



ENGINEERING

Engineering is Awesome!

Python in Minecraft

Circuit in Minecraft (Basic)

Circuit in Minecraft (Advance)

Build #ISH530
Made By Chicken-Jocky



1

PYTHON IN MINECRAFT



WHAT IS PROGRAMMING

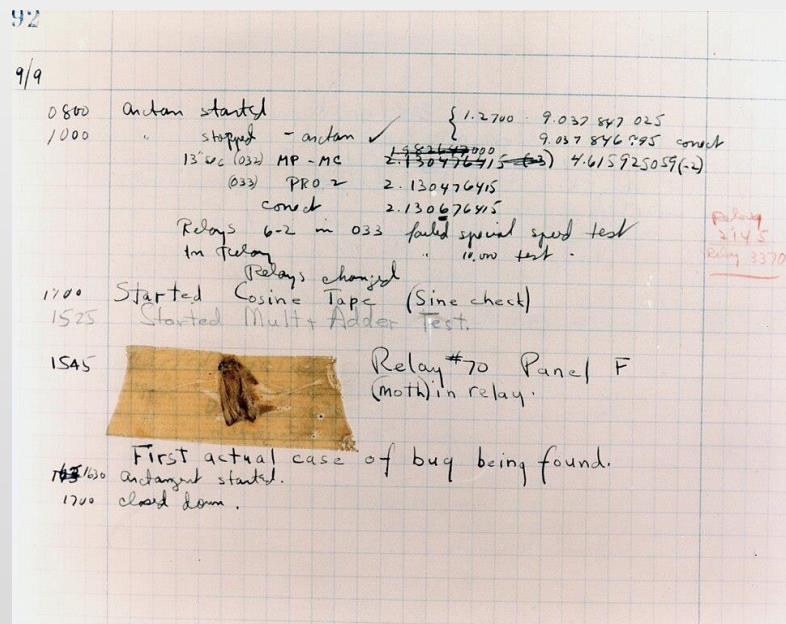


프로그램이란?

문제를 해결하거나 목표를 달성하기 위한 순차적인 명령

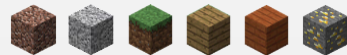
디버깅이란?

버그란 프로그램에서 잘못된 부분을 뜻하는데,
이를 고치는 것을 디버깅이라 부른다





WHAT IS PROGRAMMING



왜 프로그래밍을 배워야 하는가?

사회가 많이 변하여, 이제 거의 모든 곳에서 프로그래밍을 하고 있다.





WHAT IS PROGRAMMING



프로그래밍을 효율적으로 하는 방법

- 간단하게 시작한다.
- 한 번에 하나의 작은 작업만 수행한다.
- 각각의 작업들이 다른 작업에 영향을 주지 않도록 한다.
- 알기 쉬운 유용한 주석을 사용한다.
- 의미를 잘 전달할 수 있는 식별자를 사용한다.



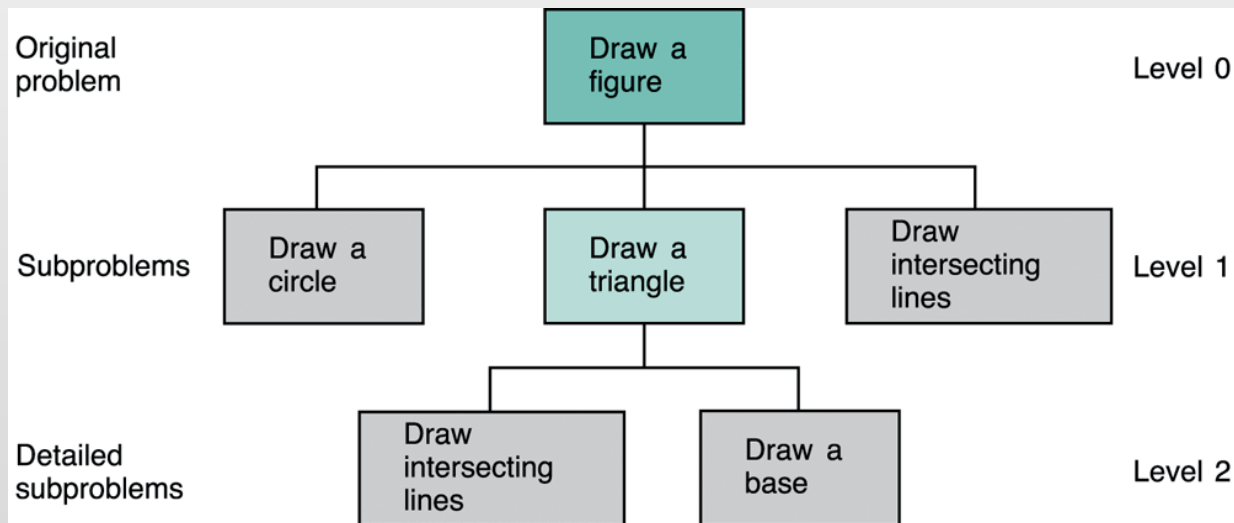


WHAT IS PROGRAMMING



하향식 설계 (Top-Down Design)

하나의 큰 문제를 쉽게 풀 수 있도록 작은 크기의 여러 문제로 나누고,
그 문제 해결책들을 모아서 큰 문제에 대한 해결책으로 활용하는 설계 방식





WHAT IS PROGRAMMING



객체 지향 프로그래밍 (Object-Oriented Programming)

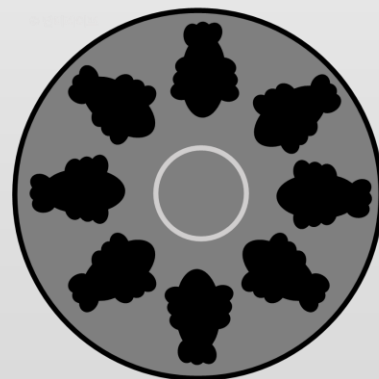
객체 및 이들간의 관계, 상호작용 등을 기반으로 프로그램을 설계하는 방법론

객체(Object)란?

