



Python



Python

프로그래밍 언어 실습 #2





Python



INDEX

part 1. / Python 소개 및 특징


part 2. / Python 설치 방법

part 3. / Python 문법

part 4. / 간단한 실습

part 5. / 과제

part 6. / 제출 방법 및 질의



Python

part 1. / Python 소개 및 특징

소개

- / 1991년 귀도 반 로섬에 의해 만들어진 인터프리터 언어
- / 파이썬 소프트웨어 재단에서 관리
- / 문법이 매우 쉬어서 초보자들이 처음 프로그래밍을 배울 때 추천되는 언어



단순함

명확성

특징

- / C언어로 구현 (다른 구현 버전도 존재)
- / 다양한 플랫폼에서 사용 가능
- / 라이브러리가 풍부
- / 인터프리터식, 객체지향적, 동적 타이핑
- / 인터프리터 언어처럼 동작하나 소스 코드를 컴파일 하여 바이트 코드를 생성
- / 들여쓰기를 통한 블록 구분

Python

part 2. / Python 설치 방법

/ Linux

- 이미 설치되어 있으며 다음 명령어로 설치 및 버전 확인
\$ python3 --version

/ Ubuntu

- \$ sudo apt-get install python2.7
- \$ sudo apt-get install python 3.9

/ Window

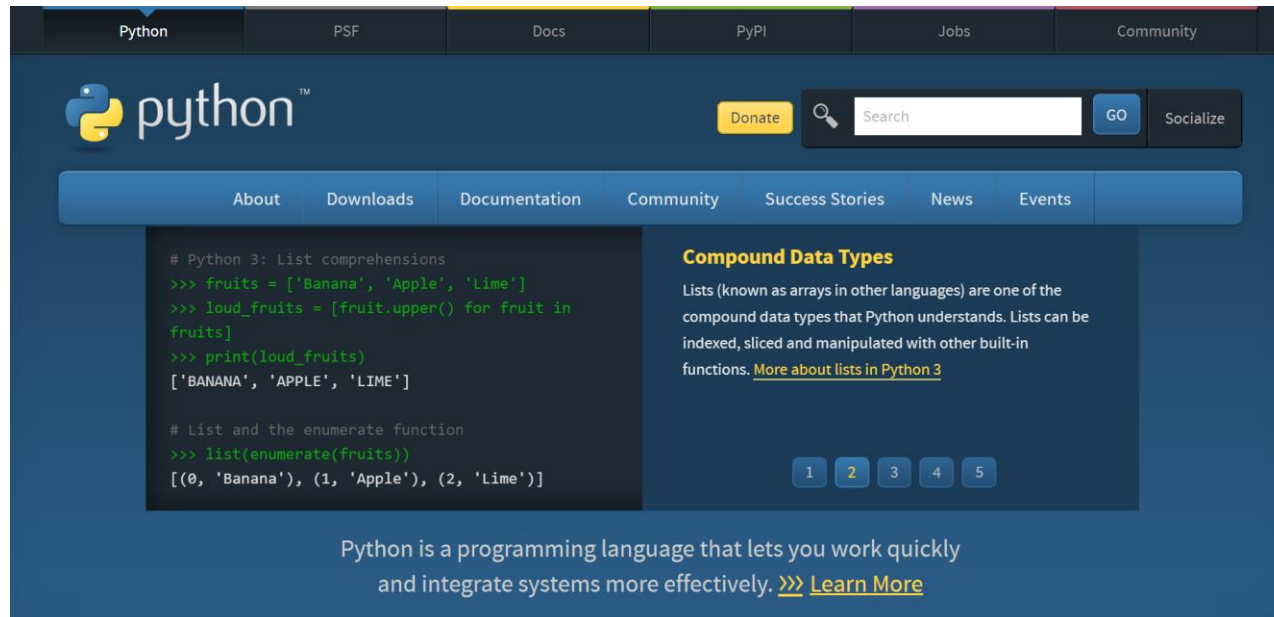
- python을 직접 설치하는 방법
- Anaconda를 통해 설치하는 방법

