Week 2 Homework #1-2 Report

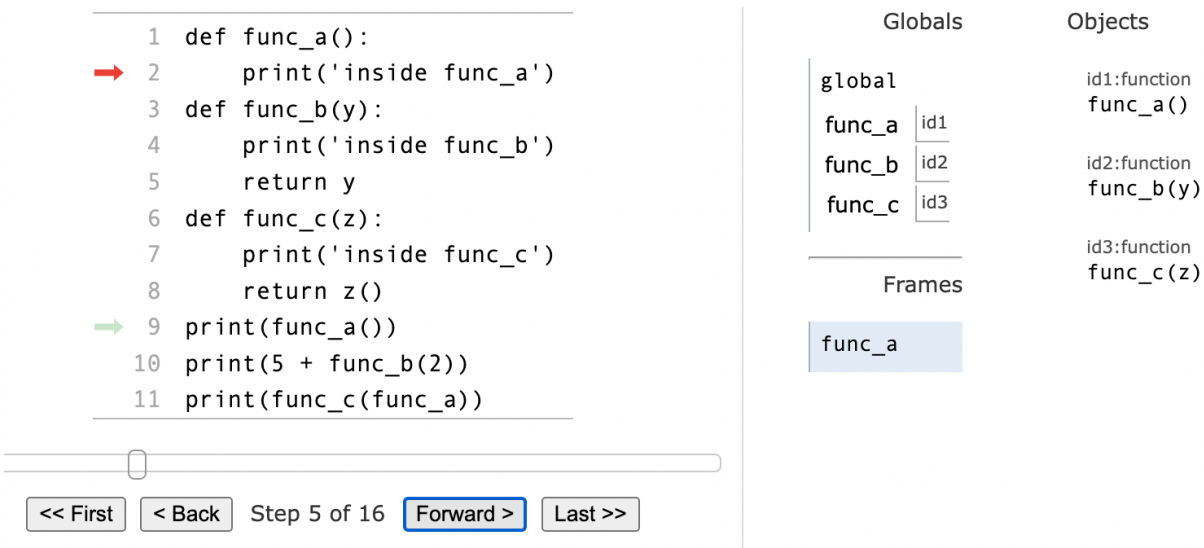
- 작성자 : B889047 윤준호
- 작성일 : 2022 / 09 / 19, Mon

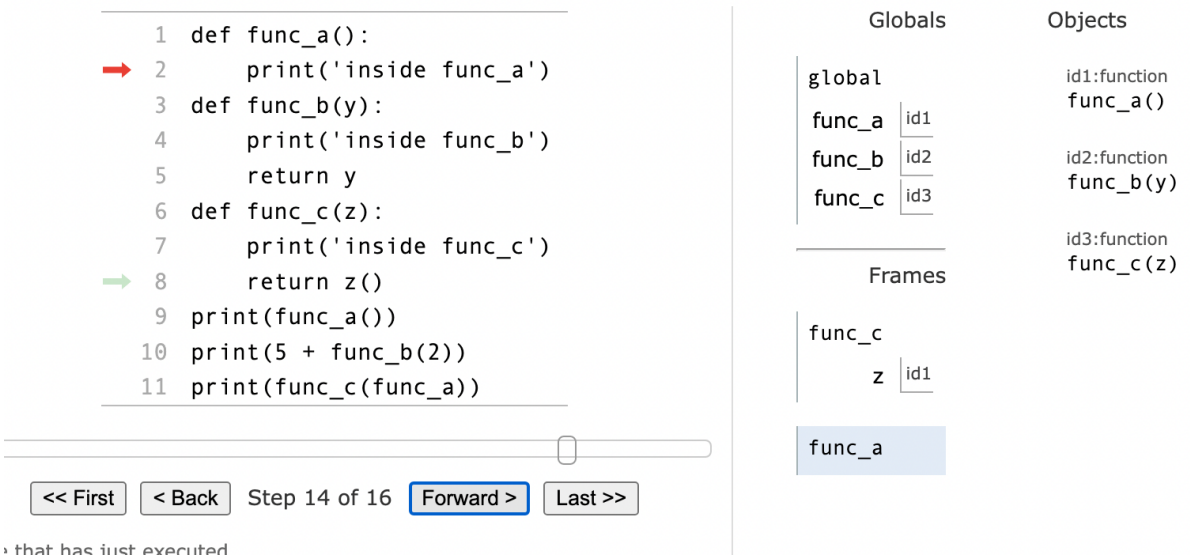
Python: Function, Modules, Lists

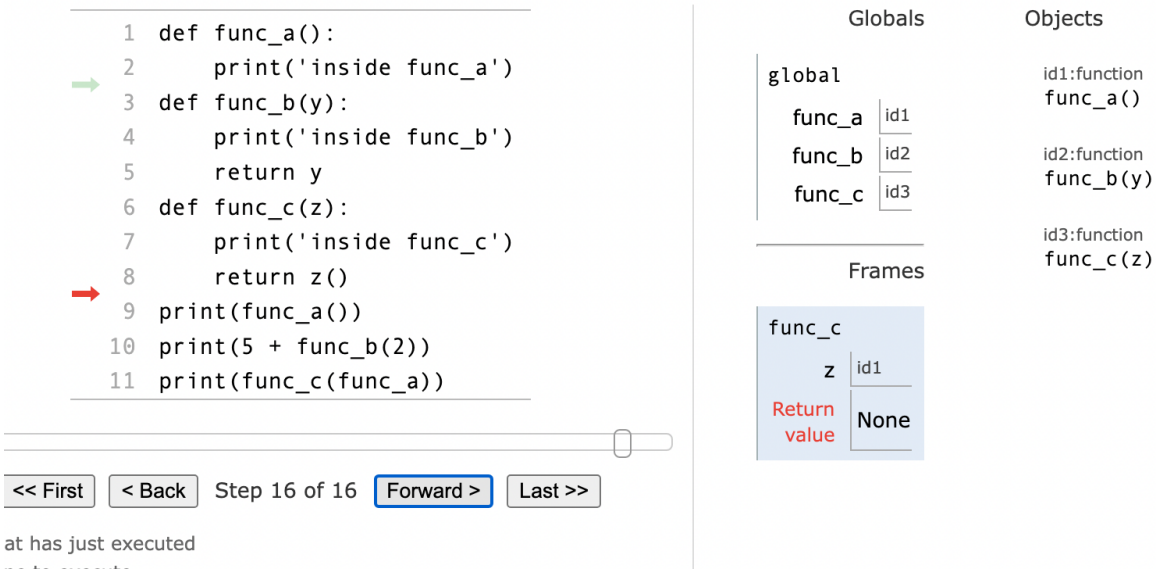
Use Python Tutor (http://cs1110.cs.cornell.edu/tutor (http://cs1110.cs.cornell.edu/tutor)) to answer and catch important snapshots for each problem along with you answers.

