Python

Q1

```
score = [80, 75, 55]
avg = ((score[0]+score[1]+score[2])/3)
avg = sum(score)//len(score)
print(avg)

total = 80+75+55
print(total/3)

print( (80+75+55) /3)

score = [80, 75, 55]
total = 0
for i in score:
    total = total + i
avg = total / 3
print(f"평균은 {avg}점 입니다.")
```

▶ avg = total / len(score) 사용해도 됨

Q2

```
print(13 // 2)
print(13 % 2)

su=int(input("숫자를 입력하세요: "))
if su % == 0:
print("짝수입니다.")
else:
print("홀수입니다.")

if 13 % 2 == 0:
print("짝수입니다.")
else :
print("홀수입니다.")
```

Q3

```
pin = "881120-1068234"
ymd = print(pin[:6])
num = print(pin[7:])
```

```
print(ymd)
print(num)

Q4
pin = "881120-1068234"
print(pin[7])
print(pin[-7])

if pin[7] == “1”:
print(“남자”)
else:
print(“여자”)

Q5
a = "a:b:c:d"
b = a.replace(":", "#")
print(b)

Q6
a = [1,3,5,4,2]
a.sort()
a.reverse()
print(a)

Q7
a = ['Life', 'is', 'too', 'short']
result = " ".join(a)
print(result)

Q8
a = (1, 2, 3)
a = a + (4,)
print(a)

Q9
a = dict()
print(a)
a['name'] = 'python'
a[('a',)] = 'python'
a[[1]] = 'python'
a[250] = 'python’
```

```
a[[1]] = 'python'
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    a[[1]] = 'python'
TypeError: unhashable type: 'list'
```

※ list 값이변하기에 key값으로 사용이 불가함.

Q10

```
a = {'A':90, 'B':80, 'C':70}
result = a.pop('B')
print(a)
print(result)
```

Q11

```
a = [1, 1, 1, 2, 2, 3, 3, 3, 4, 4, 5]
aSet = set(a)
b = list(aSet)
print(b)
```

※ set과 dictionary의 차이

```
a = [1, 1, 1, 2, 2, 3, 3, 3, 4, 4, 5]
a
[1, 1, 1, 2, 2, 3, 3, 3, 4, 4, 5]
aSet = set(a)
aSet
{1, 2, 3, 4, 5}
type(aSet)
<class 'set'>
```

dictionary는 같은 중괄호{}를 쓰지만 {key값:value값}으로 구성됨.

Q12

```
a = b = [1, 2, 3]
a[1] = 4
print(b)
```

[1, 4, 3]이 출력된다.

a와 b 변수는 모두 동일한 [1, 2, 3]이라는 리스트 객체를 가리키고 있기 때문이다.

GitHub

버전관리(형상관리)

프로그램을 수정한다고 가정할 때

잘못 수정하였을 때 Rollback을 해야 함.
그냥 수정을 하면 Rollback이 어렵기에 원본을 복사하고 수정을 해야 함.
동시성 작업(병행처리)을 제어를 GitHub에서 함
보통은 프로그래머들이 많이 사용
시스템 서버를 관리 할 때도 사용

구글 - 개발자를 위한 github 사용법(참고하여 초기설정)