Python

part 2. / Python 설치 방법

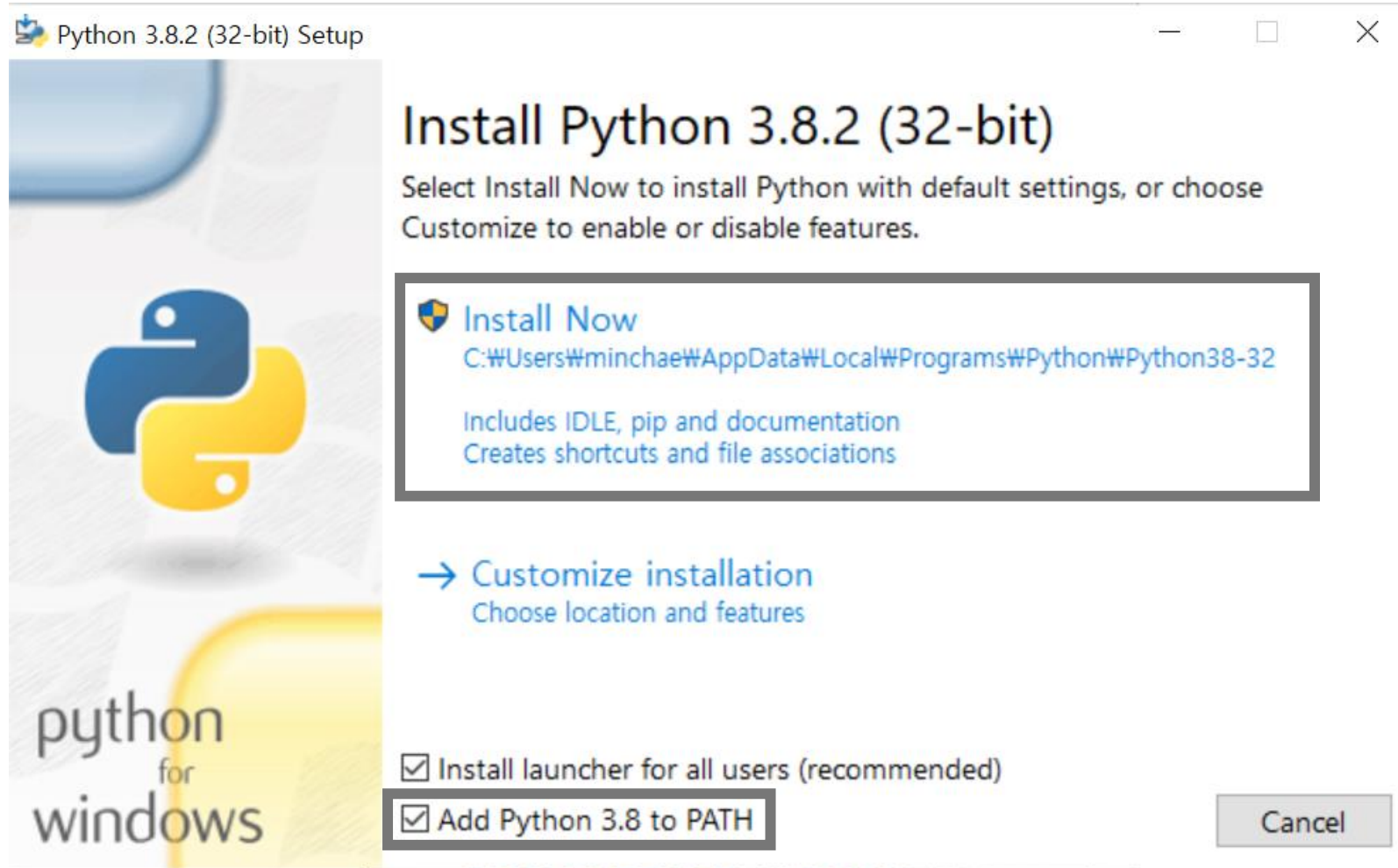
/ python 직접 설치

- python.org 접속
- Python 설치



Python

part 2. / Python 설치 방법

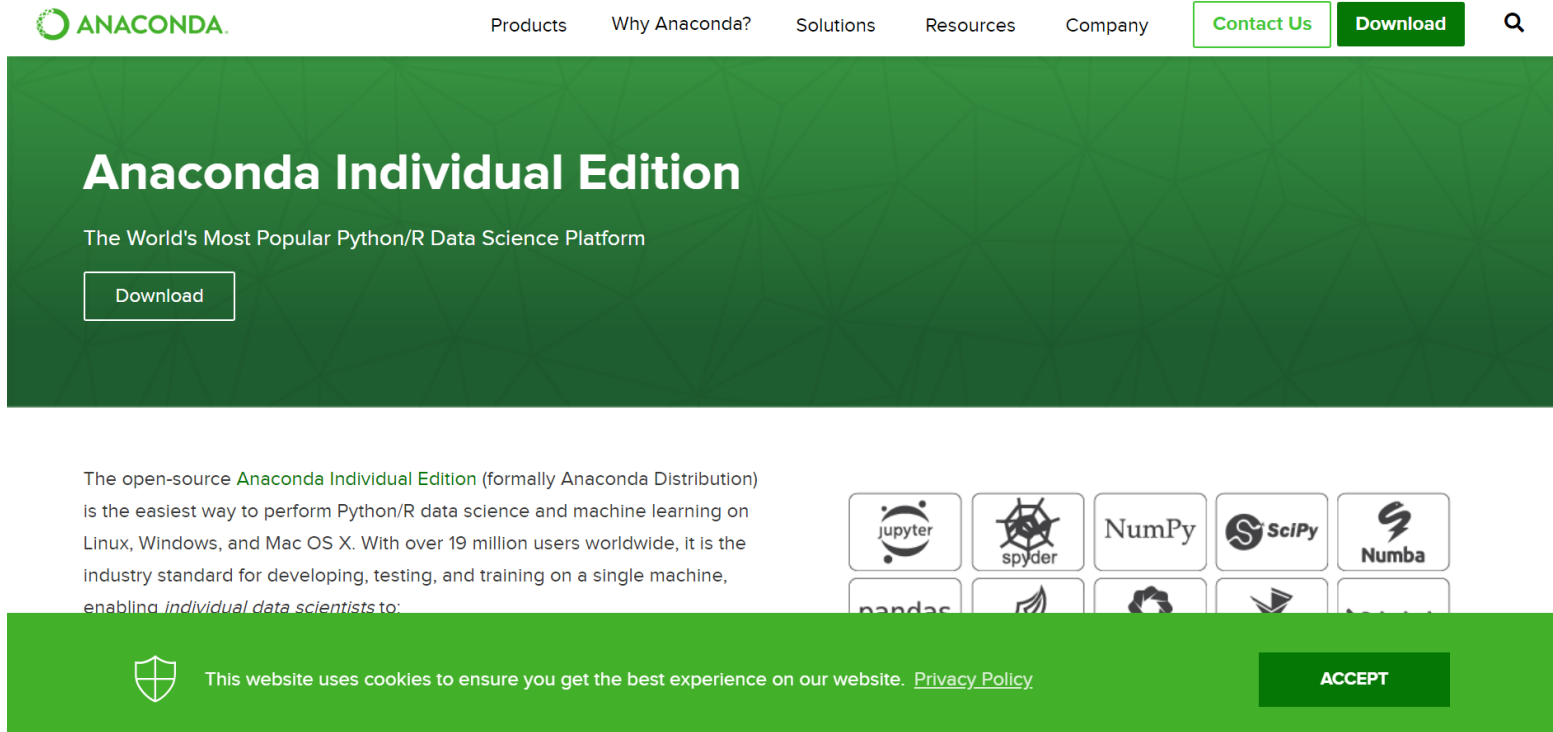



Python

part 2. / Python 설치 방법

/ anaconda 설치

- <https://www.anaconda.com/download/> 접속
- Anaconda 설치













ANACONDA. Products Why Anaconda? Solutions Resources Company [Contact Us](#) [Download](#) 


Anaconda Individual Edition

The World's Most Popular Python/R Data Science Platform

[Download](#)

The open-source [Anaconda Individual Edition](#) (formerly Anaconda Distribution) is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 19 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:



 This website uses cookies to ensure you get the best experience on our website. [Privacy Policy](#) [ACCEPT](#)

Python

part 2. / Python 설치 방법

+ / pycharm

- <https://www.jetbrains.com/pycharm/>



Python

part 3. / Python 문법

/ 변수 및 문자열 선언

- 자료형을 따로 정해주지 않아도 됨
- 형 변환, 자료형은 '자료형()' 함수를 이용해 변경할 수 있음

```
: a=123  
print(type(a))  
  
<class 'int'>
```

```
: b='123'  
c="123" #string을 표시할 때 ', "" 둘 중 무엇을 사용해도 상관 없음  
print(type(b))  
print(type(c))  
  
<class 'str'>  
<class 'str'>
```

```
: a=str(a) #string으로의 형변환  
print(type(a))  
  
<class 'str'>
```

```
: b=int(b) #int로의 형변환  
print(type(b))  
  
<class 'int'>
```

Python

part 3. / Python 문법

/ 연산

사칙 연산은 기타 다른 언어와 동일

```
a=10
b=3

print(a+b) #덧셈
print(a-b) #뺄셈
print(a*b) #곱셈
print(a/b) #나눗셈
print(a//b) #몫
print(a%b) #나머지
print(a**b) #제곱
```

```
13
7
30
3.3333333333333335
3
1
1000
```

/ 입출력

```
a=input()
b=input('입력 : ')
```

```
a
입력 : 44
```

```
print(type(a))
print(type(b))
```

```
<class 'str'>
<class 'str'>
```

```
print("python")
print("python",'\\n')
print('python','!!') #print(출력할 내용,끝내용)
```

```
python
python
```

```
python !!
```

```
a='python'
b='practice'
```

```
print(a,b)
print(a+b)
print(a,",",b) #여러값을 한 줄에 출력하고 싶을 때
```

```
python practice
pythonpractice
python , practice
```

Python

part 3. / Python 문법

/ 튜플, 리스트

배열 대신에 존재하는 자료형

- 리스트 : 선언 후 수정 가능
- 튜플 : 선언 후 수정 불가능

```
a=[]  
b=[1,2,3,4] #리스트  
c=()  
d=(1,2,3,4) #튜플
```

```
b[0]=10  
print(b)
```

```
[10, 2, 3, 4]
```

```
d[0]=10  
print(d)
```


TypeError

Traceback (most recent call

last)

<ipython-input-33-a67b9c2a0360> in <module>

```
----> 1 d[0]=10  
      2 print(d)
```

TypeError: 'tuple' object does not support item assignment

Python

part 3. / Python 문법

```
: a=[1,2,3,4]
a.insert(2,10) #insert(원하는 위치의 인덱스, 추가할 값)
print(a)
[1, 2, 10, 3, 4]
```

```
: a.append(9) #list의 맨 뒤에 값이 추가됨
print(a)
[1, 2, 10, 3, 4, 9]
```

```
: a.pop() #맨 뒤의 값을 return하고 제거함
print(a)
[1, 2, 10, 3, 4]
```

```
: a.pop(0) #맨 앞의 값을 return하고 제거함
print(a)
[2, 10, 3, 4]
```

```
: a=[1,2,3]
b=[4,5,6]
c=a+b #list 합치기
print(c)
[1, 2, 3, 4, 5, 6]
```

```
a=[1,2,3,4]
del a[2] #인덱스 2의 값을 제거
print(a)
a.remove(2) #임력받은 값을 제거
print(a)
[1, 2, 4]
[1, 4]
```

```
print(a.index(4)) #원하는 값의 인덱스 값을 확인
1
```

```
print(1 in a) #값이 해당 리스트에 존재하는지 확인
print(2 in a)
True
False
```

```
len(a) #list의 길이
2
```

Python

part 3. / Python 문법

/ 리스트 slicing

```
a=[1,2,3,4,5,6,7,8,9]
```

```
b=a[0:] #인덱스 0으로부터 시작하는 list return  
print(b)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
c=a[1:4] #인덱스 1부터 4까지의 list return  
print(c)
```

```
[2, 3, 4]
```

```
d=a[::2] #2칸씩 뛰어 얻은 값을 return  
print(d)
```

```
[1, 3, 5, 7, 9]
```

```
e=a[::-1] #reverse  
print(e)
```

```
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

/ 문자열 - 인덱싱 가능