문제 #0: (함수를 함수의 인자로 넘기기)

```
def func_a():
    print('inside func_a')
def func_b(y):
    print('inside func_b') return y
def func_c(z):
    print('inside func_c') return z()
print(func_a())
print(5 + func_b(2))
print(func_c(func_a))
```



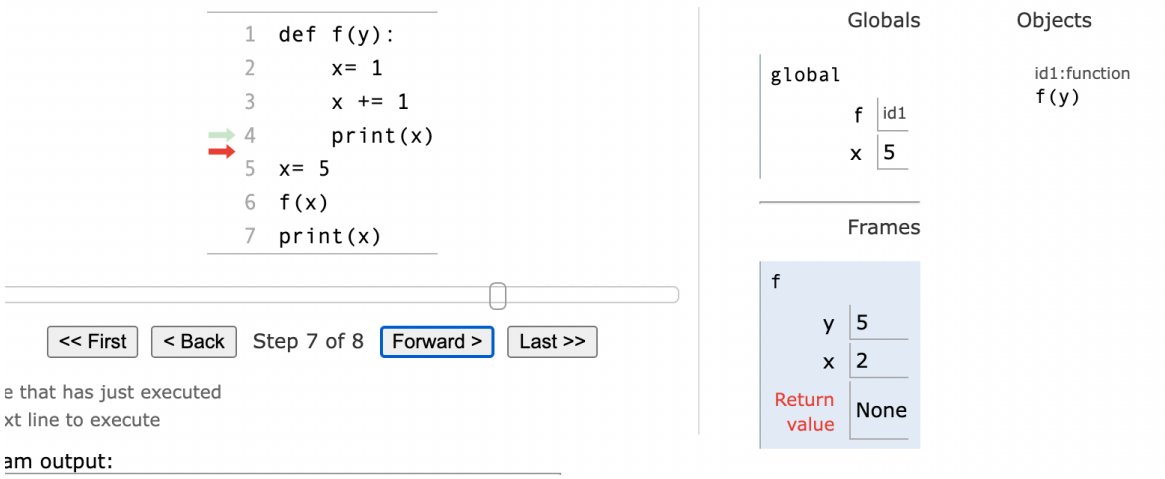


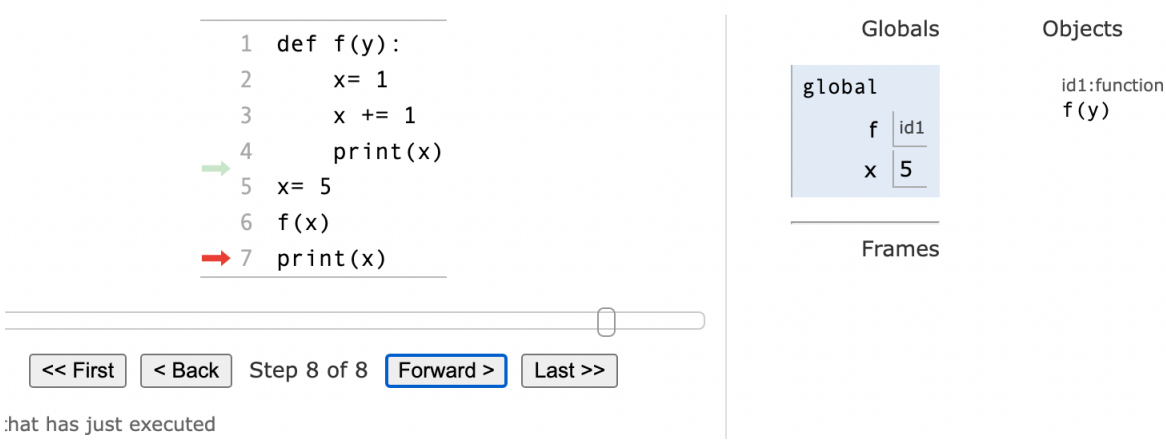


문제 #1: (함수에서 변수의 영역 문제: 기본 개념) a)