↑ 윈도우에서 Git설치하는 방법

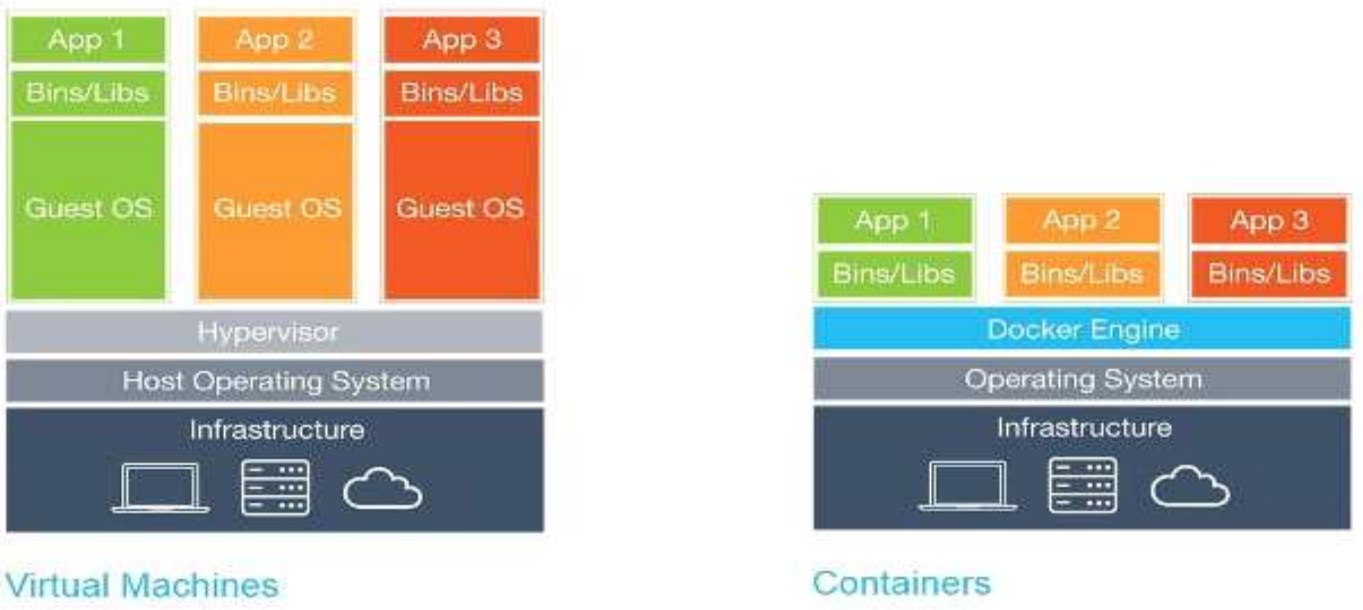
<Git 원격저장소 초기 설정>
echo "# busanit" >> README.md
git init
git add README.md → 추가할 파일
git commit -m "first commit" → 메세지
git branch –M main =>설치 초기설정 Master로 했을시에만 쓰면 됨.
git remote add origin https://github.com/abc1234/busanit.git
git push -u origin main → 옮기기

※ >, >>
> : 파일을 만드는데 파일이 있으면 지우고 새로 만듬
>> : 파일을 만드는데 파일이 있으면 밑에 내용 추가

온프레미스
라우터나 스위치같은 네트워크 장비
PC와 서버 같은 시스템
방화벽과 안티바이러스 같은 보안 장비

온프레미스란 필요한 시스템을 구축하기 위해서 값비싼 하드웨어와 애플리케이션을 구매하여 기업 상황에 맞게 커스터마이징 하는 것을 의미한다.
즉. 데이터센터나 서버 룸과 같은 특정 공간에 IT 인프라를 구축하여 소프트웨어를 사용하는 방식으로, 인프라를 구축하기 위한 시간도 수개월 이상 걸리며 초기도입 비용, 운영 및 관리를 위한 유지보수 등 비용이 많이 드는 단점이 있다.
일반적으로 온프레미스 시스템을 구축하는데 시간이 수개월 이상 걸리고 비용 또한 많이 든다.
퍼블릭 클라우드(가상머신)가 나올 당시만 해도 온프레미스 환경이 금방이라도 모두 사라질 것 같았지만 보안 적인 이유로 비즈니스에 중요하고 보안이 필요한 서비스와 데이터는 온프레미스 환경에서, 덜 중요한 것은 퍼블릭 클라우드(가상머신) 환경을 사용하는 하이브리드 IT 인프라가 대세를 이루고 있다.

<가상머신과 도커>



도커 : 도커 엔진을 붙여서 미리 만들어놓은 (운영체제+서버) 이미지를 붙여 넣음.
클라우드 시스템 : 기계를 모두 구비해서 하는 온프레미스 -> 도크 시스템으로 넘어감.
AWS나 구글 클라우드 오라클 클라우드 = OS를 대여
웹호스팅 운영체제와 네트워크 공간을 대여
네트워크 토플로지(구조) 웹호스팅에서는 마음대로 하기 힘듬
클라우드에서는 원하는 대로 가능.