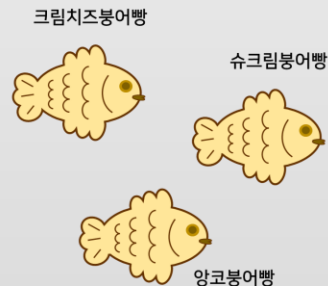
속성과 행동을 묶어 하나의 독립적인 단위로 관리하는 것

클래스(Class)란?

객체를 정의하는 틀



붕어빵 틀은 클래스



붕어빵은 객체





WHAT IS PROGRAMMING



객체 지향의 4대 특성

- 캡슐화: 객체의 속성과 행위를 하나로 묶을 수 있다
- 상속: 상위 클래스의 속성과 행위를 상속받을 수 있다
- 추상화: 객체의 공통적인 속성과 기능을 추출하여 정의해야 한다
- 다형성: 객체의 속성이나 기능이 상황에 따라 여러 형태를 가질 수 있다





HELLO WORLD



사용할 라이브러리

mcpi - Python으로 Minecraft를 제어할 수 있는 라이브러리





HELLO WORLD

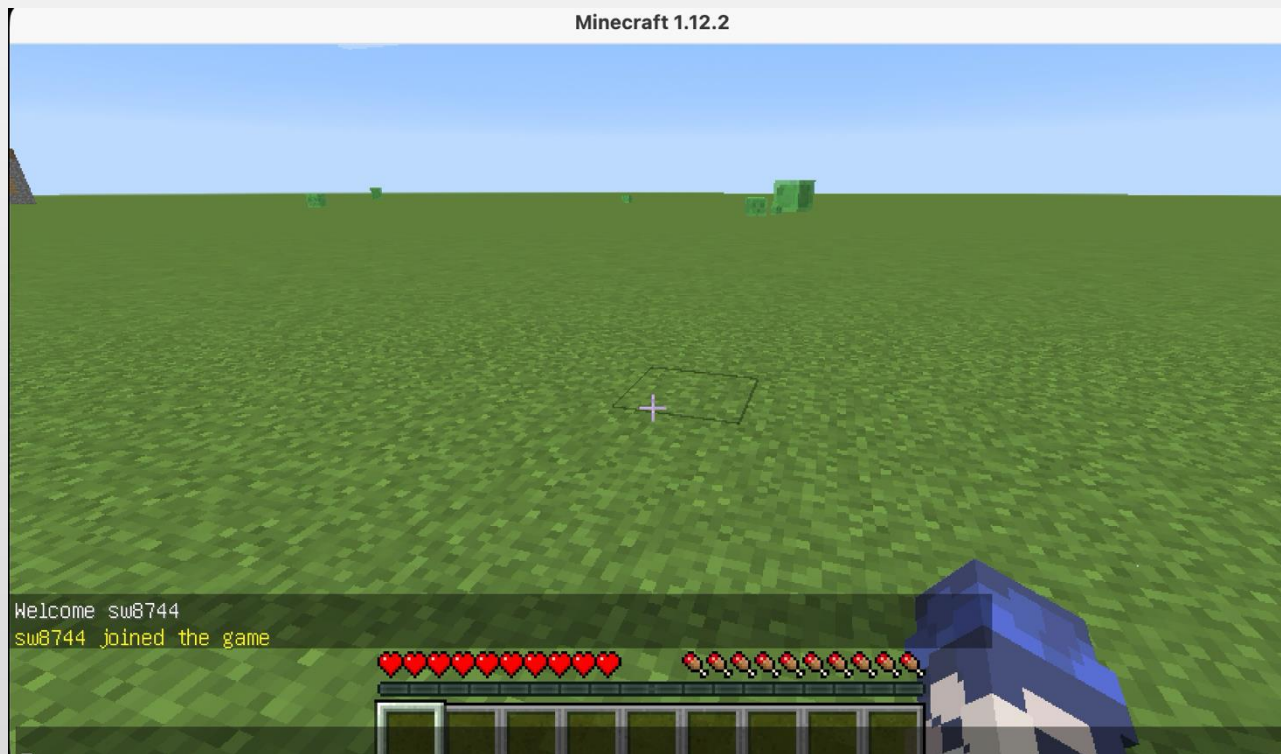


```
from mcpi.minecraft import Minecraft  
world = Minecraft.create()  
  
world.postToChat("Hello, World!")
```





HELLO WORLD





HELLO WORLD



퀴즈: 콘솔 창에서 메시지를 입력받고 출력해 보세요.

힌트: `input()` 함수를 통해 입력받을 수 있어요.

```
minecraft-python — sw8744@iseung-won-ui-MacBookPro — ..ecraft-python — -zsh — 80x25
sw8744@iseung-won-ui-MacBookPro ~ /Desktop/dev/minecraft-python main ↵
python3 01-Quiz.py
```





HELLO WORLD



퀴즈: 콘솔 창에서 메시지를 입력받고 출력해 보세요.

힌트: `input()` 함수를 통해 입력받을 수 있어요.

```
from mcpi.minecraft import Minecraft
world = Minecraft.create()

msg = input("메시지를 입력하세요: ")
world.postToChat(msg)
```





