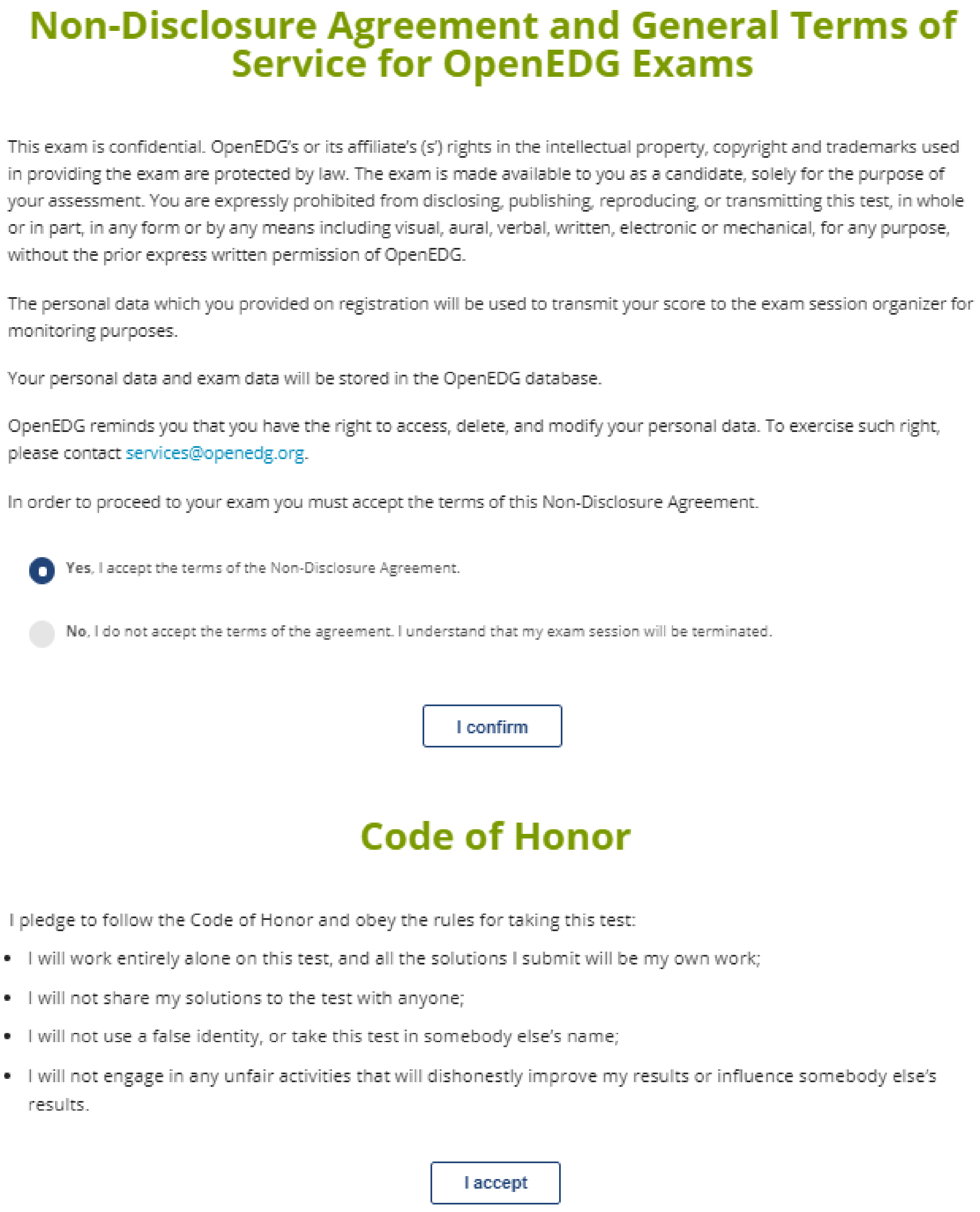


Non-Disclosure Agreement and General Terms of Service for OpenEDG Exams

This exam is confidential. OpenEDG's or its affiliate's (s') rights in the intellectual property, copyright and trademarks used in providing the exam are protected by law. The exam is made available to you as a candidate, solely for the purpose of your assessment. You are expressly prohibited from disclosing, publishing, reproducing, or transmitting this test, in whole or in part, in any form or by any means including visual, aural, verbal, written, electronic or mechanical, for any purpose, without the prior express written permission of OpenEDG.

The personal data which you provided on registration will be used to transmit your score to the exam session organizer for monitoring purposes.

Your personal data and exam data will be stored in the OpenEDG database.

OpenEDG reminds you that you have the right to access, delete, and modify your personal data. To exercise such right, please contact services@openedg.org.

In order to proceed to your exam you must accept the terms of this Non-Disclosure Agreement.

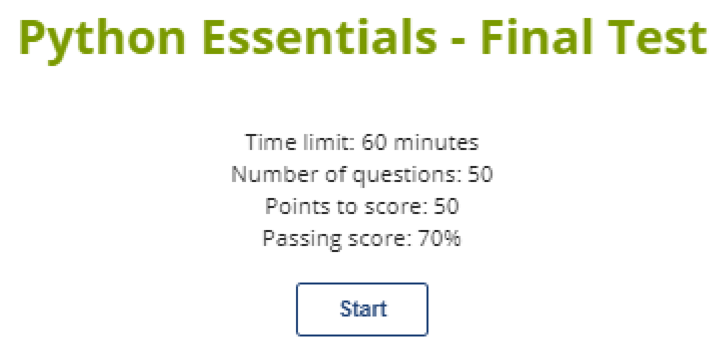
● **Yes**, I accept the terms of the Non-Disclosure Agreement.

