모두 거짓
or 연산자 : 피연산자가 모두 거짓 -> 거짓, 나머지는 모두 거짓
not 연산자 : 논리값을 반대로 뒤집어 출력

True and False #False
True and True #True
True or False #True
not True #False

not True and False or not False
#False and False or True
#False or True

#True
((not True) and False) or (not False)

#연산 우선순위 : not > and > or

Bool 자료형의 참과 거짓

#bool 함수 : 실수, 정수, 문자열 -> boolean값 변환
bool(5) #숫자형에서는 0제외 모두 True
bool(0)
bool(0.231685135)
bool('abc') #빈문자열 제외 모두 True
bool('')

'ab' and 'cd' #cd 뒤에것이 나옴
True and 'cd' #cd 뒤에것이 나옴
'cd' and True #True

False and 'ab'
0 or False

▼ 변수

1. 변수는 어떻게 만들까?

   변수이름 = 변수에 저장할 값
   a=1
   b=’python’
   c=[1,2,3]
   변수를 만들 때는 = (assignment) 기호를 사용한다.

2. 변수란?

   x=1 #x라는 저장소가 만들어지고, 1이 저장된다.
   변수이름 = 값
   변수이름 정의 규칙
   대소문자 구분
   영문자 + 숫자 사용가능, 한글변수명 가능하지만 에러 많이 발생
   미리 약속된 단어(if, for…)는 변수명으로 사용 불가

   변수 다양한 저장방법
   a=b=c=1 # a=1, b=1, c=1
   ———————————————————-

a=1, b=2
a,b = b,a # a=2, b=1
————————————————

k=3 하고 k=3이란 식을 지워도 #k=3이 여전히 출력(메모리 남아있음)
del k # 변수 삭제
————————————

j = None #null 상태로 만듬
————————————————————-

3. 리스트를 복사하고자 할 때
   a. :
   b. copy
4. 변수를 만드는 여러가지 방법
5. Print 함수

   print('hello','my love','it\'s me',sep='&') #구분자(seperator)를 넣을 수 있다.
   print('hello','my love','it\'s me',sep=' ')
   print('hello','my love','it\'s me',sep='\n') #제어문자도 사용가능하다.
   print('hello','my love','it\'s me',sep='\t')
   sep인수는 내용 사이에 삽입할 문자열을 , end 인수는 내용 마지막에 출력할 문자열

   제어문자

   \n : 줄 바꿈
   \t : 탭 간격

   ---

   print(1) #기본 옵션 값 print(end=\n) -> 그래서 줄바꿈이 실행된다.
   print(2)
   print(3)
   print(1,2,3,sep='')
   print(1, end='*') #다음 함수와 이어서 출력 됨
   print(2, end='')
   print(3)

▼ 연습문제

▼ 컴퓨터 공학 지식

why?

0.1 + 0.2 == 0.3 False 일까?

1. 사람은 10진법, 컴퓨터는 2진법을 사용한다.
2. 10진수에서 2진수로 변환했을 때, 무한 소수인 경우가 있다.
3. 무한 소수인 경우에 입력받은 10진수 값에 가장 근삿값이 되도록 반올림 한 값으로 저장된다.
4. 0.1, 0.2, 0.3은 모두 2진법으로 변환했을 때 , 무한 소수이다.
5. 0.1과 0.2를 변환하면서 생긴 오차 때문에, 0.1과 0.2를 각각 2진수로 변환하고 합한 값과 0.3을 2진수로 변환한 값 간에 오차가 생긴다.
6. 때문에, 0.1 + 0.2 === 0.3은 같지 않게 된다.

▼ 내가 궁금했던 것

int(print(5/3)) # 문법적으로 잘못 print 함수에서 나온 결과는 숫자형식이 아닌 단순한 화면출력

▼ Big → fast

1. 우리가 앞으로 해야 될 일은 "BIG" 이다. 빅데이터이기 때문에 속도를 줄이는게 중요!
   a=a+1 # 반드시 할당해야지 값이 변경된다. , cpu가 연산을 두번한다.
   a+=1 #cpu 연산한번
   a+=1 같은의미임 # cpu가 연산 한번한다.
2. 튜플은 리스트보다 빠르다. 내부구조도 훨씬 단순하고, 읽는속도도 빠르다.

▼ trend

few shot learning

▼ 코딩파일

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/b4c1f506-65f2-4ca4-bd92-6072999010d3/day1.ipynb

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/f303fc7b-0433-4ef9-96da-b93487981457/Untitled.txt

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/d4f8514a-6791-476e-a8d1-8415b0e2b483/day3.ipynb

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/331d8a88-f80f-479f-b897-f5f042dc2c14/day4.ipynb

▼ 과제제출

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/7affebce-2a81-4927-b72e-78c8061531f0/Untitled.txt

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/5a8ae3df-13c7-4a28-93b2-ba10f3fd77e5/day2_%EC%97%B0%EC%8A%B5%EB%AC%B8%EC%A0%9C.ipynb