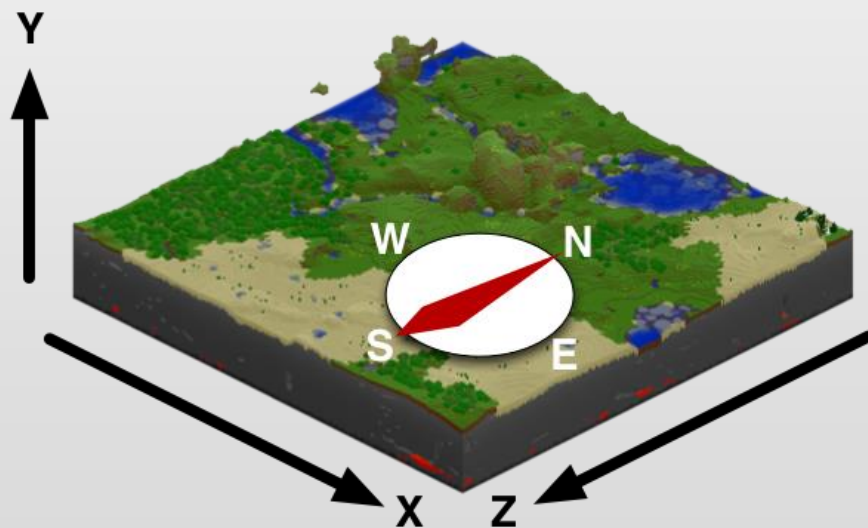
3

PRINT LOCATION



마인크래프트 좌표계

x, y, z 좌표를 가지는 3차원 좌표계





```
from mcpi.minecraft import Minecraft
```

```
world = Minecraft.create()
```

```
while True:
```

```
    pos = world.player.getTilePos()
```

```
    x = str(pos.x)
```

```
    y = str(pos.y)
```

```
    z = str(pos.z)
```

```
    world.postToChat("x: " + x + ", y: " + y + ", z: " + z)
```