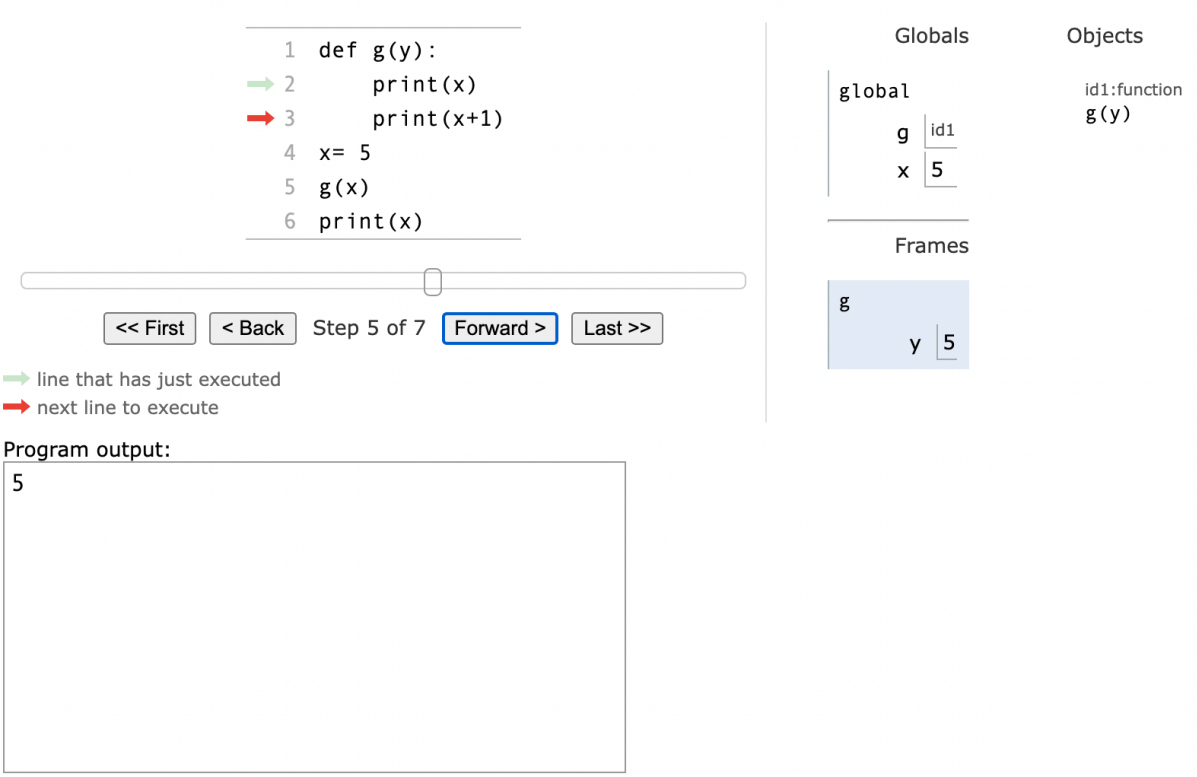
```
#a)
def f(y):
    x= 1
    x += 1
    print(x)
x= 5
f(x)
print(x)

#b)
def g(y):
    print(x)
    print(x+1)
x= 5
g(x)
print(x)

#c)
def h(y):
    x= x+ 1
x= 5
h(x)
print(x)
```

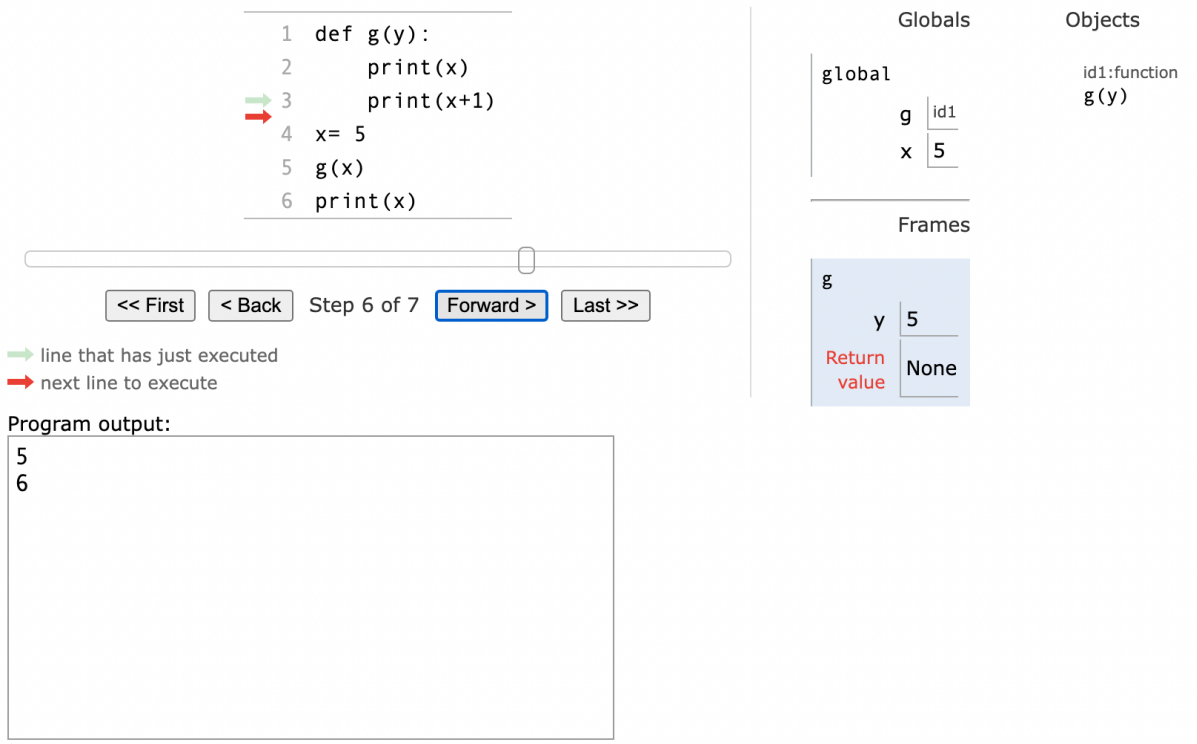
a)