```
print("Hello"+"World") #string의 연산이 가능함
```

```
HelloWorld
```

```
len("Hello World") #string의 길이 구하기
```

```
11
```

```
lang="python"  
print(f'{lang} is simple language') #formatting
```

```
test='{} is simple language'.format(lang)  
print(test)
```

```
python is simple language  
python is simple language
```

```
print(lang[0])  
print(lang[5]) #string indexing
```

```
p  
n
```

Python

part 3. / Python 문법

/ 집합(set)

- 중복을 허용하지 않음
- 순서가 없음 = 인덱싱 불가

```
s=set([1,1,2,3]) #빈 집합은 s=set()으로 선언
print(s)
{1, 2, 3}
```

```
s=set("python") #문자열도 넣을 수 있음
print(s)
{'n', 't', 'y', 'o', 'p', 'h'}
```

```
s=set([1,2,3])
s.add(4) #집합에 값 추가
print(s)
s.update([5,6,7]) #한번에 여러 값 추가
print(s)
s.remove(1) #특정 값 제거
print(s)
{1, 2, 3, 4}
{1, 2, 3, 4, 5, 6, 7}
{2, 3, 4, 5, 6, 7}
```

```
s1=set([1,2,3])
s2=set([2,3,4])

print(s1&s2)
print(s1.intersection(s2)) #교집합

{2, 3}
{2, 3}
```

```
print(s1|s2)
print(s1.union(s2)) #합집합

{1, 2, 3, 4}
{1, 2, 3, 4}
```

```
print(s1-s2)
print(s1.difference(s2)) #차집합

{1}
{1}
```

Python

part 3. / Python 문법

/ dictionary

- Key : value의 쌍으로 이루어져 있음

```
dic={'name':'hongik','age':10} #빈 dictionary는 dic={}로 선언  
#key:value가 ,로 구분되어 {}안에 존재하는 형태
```

```
dic['age']=20 #value 값 변경  
print(dic['age'])
```

20

```
dic['phone number']='010-1111-2222' #key-value 쌍 추가  
print(dic)
```

```
{'name': 'hongik', 'age': 20, 'phone number': '010-1111-2222'}
```

```
del dic['phone number'] #값 제거  
print(dic)
```

```
{'name': 'hongik', 'age': 20}
```

```
print(dic.get('name'))  
print(dic['name']) #원하는 key 값에 접근하기
```

hongik
hongik

```
print(dic.get('phone number'))  
print(dic.get('phone number','010-1111-2222'))
```

None
010-1111-2222

```
test={1:'a',1:'A'} #key 값은 중복 불가  
print(test)
```

{1: 'A'}

```
k=dic.keys() #key 리스트 만들기  
v=dic.values() #value 리스트 만들기  
print(list(k))  
print(list(v))
```

['name', 'age']
['hongik', 20]

```
print('name' in dic)  
print('univ' in dic) #key 값이 존재하는지 확인
```

True
False

Python

part 3. / Python 문법

/ 조건문
- if, elif, else

```
a=input()
```

100

```
if a=="100":  
    print("백") #들여쓰기로 해당 조건일때 수행하는 것임을 표현  
elif a=="200":  
    print("이백")  
else:  
    print("not exist")
```

백

```
univ=input("univ : ")  
loc=input("location : ")
```

```
if univ=='hongik' and loc=='seoul': #&&가 아닌 and  
    print('correct')  
elif univ=='hongik' or loc=='seoul': #||가 아닌 or  
    print('check univ or location again')  
else:  
    print('check your input')
```

```
univ : hongik  
location : seoul  
correct
```

```
univ : hongik  
location : suwon  
check univ or location again
```


Python

part 3. / Python 문법

/ 반복문

- for