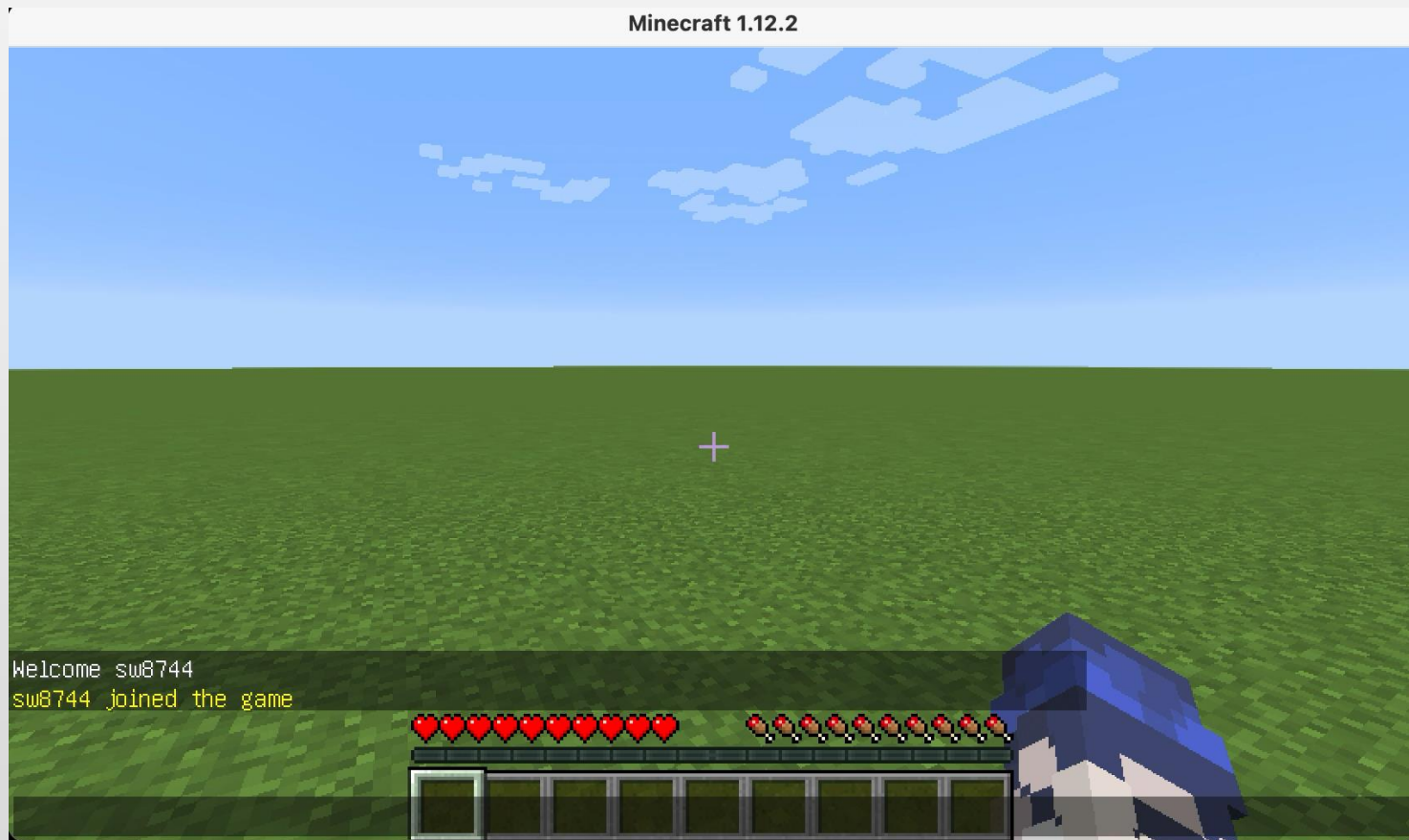




PRINT LOCATION



Minecraft 1.12.2





BUILD A PYRAMID FUNCTION



함수

하나의 특별한 목적의 작업을 수행하기 위해 독립적으로 설계된 코드의 집합

매개 변수

```
def function(a, b):  
    result = a + b  
    return a + b
```

작업 수행

결과 리턴





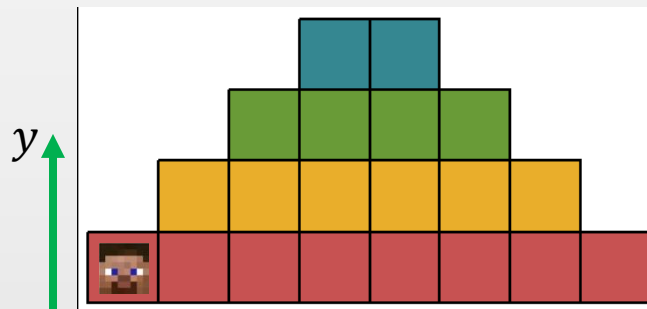
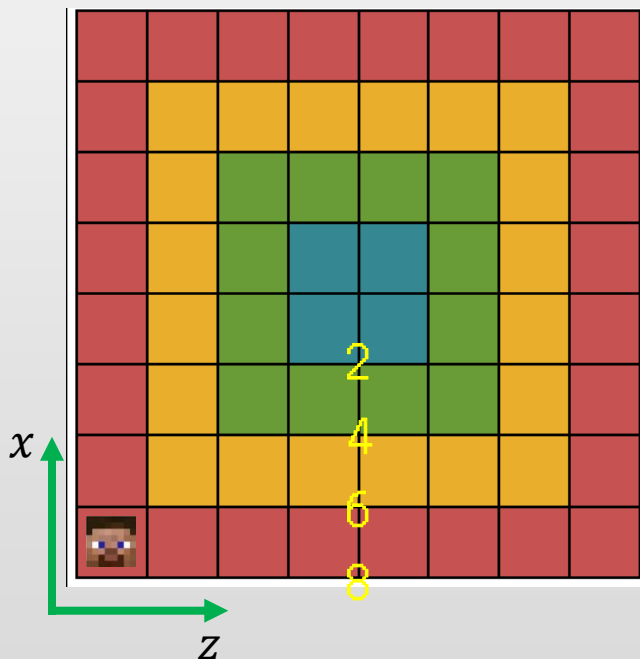
BUILD A PYRAMID FUNCTION



Minecraft 1.12.2



알고리즘 설계



블럭의 개수 n 과 좌표 차이 x 와의 관계

$$\rightarrow x = n - 1$$

전체 층수 f 인 피라미드에서 i 층의 한 변 좌표 차

$$\rightarrow 2(f - (i - 1)) - 1$$

전체 층수 f , 플레이어의 초기 좌표 (x_0, y_0, z_0) 일 때 피라미드에서의

$$i\text{층의 시작 좌표} \rightarrow (x_0 + i, y_0 + i - 1, z_0 + i)$$





BUILD A PYRAMID FUNCTION



```
from mcpi.minecraft import Minecraft
```

```
SAND = 12
```

```
world = Minecraft.create()
```

```
def pyramid(f): 1 usage 🧑 thislife_won *
```

```
    pos = world.player.getTilePos()
```

```
    for i in range(1, f + 1):
```

```
        delta = 2 * (f - (i - 1)) - 1
```

```
        world.setBlocks(pos.x + i - 1, pos.y + i - 1, pos.z + i - 1,  
                        pos.x + i + delta, pos.y + i - 1, pos.z + i + delta,  
                        SAND)
```

```
pyramid(3)
```





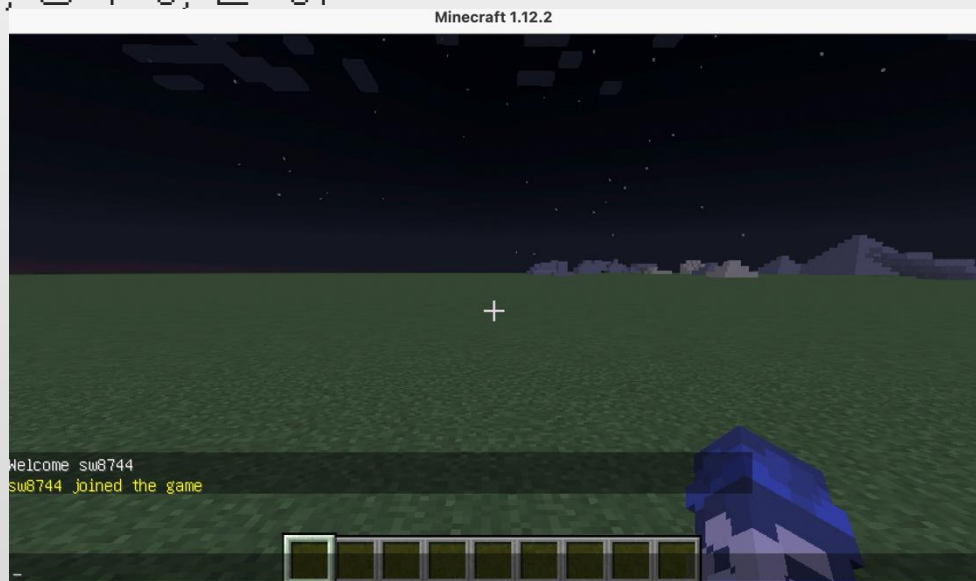
BUILD A PYRAMID FUNCTION



퀴즈: width, height를 입력받아 그에 맞는 활성화된 네더 포탈을 만드는 함수를 만들어 보아요.

힌트: 공기를 채워 구멍을 뚫을 수 있어요. 중간에 불을 설치하면 네더 포탈이 활성화 되어요.

흑요석: 49, 공기: 0, 불: 51





BUILD A PYRAMID FUNCTION



퀴즈: width, height를 입력받아 그에 맞는 활성화된 네더 포탈을 만드는 함수를 만들어 보아요.

힌트: 공기를 채워 구멍을 뚫을 수 있어요. 중간에 불을 설치하면 네더 포탈이 활성화 되어요.

흑요석: 49, 공기: 0, 불: 51

```
from mcpi.minecraft import Minecraft
```

```
OBSIDIAN = 49
```

```
AIR = 0
```

```
FIRE = 51
```

```
world = Minecraft.create()
```

```
def portal(width, height): 1 usage new *
```

```
    pos = world.player.getPos()
```

```
    world.setBlocks(pos.x, pos.y, pos.z,  
                    pos.x + width - 1, pos.y + height - 1, pos.z,  
                    OBSIDIAN)
```

```
world.setBlocks(pos.x + 1, pos.y + 1, pos.z,  
                pos.x + width - 2, pos.y + height - 2, pos.z,  
                AIR)
```

```
world.setBlock(pos.x + 1, pos.y + 1, pos.z, FIRE)
```

```
portal( width: 4, height: 5)
```