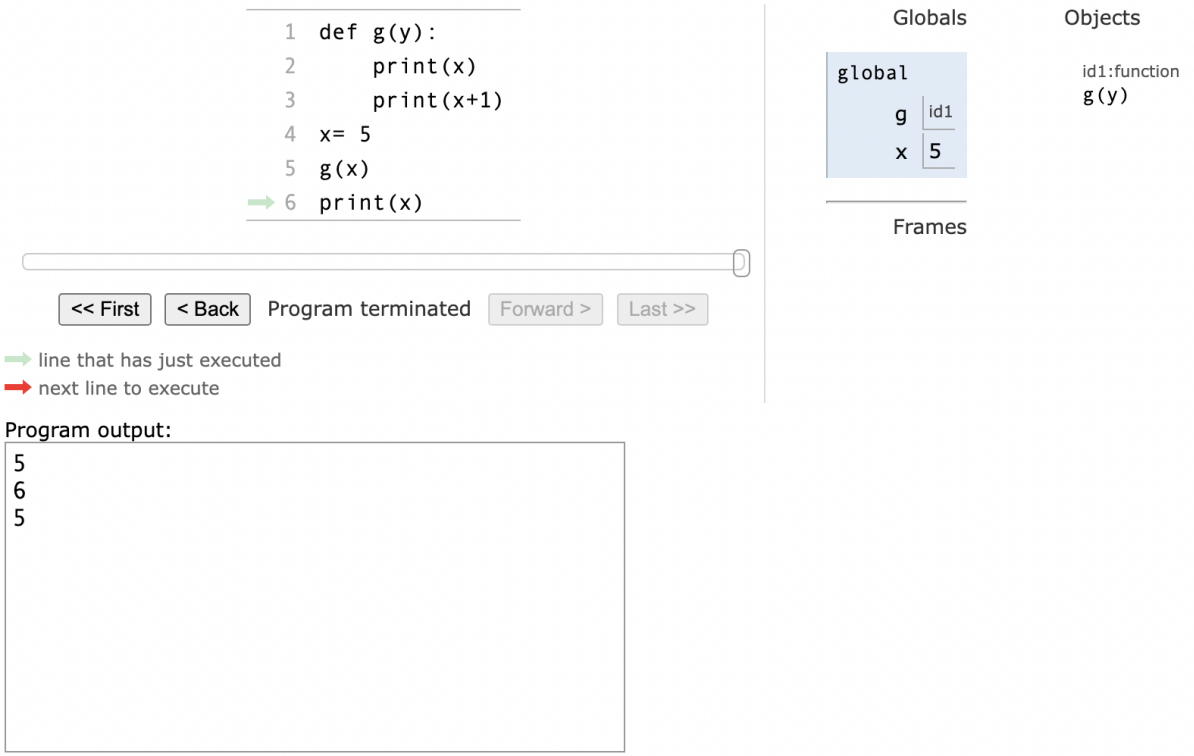




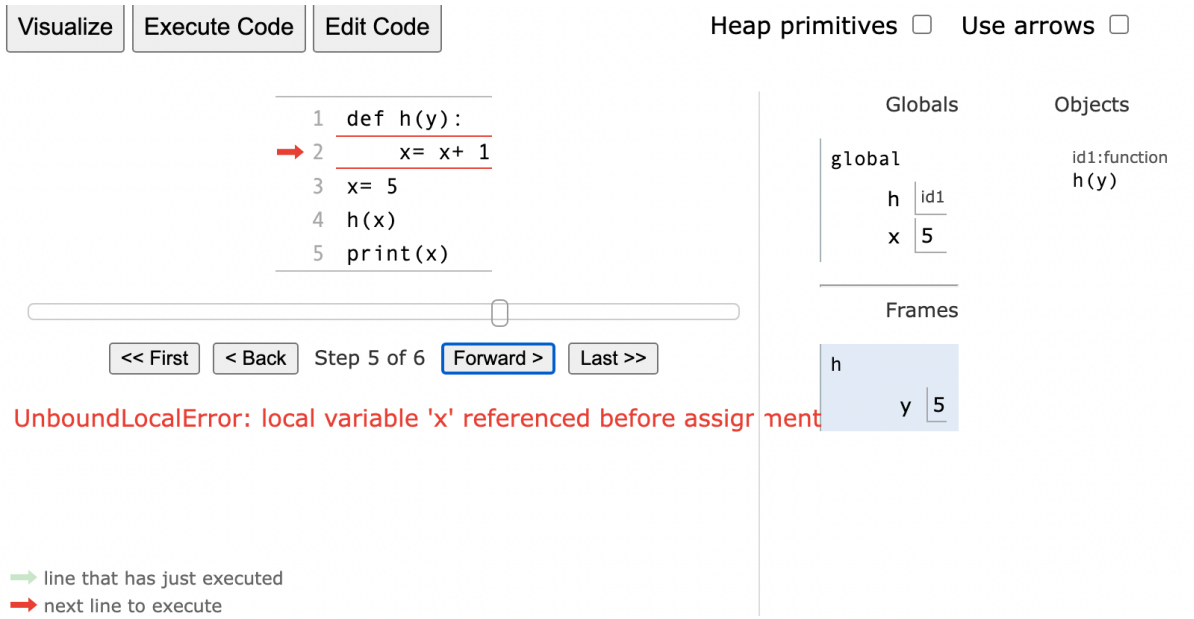
b)


1 def g(y):
2 print(x)
3 print(x+1)
4 x= 5
5 g(x)
6 print(x)
<< First
< Back
Step 5 of 7
Forward >
Last >>
line that has just executed
next line to execute
Program output:
5
Globals
Objects
global
g id1
x 5
id1:function
g(y)
Frames
g
y 5


1 def g(y):
2 print(x)
3 print(x+1)
4 x= 5
5 g(x)
6 print(x)
<< First
< Back
Step 6 of 7
Forward >
Last >>
line that has just executed
next line to execute
Program output:
5
6
Globals
Objects
global
g id1
x 5
id1:function
g(y)
Frames
g
y 5
Return value
None



c)