```
for i in range(4): #range를 이용하여 반복 범위 지정  
    print(i)
```

0
1
2
3

```
for i in range(1,10,2):  #(시작값, 마지막값, jump)  
    print(i)
```

1
3
5
7
9

```
a=[1,2,3,4]
```

```
for i in a: #list를 반복문에 사용할 수 있음  
    print(i)
```

1
2
3
4

```
for i,e in enumerate(a): #index값과 요소값을 한번에 받음  
    print("index : " + str(i)+" element : "+str(e))
```

index : 0 element : 1
index : 1 element : 2
index : 2 element : 3
index : 3 element : 4

Python

part 3. / Python 문법

/ 반복문

- while

```
i=0
while i<=10:
    print(i)
    i+=1
```

0
1
2
3
4
5
6
7
8
9
10

```
i=0
while True:
    print(i)
    i+=1
    if(i==10):
        break
```

0
1
2
3
4
5
6
7
8
9

```
a=[1,2,3,1,1,4]
while 1 in a:
    a.remove(1)
```

```
print(a)
```

[2, 3, 4]

Python

part 3. / Python 문법

/ 함수

- 반환 값이 있는 함수

```
def print_return_insert(a):  
    print(a)  
    return(a)
```

```
a="hello world"  
b=print_return_insert(a)  
print(b)
```

hello world
hello world

```
def swap(a,b):  
    return b,a
```

```
a=1  
b=2  
a,b=swap(a,b)  
print(a,b)
```

2 1

- 반환 값이 없는 함수

```
def print_insert(a):  
    print(a)
```

```
a="hello world"  
print_insert(a)
```

hello world

- Default 값 설정 함수

```
def default_exist(a=1,b=2):  
    print(a,b)
```

default_exist()

default_exist(3)

1 2
3 2

- 입력 값의 수가 정해지지 않을 때

```
#입력 받은 여러 개의 값을 tuple로 받음  
def several_inputs(*num):  
    print(num)
```

```
several_inputs(1,2,3,4)
```

(1, 2, 3, 4)

```
#입력 받은 여러 개의 값을 dictionary로 받음  
def several_strings(**info):  
    print(info)
```

```
several_strings(name='cho',age='10')
```

{'name': 'cho', 'age': '10'}

Python

part 3. / Python 문법

/ 클래스

- 생성 시 입력 값을 받지 않는 경우

```
: class Clac:
    def __init__(self): #생성자
        self.a=10
        self.b=3
    def plus(self):
        return self.a+self.b
    def plus_with_input(self,new_int):
        return self.a+new_int
```

```
: c=Clac()
   print(c.plus())
```

13

```
: print(c.plus_with_input(5))
```

15

- 생성 시 입력 값을 받는 경우

```
class Student:
    univ='hongik' #모든 객체가 공통적으로 갖게되는 값
    def __init__(self,name,age):
        self.name=name
        self.age=age
    def get_info(self):
        return "univ : "+self.univ+" ,name : "+self.name+" ,age : "+str(self.age)
```

```
s1=Student('cho','10')
print(s1.get_info())
```

univ : hongik ,name : cho ,age : 10

Python

part 4. / 간단한 실습

/ 최대공약수를 구하는 함수를 포함하는 Math 클래스를 생성하라

1. 최대공약수를 구하는 함수 $\text{gcd}(m,n)$
2. $n = 0$ 이면 $\text{gcd}(m,n)$
3. $m \bmod n = 0$ 이라면, $\text{gcd}(m,n) = n$
4. $m \bmod n \neq 0$ 이면, $\text{gcd}(m,n) = \text{gcd}(n, m \bmod n)$

Python

part 4. / 간단한 실습

```
class Math:
    def gcd(self,m,n):
        if m<n:
            m,n=n,m
        if n==0:
            return m
        if (m%n)==0:
            return n
        else:
            return self.gcd(n,m%n)
```

