```
# Core 2 Web
# Funtion Assignment 1
# Write the following code given below and perform a dry run for the provided codes.



def add(x):
  def innner(y):
    return x*y
  return innner

if __name__=='__main__':
  add_3 =add(3)
  result=add_3(7)
  print(result)
```

```
21
```

```
def outer():
  def inner():
    return "Greetings from the inner function!"
  return inner()
if __name__=="__main__":
  result= outer()
  print(result)
```

```
Greetings from the inner function!
```

```
def outer():
  def inner():
    return "This is the inner function."
  return inner
if __name__=='__main__':
  retObj = outer()
  retInner = retObj()
  print(retInner)
```

```
This is the inner function.
```

```
def outer():
  def inner():
    return outer
  return inner
if __name__=='__main__':
  retObj = outer()
  retInner = retObj()
  print(retInner)
```

```
<function outer at 0x7c596893f9c0>
```

```
def outer():
  def inner(outer):
    print(outer)
    return inner
  return inner(outer)
if __name__=="__main__":
  retObj = outer()
  print(retObj)
```

```
<function outer at 0x7c59817493a0>
<function outer.<locals>.inner at 0x7c596893cd60>
```

```
def outer():
  def inner1(a, b):
    print("In inner1")
    return a - b
  def inner2(obj):
    print("In inner2")
    print(obj)
    return inner2
  retInner1 = inner1(10, 4)
  retInner2 = inner2(retInner1)
  return retInner2
```

```
if __name__=="__main__":
  retObj = outer()
  print(retObj)
```

```
In inner1
In inner2
6
<function outer.<locals>.inner2 at 0x7c596893d080>
```

```
def outer():
  def inner():
    return "Hello, Core2web!"
  return inner
  print("In Outer Function")
if __name__=="__main__":
  result = outer()()
  print(result)
```

```
Hello, Core2web!
```

```
def outer():
  message = "I am the outer function."
  def inner():

    return message
  return inner
if __name__=="__main__":
  inner_function = outer()
  result = inner_function()
  print(result)
```

```
I am the outer function.
```

```
def outer():
  count = 0
  def inner():
    nonlocal count
    count += 1
    return count
  return inner
if __name__=="__main__":
  counter = outer()
  print(counter())
  print(counter())
```

```
1
2
```

```
def outer(flag):
  def inner():
    return "This is true." if flag else "This is false."
  return inner

if __name__=="__main__":
  true_function = outer(True)
  false_function = outer(False)
  print(true_function())
  print(false_function())
```

```
This is true.
This is false.
```

```
Start coding or generate with AI.
```