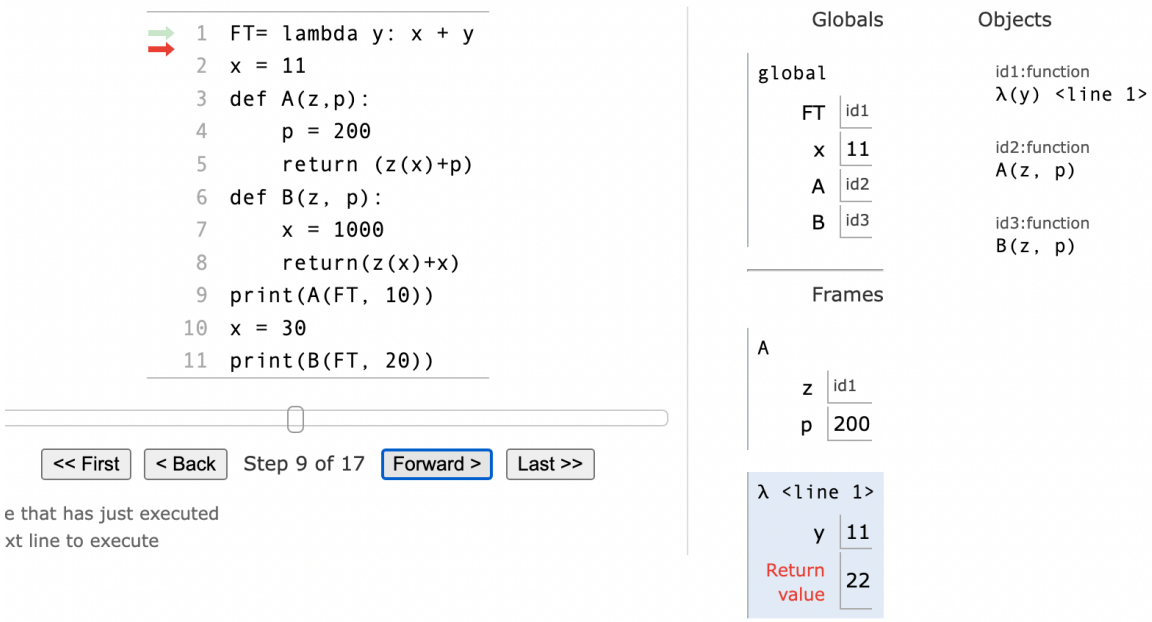


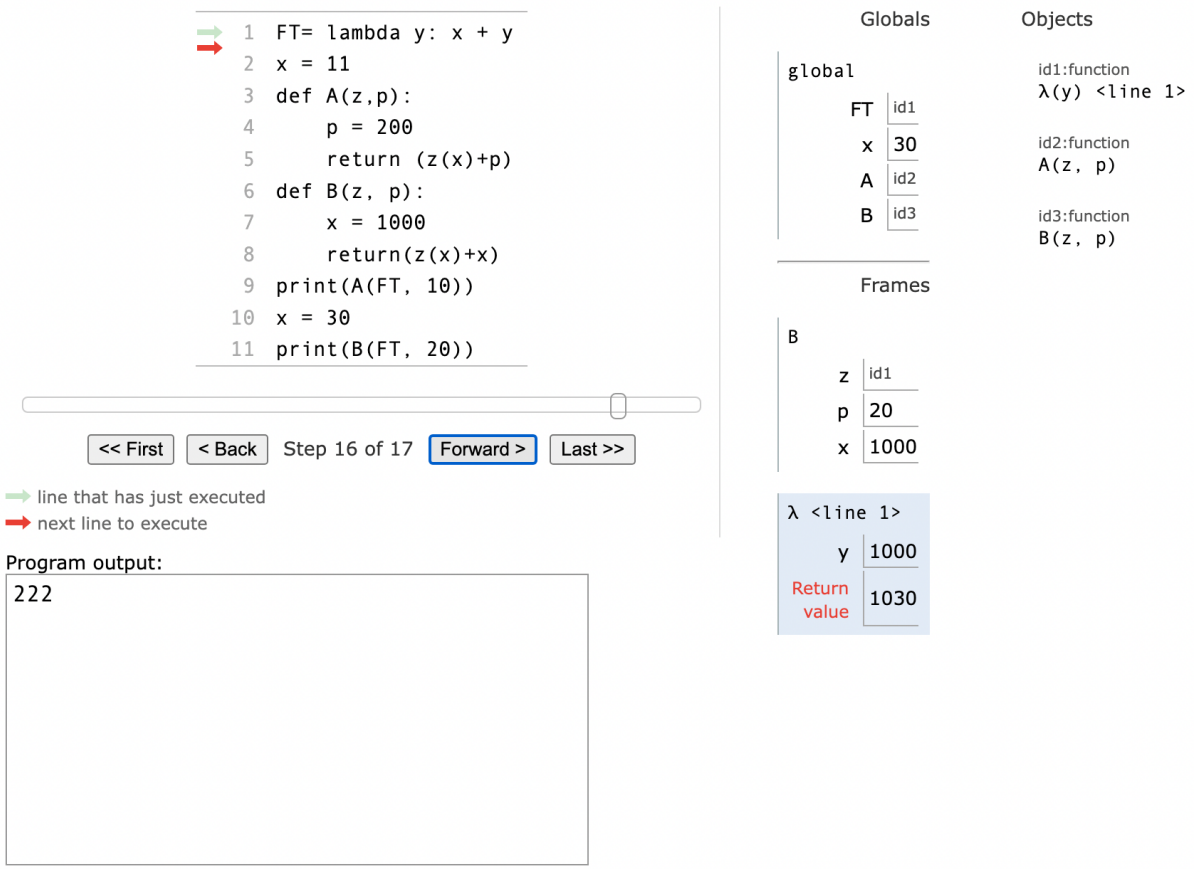
문제 #2: (람다 함수에서 변수의 영역 문제)

아래 코드가 수행된 결과값과 값의 타입을 말하라. 혹은 에러가 발생한다면 이유 는 무엇인가?

```
#a)
FT= lambda y: x + y
x = 11
def A(z,p):
    p = 200
    return (z(x)+p)
def B(z, p):
    x = 1000
    return(z(x)+x)
print(A(FT, 10))
x = 30
print(B(FT, 20))

#b)
def ft (f,x):
    y =4
    return f(x)
y =3
ft(lambda x: x * y,2)
```