클라우드의 종류
Public Cloud : 일반적으로 공개된 Cloud(AWS, 구글, 오라클 등)
Private Cloud : Cloud 자체 구축하여 사용(내부적인 클라우드)

<Python>

■ if 조건문

```
# 논리값은 True, False
money = True
# if <조건문>:
if money:
    print("택시를 타고 가라")
else:
    print("걸어 가라")
```

■ and, or, not 연산

```
# x or y : 둘 중 하나만 조건이 맞아도 참이 됨
# x and y : 두 조건이 맞아야 참이 됨
money = 2000
card = True
if money >= 3000 or card:
    print("택시를 타고 가라")
else:
    print("걸어가라")
```

■ in, not in 조건문

```
a=[1,2,3,4,5]
5 in a
True
2 in a
True
7 in a
False

2 not in a
False
5 not in a
False
8 not in a
True

pocket = ['paper', 'cellphone', 'money']
if 'money' in pocket:
    print("택시를 타고 가라")
else:
    print("걸어가라")
```

```
# 조건문에서 아무 일도 하지 않게 설정을 싶다면?
pocket = ['paper', 'cellphone', 'money']
if 'money' in pocket:
    pass
else:
    pass
```

■ 다중 조건 사용하기(elif = else+if의 의미)

```
pocket = ['paper', 'cellphone', 'money']
if 'card' in pocket:
    print("카드를 꺼내라")
else:
    if 'money' in pocket:
        print("돈을 꺼내라")
    else:
        print("걸어가라")
```

만약에 그렇지 않으면 만약에 그렇지 않으면 같은 형태이기에 산만한 느낌이 듬

```
pocket = ['paper', 'cellphone', 'money']
if 'card' in pocket:
    print("카드를 꺼내라")
elif 'money' in pocket:
    print("돈을 꺼내라")
else:
    print("걸어가라")
```

```
pocket = ['paper', 'cellphone', 'money']
if 'money' in pocket: pass
else: print("카드를 꺼내라")
```

※if, else 조건문이 1줄인 경우 이같이 해도 됨.

```
# 조건문이 참인경우 if 조건문, 조건문이 거짓인 경우 else 조건문
score = 70
if score >= 60:
    message = "success"
else:
    message = "failure"
print(message)
```

```
pocket = ['paper', 'cellphone', 'money']
message = "택시를 타고가라" if 'money' in pocket else "걸어가라"
```

print(message)
이와 같이 사용하는 것을 조건부 표현식이라고 한다.

■ while 조건문

```
treeHit = 0
while treeHit < 10:
  # treeHit +=1
    treeHit = treeHit +1
    print("나무를 %d번 찍었습니다." % treeHit)
if treeHit == 10:
    print("나무 넘어갑니다.")
```

■ while 조건문 실행 예제문

```
prompt = """
1. Add
2. Del
3. List
4. Quit

Enter number: """
print(prompt)

number = 0
while number !=4:
    print(prompt)
    number =int(input())
```

■ while 조건문 강제로 빠져나가기

```
coffee = 10
money = 300
while money:
    print("돈을 받았으니 커피를 줍니다.")
    coffee = coffee -1
    print("남은 커피의 양은 %d개입니다." % coffee)
    if coffee == 0:
        print("커피가 다 떨어졌습니다. 판매를 중지합니다.")
        break
```

※ break를 만나면 강제로 빠져나감.

Tip>
※ 윈도 가상 데스크탑 만들기

Win+Tab
Win+Ctrl+방향키(좌or우)

```
coffee = 10
while True:
    money = int(input("돈을 넣어 주세요: "))
    if money == 300:
        print("커피를 줍니다.")
        coffee = coffee -1
    elif money > 300:
        print("거스름돈 %d를 주고 커피를 줍니다." % (money -300))
        coffee = coffee -1
    else:
        print("돈을 다시 돌려주고 커피를 주지 않습니다.")
        print("남은 커피의 양은 %d개 입니다." % coffee)
    if coffee == 0:
        print("커피가 다 떨어졌습니다. 판매를 중지 합니다.")
        break
```