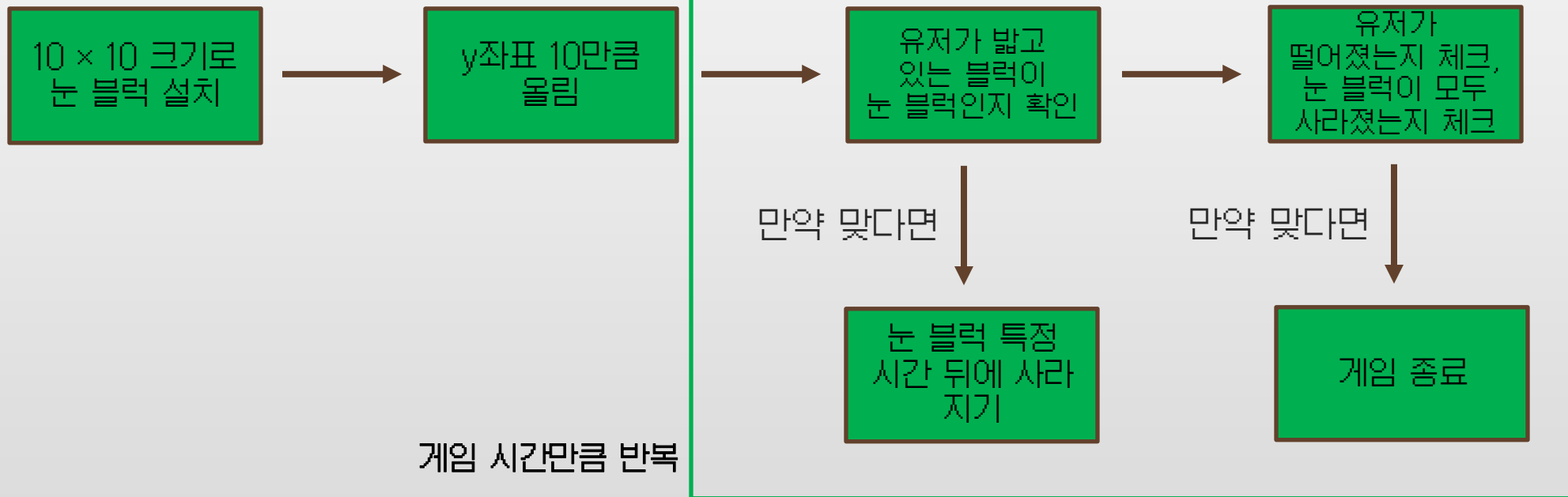
MAKE A MINIGAME



게임 <Fall Guys>에 나오는 '바닥 떨어져유'를 만들어 보아요!



게임 설계





MAKE A MINIGAME



```
from mcpi.minecraft import Minecraft
import time
```

```
SNOW = 80
RANGE = 10
FALLING_TIME = 0.3
```

```
world = Minecraft.create()
START_POS = world.player.getTilePos()
```

```
def fall_block(): 1 usage new *
    pos = world.player.getTilePos()
    if world.getBlock(pos.x, pos.y - 1, pos.z) == SNOW:
        time.sleep(FALLING_TIME)
        world.setBlock(pos.x, pos.y - 1, pos.z, 0)
```

```
def check_block(): 1 usage new *
    for i in range(0, RANGE + 1):
        for j in range(0, RANGE + 1):
            if world.getBlock(START_POS.x + i, START_POS.y + 10, START_POS.z + j) != SNOW:
                return False
    return True
```

```
def check_y(): 1 usage new *
```

```
    pos = world.player.getTilePos()
    if pos.y < START_POS.y + 10:
        return False
    return True
```

```
world.postToChat("게임을 시작합니다.")
world.player.setTilePos(START_POS.x, START_POS.y + 10, START_POS.z)
pos = world.player.getTilePos()
world.setBlocks(pos.x, pos.y - 1, pos.z,
                pos.x + RANGE, pos.y - 1, pos.z + RANGE,
                SNOW)
world.setBlock(pos.x, pos.y - 1, pos.z, 1)
```

```
while True:
    fall_block()
    if check_block():
        world.postToChat("게임을 종료합니다.")
        break
    if not check_y():
        world.postToChat("떨어졌습니다. 게임을 종료합니다.")
        world.setBlocks(START_POS.x, START_POS.y + 9, START_POS.z,
                        START_POS.x + RANGE, START_POS.y + 9, START_POS.z + RANGE,
                        0)
        break
```





MAKE A MINIGAME



Minecraft 1.12.2





2

CIRCUIT IN MINECRAFT BASIC



REDSTONE CIRCUIT



레드스톤 회로

레드스톤과 관련 블록을 이용하여
신호를 생성, 전달, 조절, 출력하는 시스템



현실의 전기를 통한 회로 시스템을
마인크래프트에서 신호를 통한 레드스톤 회로로 구현





REDSTONE CIRCUIT



신호 전달

레드스톤 가루

최대 15칸까지 신호 전달이 가능하며,
처음 신호로부터 멀어질수록 신호의 세기가 약해진다



실제 회로에서의 전선에 대응된다





REDSTONE CIRCUIT

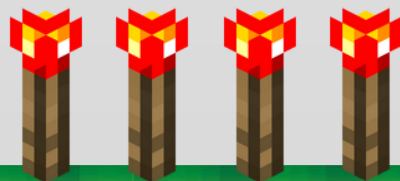


신호 전달

레드스톤 횃불

신호 15를 발생

다른 블록에서 신호를 받으면 3틱 후 횃불이 꺼지며,
횃불로부터 신호를 받던 다른 블록들에게도 신호 차단



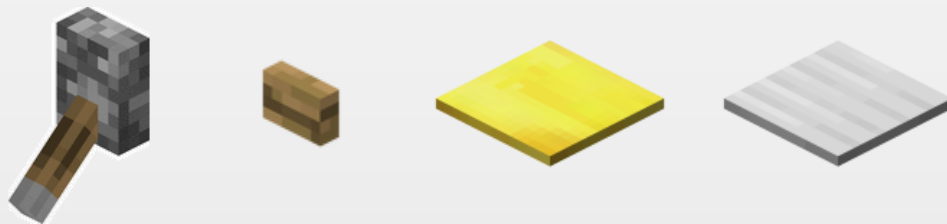


REDSTONE CIRCUIT



신호 발생

레버, 버튼, 압력판, 관측기 등



실제 회로에서의 버튼, 조도 센서, 초음파 센서 등의
여러 측정 센서와 대응된다.