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/65aa31eb-f618-488b-aa35-cd5303f93e7d/%EB%94%A5%EB%9F%AC%EB%8B%9D_%EC%9C%A0%EB%8F%99%ED%9B%88_day_3_%EA%B3%BC%EC%A0%9C_%EC%A0%9C%EC%B6%9C.ipynb

▼ 함수

range함수

range(시작하는 숫자, 끝나는 숫자, 증가분)

#연속 된 숫자를 생성함수 0~19로 초기화 된 리스트 생성
range(5)
range(5,10)
range(10,20) #10~19까지
list(range(10,20)) # range 객체를 리스트 함수에 넣으면 됨
list(range(10,20,2)) # 2씩 증가함
list(range(20,10,-1)) # 1씩 감소함

map

#map함수 : split 된 문자열 각각에 대해 특정 함수를 일괄적으로 적용

```
x=input('수 2개를 입력하세요:').split()
print(x) # 여기서 리스트로 자동으로 받는건 뭐임?
int(x[0])
int(x[1])
int(x[0])-int(x[1])
```

#map(적용하고자하는 함수명, 대상)

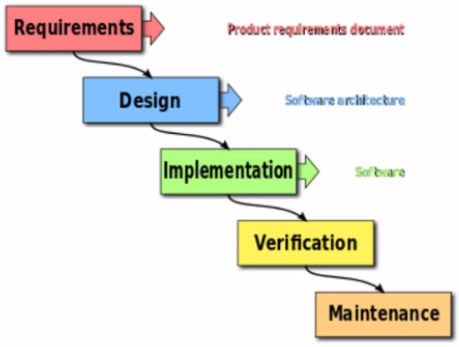
```
a,b=map(int,x)
```

```
a,b=map(int, input('수 2개를 입력하세요').split())
print(a-b)
```

▼ 취업특강

1. 역사

   폭포수 모델



폭포수 모델

- 사전 기획 – 제작 – 포스트 프로덕션 이러한 방식은 기획에서 완성까지 폭포수 모델(Water Fall)의 프로세스를 따라 제작하는 것과 유사하다고 볼 수 있다.
- 폭포수 모델은 순차적인 소프트웨어 제작 프로세스로 한 단계, 한 단계를 진행해 나가야 한다.
- 폭포수 모델은 제작 단계의 이해가 쉽고 체계적으로 문서화와 프로세스 진행 단계를 명확히 파악할 수 있다.
- 폭포수 모델의 단점은 실제 완성 전까지 실행이 어렵고 각 단계가 완벽히 마무리 되었는지를 정의 내리기가 어렵다.
- 초반 설계의 문제가 구현의 난이도를 예측하기 힘들다.
- 기술적 난이도가 높은 R&D 등을 진행해야 할 때 새로운 기술을 도입해야 할 때 폭포수 모델은 지연을 일으킬 수 있다.

폭포수 모델의 부작용

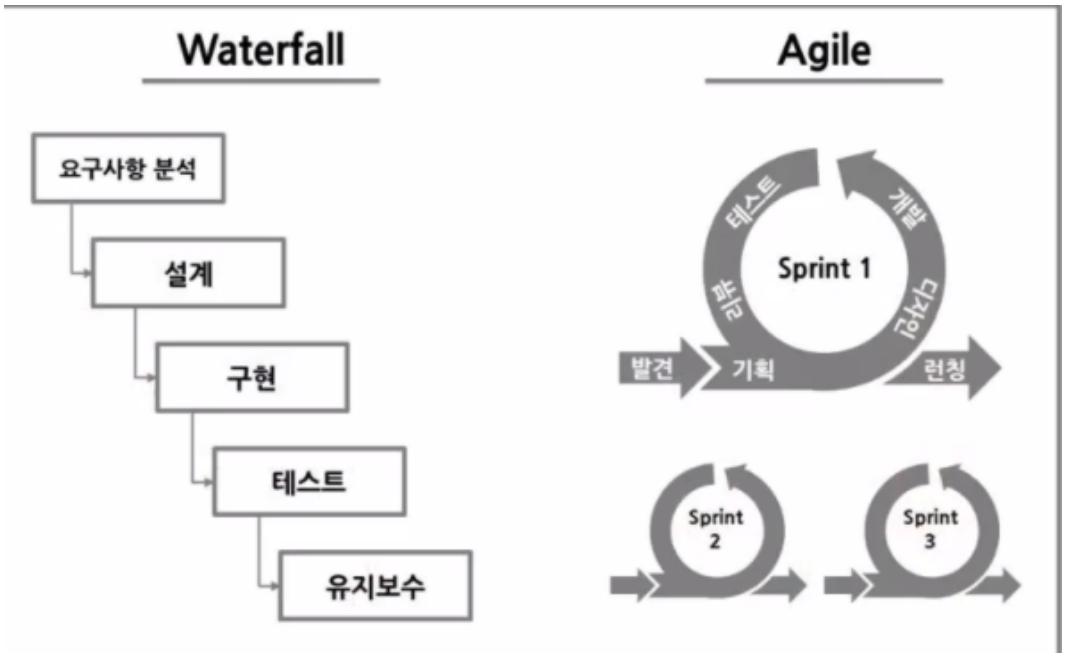
폭포수 모델의 단점은 실제 완성 전까지 실행이 어렵고 각 단계가 완벽히 마무리 되었는지를 정의 내리기가 어렵다.