```
if __name__=='__main__': #main 함수
    math=Math()
    print(math.gcd(4,10))
```

Python

part 5. / 과제

과제 1

/ 리스트에서 숫자 n이 몇 번째 순서에 위치하여 있는지를 Binary Search로 찾아 출력하는 함수를 구현하라

- 입력한 숫자가 없을 경우, None을 출력
- 입력은 input()을 통해 받을 것
- 입력 값의 첫 줄은 list의 요소가, 두번째 줄은 n의 값이다

Input : 1 11 15 19 37 40 59 61

15

Output : 3

Input : 1 11 15 19 32 48 59 65

58

Output : None

Python

part 5. / 과제

과제 2

/ 리스트를 Quick Sort로 정렬하라

- 입력은 input()을 통해 받을 것
- 입력은 리스트의 요소들이다

Input : 26 5 37 1 61 11 59 15 48 19

Output : [1, 5, 11, 15, 19, 26, 37, 48, 59, 61]

Python

part 5. / 과제

과제 3

/ 리스트를 Merge Sort로 정렬하라

- 입력은 input()을 통해 받을 것
- 입력은 리스트의 요소들이다

Input : 100 23 31 123 435 642 1

Output : [1, 23, 31, 100, 123, 435, 642]

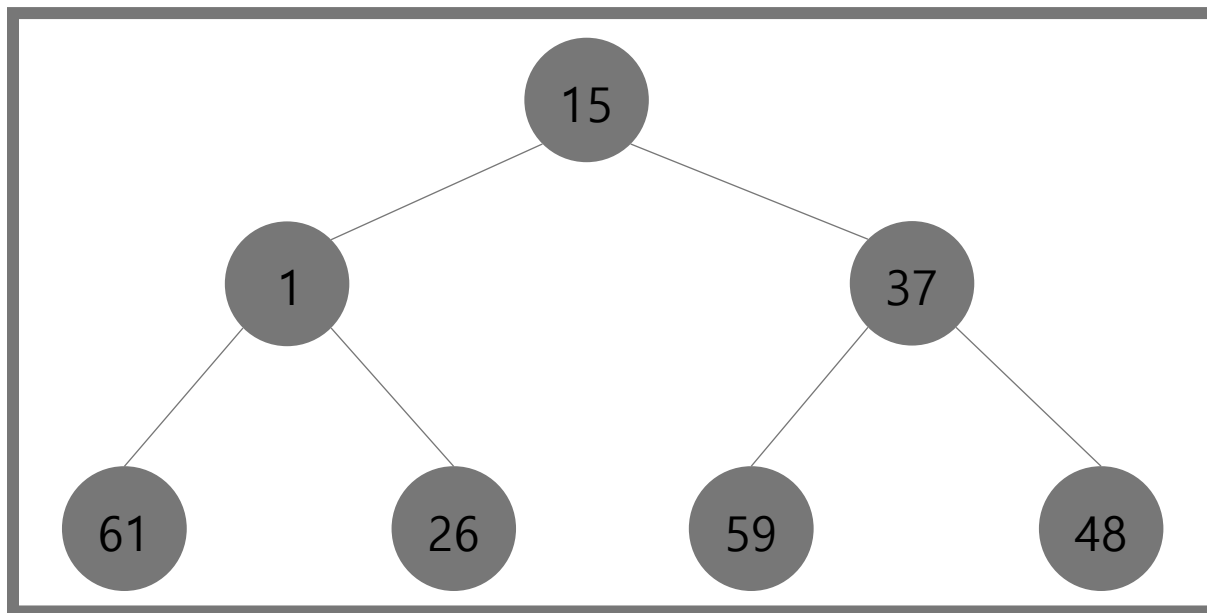
Python

part 5. / 과제

과제 4

/ 아래 그림과 같은 이진 트리를 생성하고 전위 순회, 중위 순회, 후위 순회 방식으로 각 방문 노드를 방문하고 방문 순서를 출력하라.

- 클래스로 구현할 것
- 전위 순회, 중위 순회, 후위 순회는 함수를 각각 따로 만들 것
- Input : null



Output : Preorder Traverse

15

1

61

26

37

59

48

Inorder Traverse

61

1

26

15

59

37

48

Postorder Traverse

61

26

1

59

48

37

15

Python

part 5. / 과제

과제 5

- / 한 개의 강의실이 있고 이를 사용하고자 하는 N개의 수업이 존재한다
각 강의들마다 시작 시간과 종료 시간은 다르므로, 강의실에 강의들이 겹치지 않고 배치되는 강의의 수가
최대가 되도록 하는 배치 방법을 찾아라
- input의 첫번째 줄은 강의의 개수
두번째 줄부터는 강의 번호, 강의 시간, 종료 시간으로 이루어져 있다
 - output은 최대 배치 강의 수
배치되는 강의의 리스트 로 출력한다
 - 입력을 input()을 통해 받을 것

Input : 11

1 3 8
2 5 9
3 3 5
4 1 4
5 8 12
6 0 6
7 8 11
8 2 13
9 4 7
10 6 10
11 12 14

Output 4

[4,9,7,11]

Python

part 5. / 과제

과제 6

/ 홍익대학교 리눅스 서버 접속 시 데이터는 0,1 두 비트로 변경되어 전송된다
서버의 안전을 위해 수신된 데이터는 암호화가 되어있는지 확인이 필요하다