REDSTONE CIRCUIT



신호 발생 - 감지 블록

무게 압력판 : 올려져 있는 개체의 개수에 따라 신호세기 달라짐

관측기 : 앞면 쪽 블록에 업데이트가 생긴다면 반대면에 15의
세기로 신호가 1틱 동안 발생

태양 감지기 : 태양의 높이, 날씨에 따라 신호세기 달라짐

신호 강도	경형	중형
[펼치기·접기]		
1	1	1 ~ 10
2	2	11 ~ 20
3	3	21 ~ 30
4	4	31 ~ 40
5	5	41 ~ 50
6	6	51 ~ 60
7	7	61 ~ 70
8	8	71 ~ 80
9	9	81 ~ 90
10	10	91 ~ 100
11	11	101 ~ 110
12	12	111 ~ 120
13	13	121 ~ 130
14	14	131 ~ 140
15	15 이상	141 이상





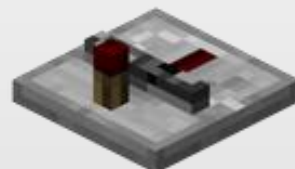
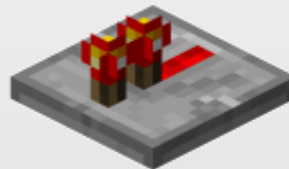
REDSTONE CIRCUIT



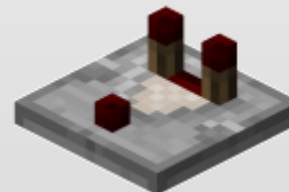
신호 조절

레드스톤 중계기, 비교기

중계기 : 레드스톤 신호 증폭 및 지연, 최소 0.1초(2틱)
최대 0.4초(8틱)까지 신호 지연 가능



비교기 : 비교 모드 - 레드스톤 신호 비교 (옆, 뒤)
감산 모드 - 뒤 신호에서 옆 신호를 빼서 출력



단방향성을 가진다.





REDSTONE CIRCUIT



신호 출력

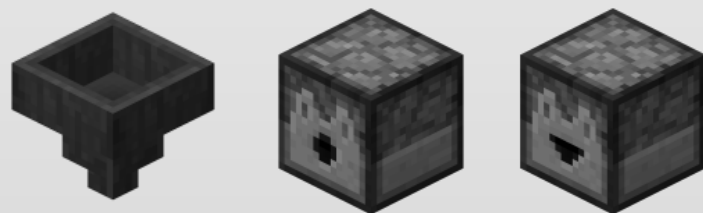
피스톤, 호퍼, 발사기, 공급기, 레드스톤 조명 등

(끈끈이) 피스톤 : 최대 12개 블록까지 움직일 수 있다



호퍼 : 아이템 흡수 및 공급

레드스톤 신호를 받으면 비활성화된다.

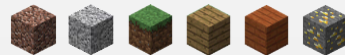


실제 회로에서의 모터, LED 조명 등과 대응된다.





DOOR LOCK



도어락 실습

상황 : 4개의 레버를 활용하여 비밀번호를 만들고

비밀번호가 맞으면 문이 열리게 된다. (사진에서 비번 -> 3, 4)





DOOR LOCK



도어락 실습

상황 : 4개의 레버를 활용하여 비밀번호를 만들고

비밀번호가 맞으면 문이 열리게 된다. (사진에서 비번 -> 3, 4)

Hint : 1. 레드스톤 횃불은 신호를 받은 블록에 붙어있으면 꺼지고,
신호를 못 받으면 켜진다.

2. 끈끈이 피스톤 사용 - 레드스톤 가루로 이어져 있는 회로를 대각선에서
블록으로 막으면 신호가 끊어짐

3. 맞는 번호 " 만 " 입력했을 때 레드스톤 회로가 작동되고, 틀린 번호를
하나라도 입력하면 끈끈이 피스톤이 작동되어 회로를 막게끔 작동





DOOR LOCK



도어락 실습

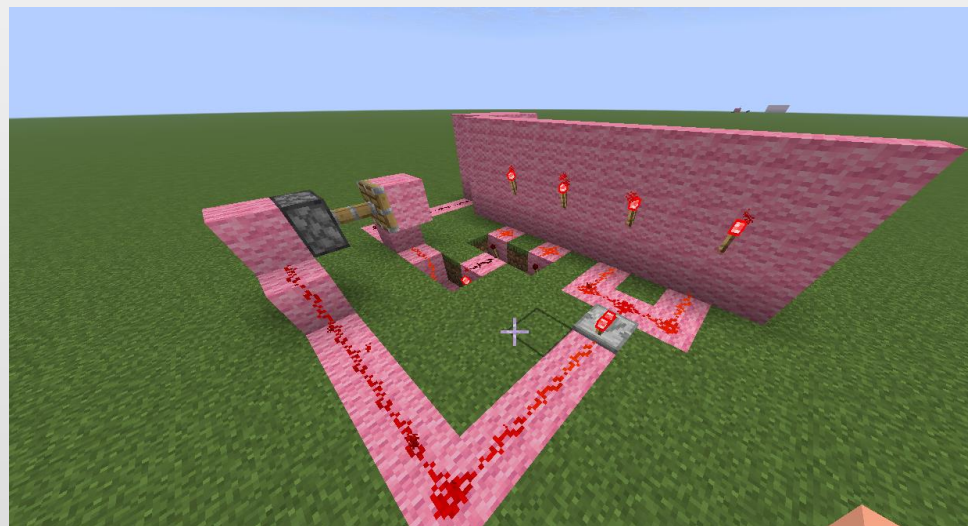
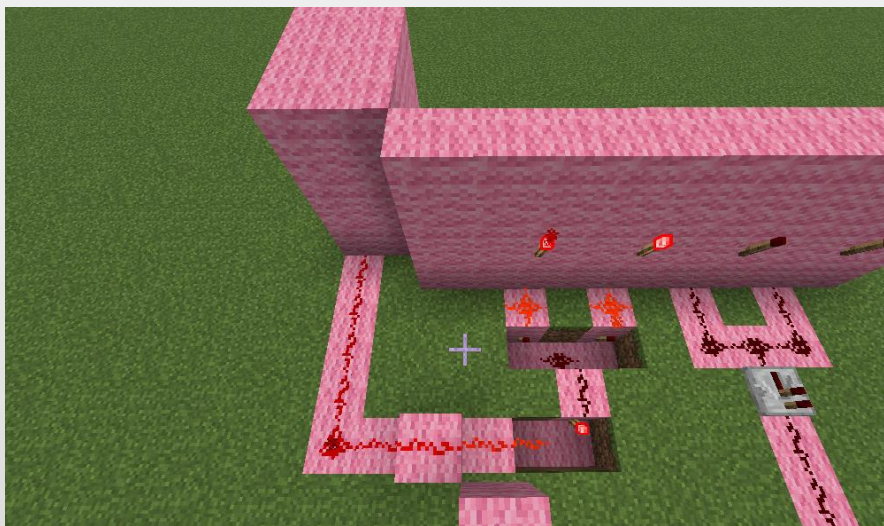