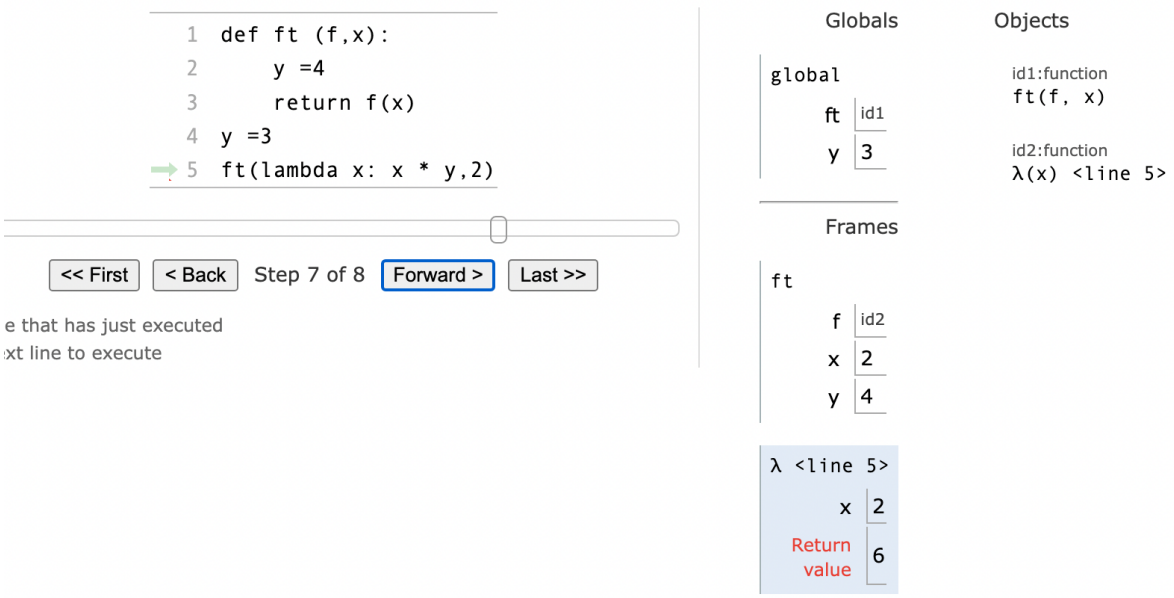
a)





A : A와 B모두 int형 값이 반환된다

b)



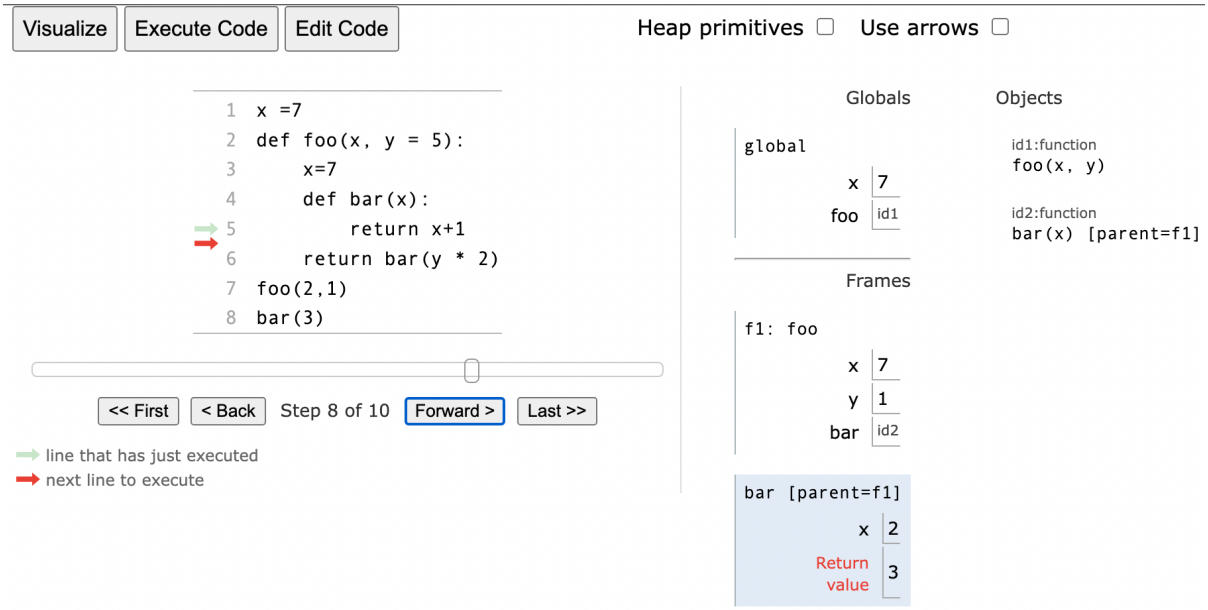
A : ft() 반환값은 int형이 반환된다. 다만 주의할 점은, ft()에 넘겨주는 lambda함수는 ft()함수의 스코프 밖에 있으므로, ft()내부의 y가 아닌 global영역의 y를 참고하게 된다