지환이는 데이터를 분석해 암호화 되지 않은 데이터는 일정한 패턴을 가지고 있다는 것을 알게 되었다.
서버의 안전을 위해 지환이는 암호화 되지 않은 데이터를 확인해 차단하고자 한다
프로그래밍을 잘 하지 못하는 지환이를 대신해 암호화 되지 않은 데이터를 차단하는 프로그램을 만들어주자

이때 ~ 기호의 의미는 다음과 같다

- $X\sim$ 는 X 가 한 번 이상 반복되는 모든 경우의 수의 집합
- $(xyz)\sim$ 는 xyz 가 한 번 이상 반복되는 모든 경우의 수의 집합

ex) $1\sim = \{1, 11, 111, 1111, \dots, \dots\}$

$10\sim 11 = \{1011, 10011, 100011, 1000\dots 11, \dots\}$

또한 | 기호의 의미는 다음과 같다

- $(x|y)$ 는 x 또는 y 중 아무거나 하나만을 선택해서 만든 경우의 수의 집합을 의미한다

ex) $(1001|0101) = \{1001, 0101\}$

만약 $(x|y)\sim$ 가 될 경우는 x 와 y 를 마음대로 섞어서 만들 수 있는 모든 경우의 수의 집합을 의미한다

ex) $(100|11)\sim = \{100, 11, 10011, 11100, 1110011, \dots\}$

Python

part 5. / 과제

- Input의 첫번째 줄은 문자열의 개수 n
두번째 줄부터는 0과 1로 이루어진 문자열이 주어진다
- Output은 패턴과 같은 데이터이면 DANGER, 아닐 경우는 PASS를 출력
- 입력은 input()을 통해 받을 것
- 주어진 패턴 : (100~1~|01)~

Input : 2

1000101

10101

Output : DANGER

PASS

Python

part 5. / 과제

과제 7

/ 과제 1 ~ 6을 하면서 구현한 사항을 **Latex**으로 작성해서 제출

- 구현한 사항에 대해서 자유롭게 적으면 됨
- 코드를 첨부할 필요는 없으나, 과제에 대한 설명 중 코드가 필요할 경우 자유롭게 첨부
- 과제마다 Section을 나누어서 깔끔하게 제출

Python

part 6. / 제출 방법 및 질의

/ 제출 마감일 : 4월 8일 목요일 23시 59분까지 (1,2분반)(기한 : 2주)
4월 9일 금요일 23시 59분까지 (3,4분반)(기한 : 2주)

/ 제출 파일 : hw1_1~1_6.py, hw1.tex, hw1.pdf

/ 제출 방법 : submit pem_ta hw1**a** (1분반)
submit pem_ta hw1**b** (2분반)
submit pem_ta hw1**c** (3분반)
submit pem_ta hw1**d** (4분반)

/ 제출 확인 : submit pem_ta hw1**x** -l

/ 질문 , 조교 메일 : pemta806@gmail.com

- 질문하기 전 ppt, pdf를 한번 더 확인해주시면 감사하겠습니다

/ 제출 유의 사항

- 과제 예시로 달아 놓은 output과 동일한 포맷으로 출력 해야 합니다
- 서버가 1시간전에 다운되지 않는 이상 Late 제출, 메일 제출은 받지 않습니다
- Cheating 시 무조건 F (조교들이 하나 하나 모든 파일 검사)