DOOR LOCK



답 / 안 되면 나 부르기



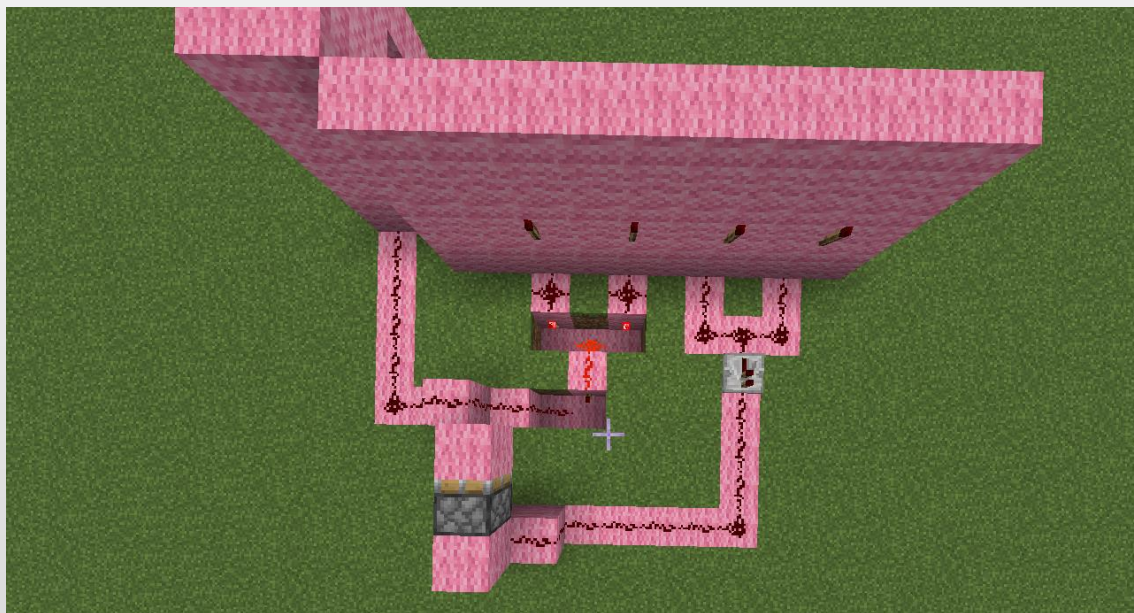


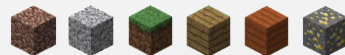
2

DOOR LOCK











답 / 안 되면 나 부르기





논리 게이트란?

디지털 회로에서 0과 1의 입력 값을 받아 특정한 규칙에 따라 출력하는 회로

Buffer  <table><tr><th>A</th><th>Z</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	A	Z	0	0	1	1	AND  <table><tr><th>A</th><th>B</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Z	0	0	0	0	1	0	1	0	0	1	1	1	OR  <table><tr><th>A</th><th>B</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Z	0	0	0	0	1	1	1	0	1	1	1	1	XOR  <table><tr><th>A</th><th>B</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Z	0	0	0	0	1	1	1	0	1	1	1	0
A	Z																																																					
0	0																																																					
1	1																																																					
A	B	Z																																																				
0	0	0																																																				
0	1	0																																																				
1	0	0																																																				
1	1	1																																																				
A	B	Z																																																				
0	0	0																																																				
0	1	1																																																				
1	0	1																																																				
1	1	1																																																				
A	B	Z																																																				
0	0	0																																																				
0	1	1																																																				
1	0	1																																																				
1	1	0																																																				
Inverter  <table><tr><th>A</th><th>Z</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	Z	0	1	1	0	NAND  <table><tr><th>A</th><th>B</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Z	0	0	1	0	1	1	1	0	1	1	1	0	NOR  <table><tr><th>A</th><th>B</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Z	0	0	1	0	1	0	1	0	0	1	1	0	XNOR  <table><tr><th>A</th><th>B</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Z	0	0	1	0	1	0	1	0	0	1	1	1
A	Z																																																					
0	1																																																					
1	0																																																					
A	B	Z																																																				
0	0	1																																																				
0	1	1																																																				
1	0	1																																																				
1	1	0																																																				
A	B	Z																																																				
0	0	1																																																				
0	1	0																																																				
1	0	0																																																				
1	1	0																																																				
A	B	Z																																																				
0	0	1																																																				
0	1	0																																																				
1	0	0																																																				
1	1	1																																																				





LOGIC GATE I NOT



NOT 게이트 (부정)

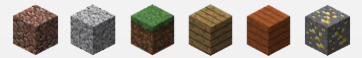
입력이 0이면 출력이 1, 입력이 1이면 출력이 0

A	Y
0	1
1	0





LOGIC GATE I AND I OR



AND 게이트 (논리곱)

모든 입력이 1일 때만 출력이 1

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR 게이트 (논리합)

입력 값 중 하나만 1이어도 출력이 1

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1





LOGIC GATE I NAND I NOR



NAND 게이트

AND의 결과를 NOT으로 뒤집음

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

NOR 게이트

OR의 결과를 NOT으로 뒤집음

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0





LOGIC GATE I XOR I XNOR



XOR 게이트 (배타적 논리합)

입력이 서로 다르면 출력이 1, 같으면 0

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

XNOR 게이트

XOR의 결과를 NOT으로 뒤집음

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1





LOGIC GATE



QUIZ

아까 만들었던 비밀번호 문에 사용된 게이트 종류는?