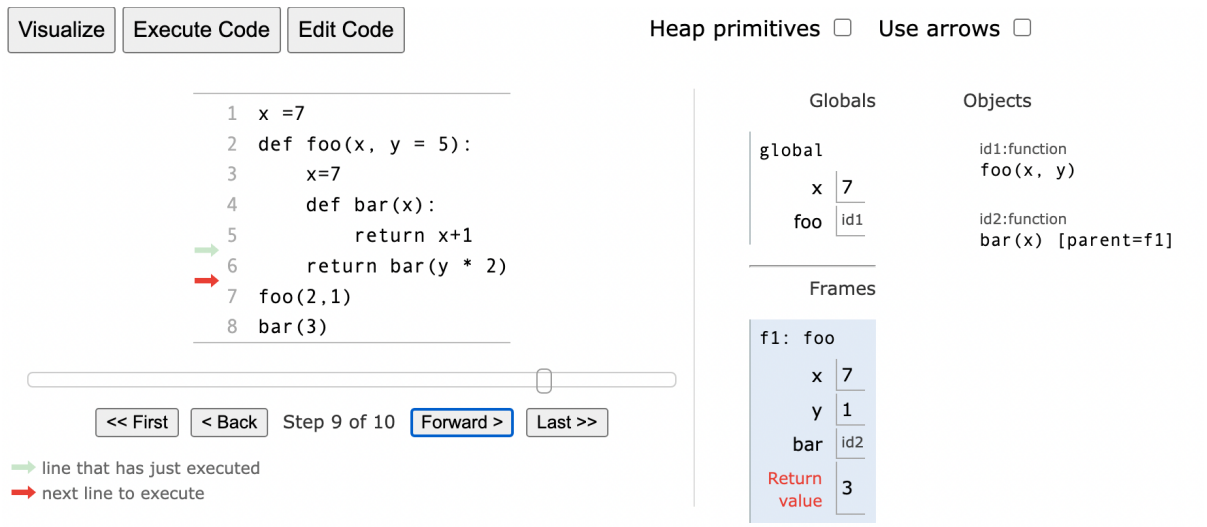
문제 #3: (함수에서 변수의 영역 문제)

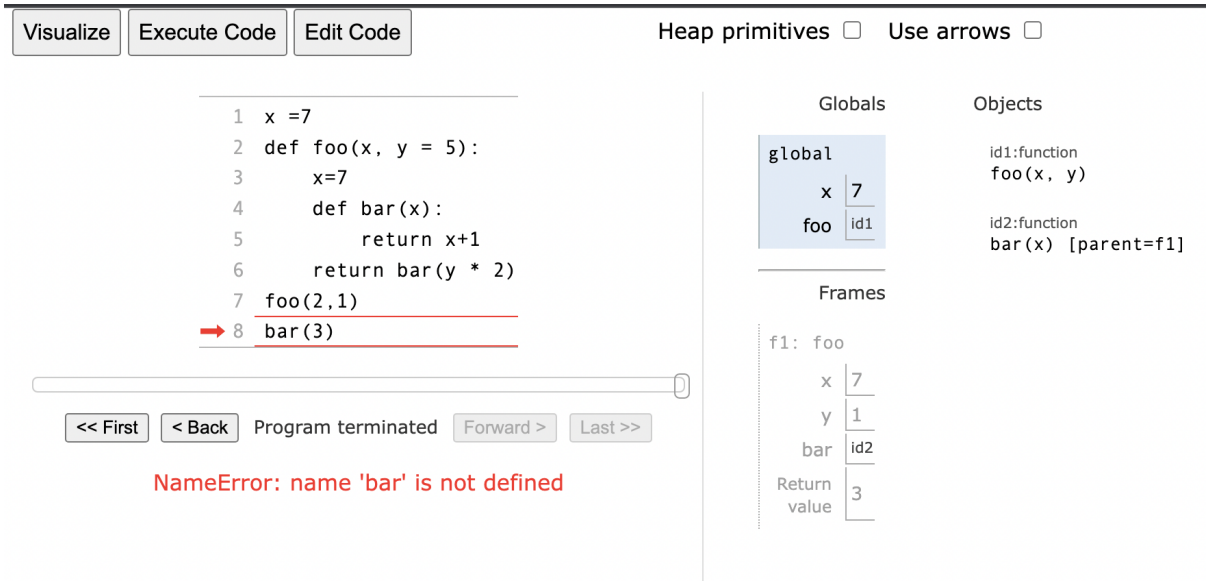
아래 코드의 수행된 결과값과 값의 타입을 말하라. 혹은 에러가 발생하는가?

```
x =7
def foo(x, y = 5):
    x=7
    def bar(x):
        return x+1
    return bar(y * 2)
foo(2,1)
bar(3)
```

A : 결론은 에러가 발생한다. `foo(2,1)` 은 int형이 반환이 되지만, bar(3)같은 경우에는 foo()함수 스코프 내부에 있는함수이므로, global스코프에서 접근이 불가능하다.







문제 #4: (Python List Handling)

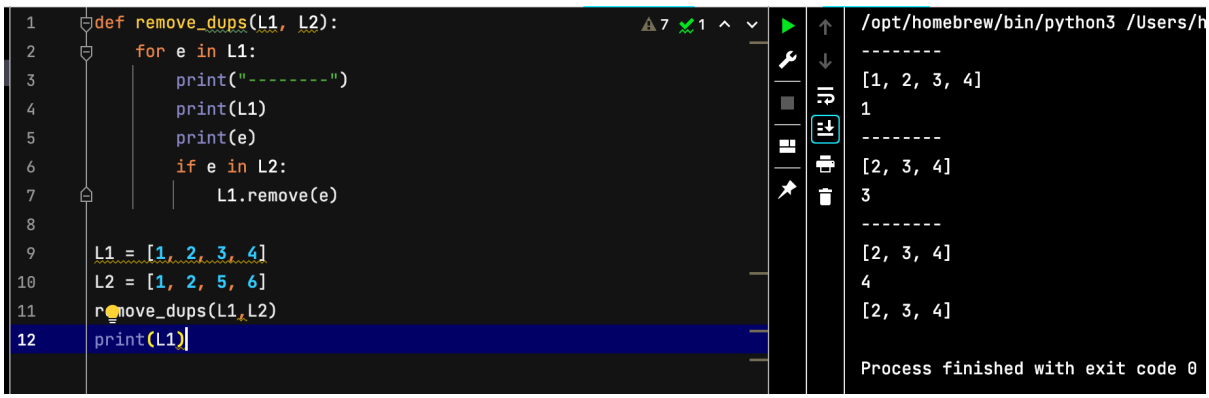
아래 문제는 리스트 L1의 원소 중에 L2에 있는 원소들은 L1에서 제거하는 코드를 작성 하는 것에 관한 것이다. 예시:

```
> L1 = [1, 2, 3, 4]
> L2 = [1, 2, 5, 6]
> remove_dups(L1, L2)
> print(L1)
[3,4]
```