초반 설계의 문제가 구현의 난이도를 예측하기 힘들다.

기술적 난이도가 높은 R&D 등을 진행해야 할 때 새로운 기술을 도입해야 할 때 폭포수 모델은 지연을 일으킬 수 있다.

4

from 폭포수 모델 to 애자일 방식



큰 물고기가 작은 물고기를 잡아 먹는 것보다 빠른 물고기가 느린 물고기를 잡아 먹는 시대

애자일의 도입은 체질변화를 기대하는 조직이 많이 시도하는 문화로 전통적인 기업들이 IT 개발 뿐만 아니라 조직 문화 전반에 걸쳐 도입을 시도 중

애자일은 조직마다 모두 다른 모습을 갖고 있으며 특정 산업 분야, 특히 한국형 애자일 모델은 없다.

개발 방법론은 익스트림 프로그래밍, 스크럼 등 다양한 방식의 방법론이 존재하며 30일 마다 동작 가능한 제품을 제공하는 스크럼 기법을 주로 도입

코드관리 시스템을 도입한다는 것은 결국 제작 팀의 파이프라인 변화를 추구하게 되고 이러한 파이프라인을 가장 효과적으로 사용하려면 민첩한 소프트웨어 제작 공정이 도입되어야 한다.

함께 적용할 수 있는 애자일 기법 역시 부분적으로라도 도입을 고려해 볼만 하다.

수평적인 조직문화가 기반이 되어야 가능

소프트웨어 영역에서의 문제점 출몰(ex : 버그/디버그, 업데이트, R&D...)

코드 관리 시스템 도입으로 인한 서비스의 영역으로 확대

코드 관리 시스템 도입으로 인한 프로그래머 또는 <트러블슈팅>의 개념 도입

코드 관리 시스템 도입으로 인한 파이프라인의 변화 및 제작 공정의 변화

제작 공정의 변화로 인한 프로젝트 매니저의 개념 도입

스크럼 또는 허들로 표현되는 단기 스프린트 제작 방식의 도입

제작된 에셋 및 프로젝트의 백로그에 대한 이해

워터폴 또는 애자일의 접근 이점을 모두 취하는 하이브리드 형태의 프로젝트 관리 방법 도입

JIRA, Slack, Trello 등의 프로젝트 관리 툴 및 BTS(Bug Tracking System)의 도입

VCS 등을 통한 통합적 코드 및 데이터 관리(SVN, Git, 퍼포스 등의 도입)

VCS Merge 전문가 및 QA 의 필요성 증가

협업툴 추천

Jira, Slack, Trello, Notion, Zeplin, SVN, Git

2. Git이란 무엇일까?

깃의 필요성

버전 관리 없이 문서를 작성한 경우

문서나 설계도, 소스 코드 등의 변경점을 관리해주는 소프트웨어.

소프트웨어 등을 작성할 때 변경점을 관리하는 것은 말할 필요도 없이 아주 중요

언리얼 엔진은 결국 하나의 소프트웨어 제작 플랫폼이므로 담겨진 모든 콘텐츠를 구분 없이 소프트웨어로 이해한다.

변경점 : 기존 코드와 변경 된 점을 체크하는 기능(스프레드시트 수정 로그 남아 있는 것)

- **버전 관리를 함으로서 얻을 수 있는 것**
- 변경점 관리: 어떤 내용을 누가 작성해서 어느 시점에 들어갔는지 확인할 수 있게 해준다.
- 버전 관리: 특정 시점에 꼬리표(Tag)를 달아 버전을 표시해주고, 브랜치(Branch) 등으로 동시에 여러 버전을 개발할 수 있게 해준다.
- 백업&복구: 무언가가 잘못되었을 때 다시 특정 시점으로 돌아가게 해주고, 사고로 내용이 날아간 경우에도 복구할 수 있게 해준다.
- 협업: 같이 일하는 사람들에게 수정사항을 쉽게 공유

버전컨트롤 시스템을 해야하는이유

버전컨트롤 종류

1. 로컬

   local vcs

2. 네트워크

   서버가 별도로 있고 클라이언트가 중앙서버에서 파일을 받아서 사용하는 방식

3. 분산버전관리 시스템

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/d737e87e-6686-45d6-9237-f50a8d561558/GitHub_%EA%B5%90%EC%95%88.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/5a8bbb7c-0cf2-42ab-93b2-c6230e99f2cd/GitHub_%EA%B5%90%EC%95%88v2.pdf

카페24 서비스와 유사 → 오픈 소스 → 유명세(팔로워 up) → 깃허브 '인용'

깃허브 '개발자들의 논문' → 개발자의 논문

깃허브를 인스타그램을 비교해서 설명하는 방식이 새롭다.

요즘에는 주로 개발 산업 취업 포트폴리오로도 활용된다.

2. Git을 사용하는 개발자의 문화
3. Git 활용법
4.