(Hint : 두 가지 사용됨)





MBC

3

LETS SILSOAP MANN

졸근 중인

Q.

지금도 집에 가고 싶다

여!!!



MU NONRI MANN



레드스톤에 대한 기초적인 설명 - 레드스톤

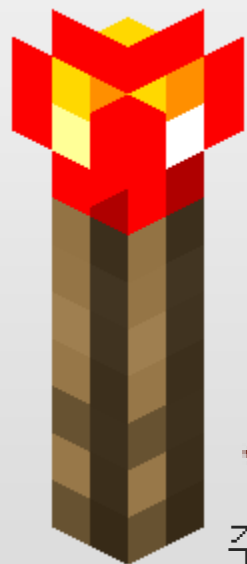




MU NONRI MANN



레드스톤에 대한 기초적인 설명 - 레드스톤 횃불



주변에서 신호를
받지 않았을 때



주변에서 신호를
받았을 때





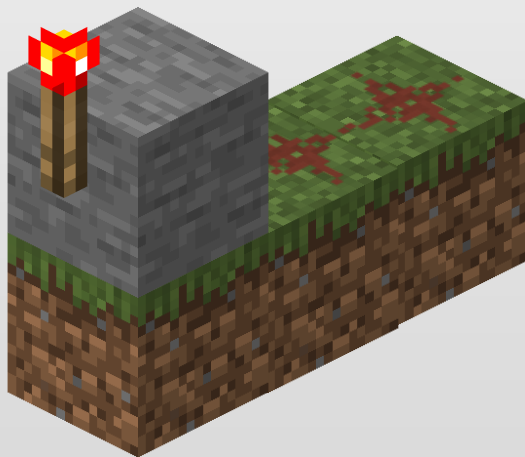
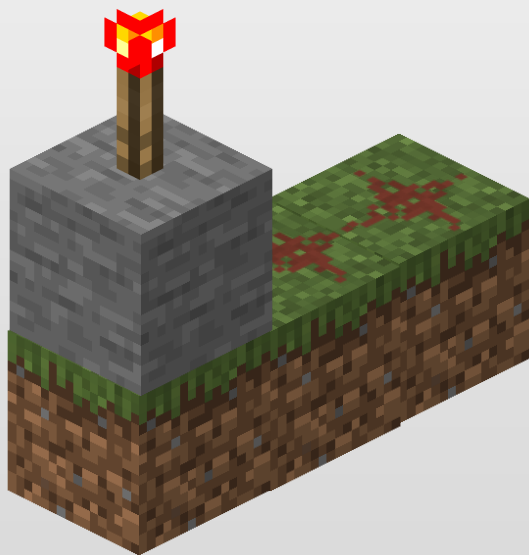
MU NONRI MANN



레드스톤에 대한 기초적인 설명 - 레드스톤 횃불

요론 식으로 설치해야됨

이 경우, 앞서 설명했던
레드스톤 횃불이 적용되지 않음





MU NONRI MANN



오늘 만들어볼 것

논리 게이트를 마크로 만들기





MU NONRI MANN



오늘 만들어볼 것(15분)

여러분이 논리 게이트를 마크로 만들기

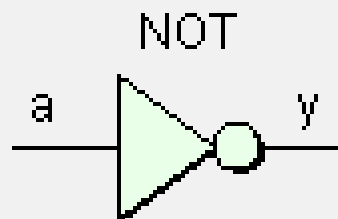
가장 인기많은 팀에겐
엄청난 상품이……?!!?!!?!!?!!?!!?

- **팁 : 피스톤, 레드스톤 횃불, 레드스톤 블록
레드스톤, 레버 등을 활용하면 좋아요**

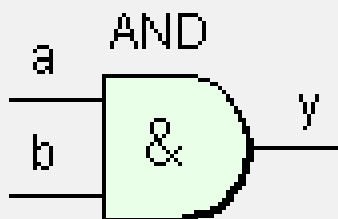




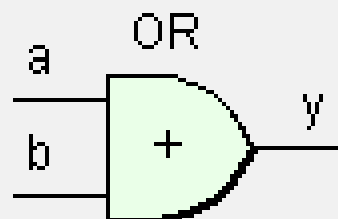
논리게이트 참의적으로 만들기



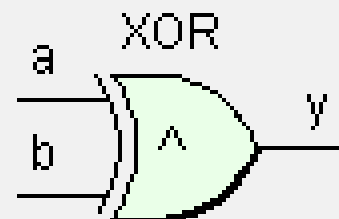
a	y
0	1
1	0



a	b	y
0	0	0
0	1	0
1	0	0
1	1	1



a	b	y
0	0	0
0	1	1
1	0	1
1	1	1



a	b	y
0	0	0
0	1	1
1	0	1
1	1	0





MU NONRI MANN



정답 중 하나

여러 가지 작품이 나올 수 있어요! 작동만 잘 되면 돼요



AND



OR



XOR



NOT





MBC

4

CALCULATOR MAKE MANNNN

졸근 중인

Q.

지금도 집에 가고 싶다

예!!!



MAKE MANNNN



이제 여러분의 작품을 뽐내 보세요!





MAKE MANNNN



레드스톤 회로로 작품 만들기

앞서 배운 게이트를 사용해도 되고,
새롭게 회로를 만들어도 돼요!





MAKE MANNNN



예시(내가 만든 것)

이진수 계산기





LETS QUIZ MANNNN



<https://kahoot.it/>

Kahoot!



A vibrant, pixelated Minecraft scene featuring a village. In the foreground, a villager with a beard and blue shirt holds a green emerald, standing next to a villager with red hair. To the left, a pig and a chicken are visible. The background shows a wooden house, a stone structure, and a hill with trees. The sky is blue with white clouds.

THANK YOU

Build #15H530
Made By Chicken-Jocky