Q : A. 아래 구현된 코드는 우리가 원하는 기능을 수행하는가? 수행 결과는 무엇이며, 이유를 설명하라.

```
def remove_dups(L1, L2):
    for e in L1:
        if e in L2:
            L1.remove(e)
```

A : 작동하지 않는다. 기본적으로 remove()는 실행되는 시퀀스의 원본 값을 변경하는 방식으로 작동이 된다. 그리고 리스트는 이터레이터라는것을 기반으로 만들어지며, for문으로 순회를 하면, 하나의 원소씩 다음 index의 값을 반환하게 된다. 그렇기 때문에 첫번째 순회에서 1이 없어지고 나서 index 1번을 반환해야 하지만, 2가 아닌 3을 선택해서 반환하게 된다. 그렇기 때문에, 2는 제거되지 못하는 현상이 발생하게 되는것이다.



B. A에 주어진 코드를 수정하여 원래 문제에서 주어진 기능을(공통원소제거) 수행하는 코드를 작성해 보라.

In [1]:

```
# B - Solution 1
def remove_dups(L1, L2):
    for e in L1[:]:
        if e in L2:
            L1.remove(e)

L1 = [1, 2, 3, 4]
L2 = [1, 2, 5, 6]
remove_dups(L1,L2)
print(L1)

# B - Solution 2
def remove_dups(L1, L2):
    return list(filter(lambda x: x not in L2, L1))

L1 = [1, 2, 3, 4]
L2 = [1, 2, 5, 6]
L1 = remove_dups(L1,L2)
print(L1)

# B - Solution 3
def remove_dups(L1, L2):
    return [ i for i in L1 if i not in L2 ]

L1 = [1, 2, 3, 4]
L2 = [1, 2, 5, 6]
L1 = remove_dups(L1,L2)
print(L1)
```

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

문제 #5: (Python Basics)

코드 1과 2 중 아래 주어진 코드와 같은 기능을 수행하는 코드는 어느 것인가? 세 코드 의 기능을 각각 말하라.

주어진 코드: while 루프안에 for 문이 있다.

```
iteration = 0
count = 0
while iteration < 3:
    for letter in "hello, world":
        count += 1
    print("Iteration " + str(iteration) + "; count is: " + str(count))
    iteration += 1
```

```
# 코드 1
for iteration in range(3):
    count = 0
    while True:
        for letter in "hello, world":
            count += 1
        print("Iteration " + str(iteration) + "; count is: " + str(count))
        break
# 코드 2
count = 0
phrase = "hello, world"

for iteration in range(3):
    while True:
        count += len(phrase)
        break
    print("Iteration " + str(iteration) + "; count is: " + str(count))
```

A : 주어진 코드와 동일한것을 수행하는 코드는 코드 2이다.

- 주어진 코드 : 큰 틀에서 총 3번의 순회를 한다. 각 순회마다 "hello, world" 문자열의 한글자씩 순회를하며 count변수에 1씩 더해준다.한번의 순회가 끝나면, iteration변수와 count변수를 출력하고, 큰 순회가 끝나기 전에 iteration변수에 1을 더해서 큰 틀의 순회를 count해준다
- 코드 1 : 큰 틀에서 총 3번의 순회를 한다. 한번의 순회마다 count변수는 0으로 초기화 되며, 한번의 순회 내의 또다른 순회가 존재한다. 내부 순회에서는 "hello, world"문자열의 한글자씩 순회하며, count변수에 1씩 더해준다. 한번의 순회가 끝나면, 외부 순회를 count하는 iteration변수와 count변수를 출력하고, 내부 순회를 끝낸다.
- 코드 2 : 주어진 코드와 동일한 기능을 한다

문제 #6: (Python List Handling)

아래 프로그램은 두개의 리스트를 합치는 코드이다. L1. L2 리스트는 같은 길이라고 하자.

- A. 리스트를 “합친다”는 의미를 아래 코드에서는 어떻게 해석(구현)하고 있는가? 예를 들어, L1=[2,4,6] L2=[1,2,5] 일 때 각 코드의 결과는?
- B. 코드 1과 코드 2의 차이점이 무엇인가?

```
# 코드 1

def Code1(L1, L2):
    temp = L1[:]
    for e2 in L2:
        flag = False
        for check in temp:
            if e2 == check:
                flag = True
                break
        if not flag:
            temp.append(e2)
    return temp


# 코드 2

def Code2(L1, L2):
    temp = []
    for e1 in L1:
        flag = False
        for e2 in L2:
            if e1 == e2:
                flag = True
                break
        if not flag:
            temp.append(e1)
    return temp + L2
```

결과값

- 코드 1 : [2, 4, 6, 1, 5]
- 코드 2 : [4, 6, 1, 2, 5]

A : 두 코드의 공통점은 중복된 원소 없이 합친다는 공통점이 있다. 차이점이라면, Code1같은 경우에는 L1의 복제본 temp를 기반으로, L2의 원소가 temp에 없는 경우에만 temp에 추가하고 반환한다. Code2같은 경우에는 빈 리스트를 기반으로, L1을 순회하고, 순회중인 L1의 원소를 기준으로 L2를 순회하며, 동일한 값이 있는 경우를 제외하고, L1의 원소를 추가하게 된다. 이렇게 하면, L1의 원소중 L2와 중복되지 않는 원소만 들어가게 되고, 마지막에 L2를 합친다. 그렇기 때문에, 코드1, 코드2 리스트 내부 값들은 동일하지만 순서가 다른것을 볼 수 있다.