○ **No**, I do not accept the terms of the agreement. I understand that my exam session will be terminated.

Code of Honor

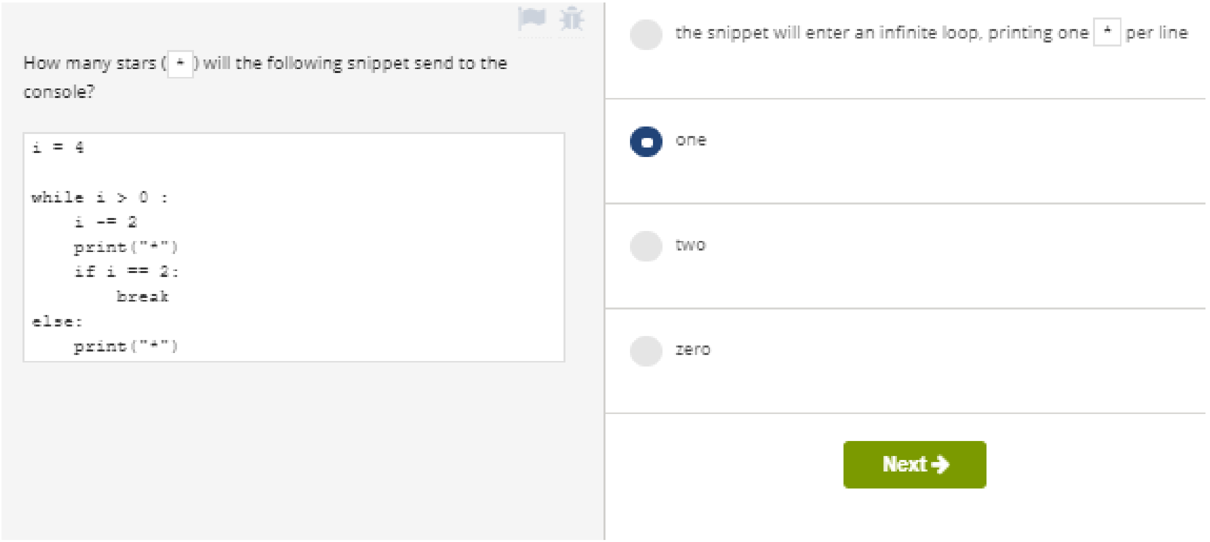
I pledge to follow the Code of Honor and obey the rules for taking this test:

- I will work entirely alone on this test, and all the solutions I submit will be my own work;
- I will not share my solutions to the test with anyone;
- I will not use a false identity, or take this test in somebody else's name;
- I will not engage in any unfair activities that will dishonestly improve my results or influence somebody else's results.



Python Essentials - Final Test

Time limit: 60 minutes
Number of questions: 50
Points to score: 50
Passing score: 70%



How many stars (*) will the following snippet send to the console?

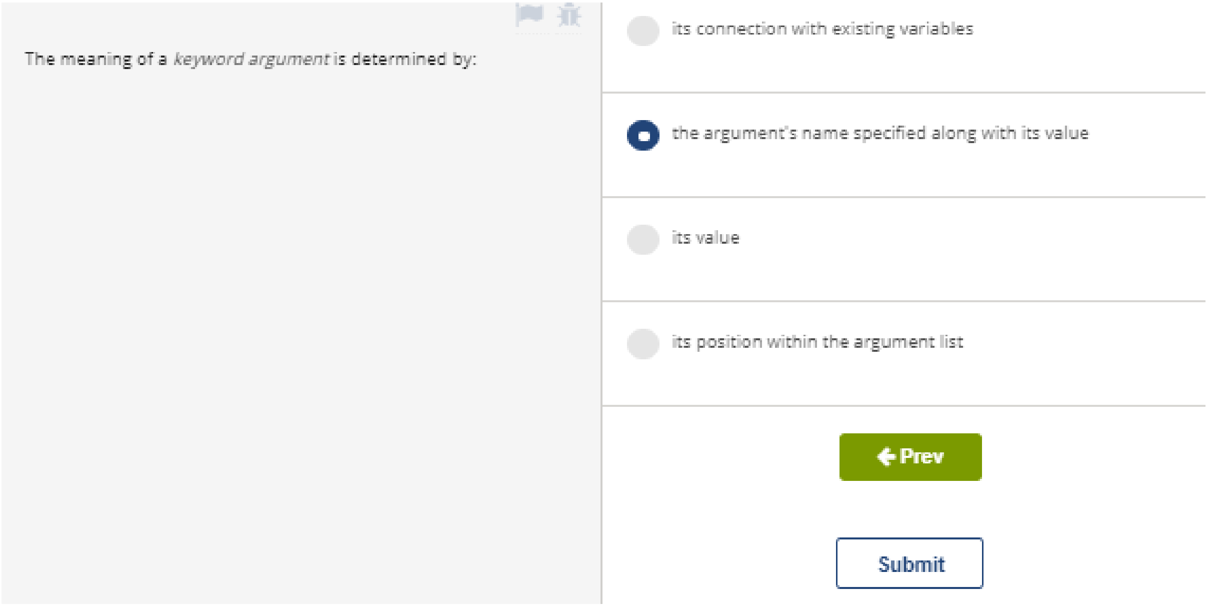
```
i = 4

while i > 0 :
    i -= 2
    print("*")
    if i == 2:
        break
else:
    print("*")
```

○ the snippet will enter an infinite loop, printing one * per line

● one

○ two

○ zero



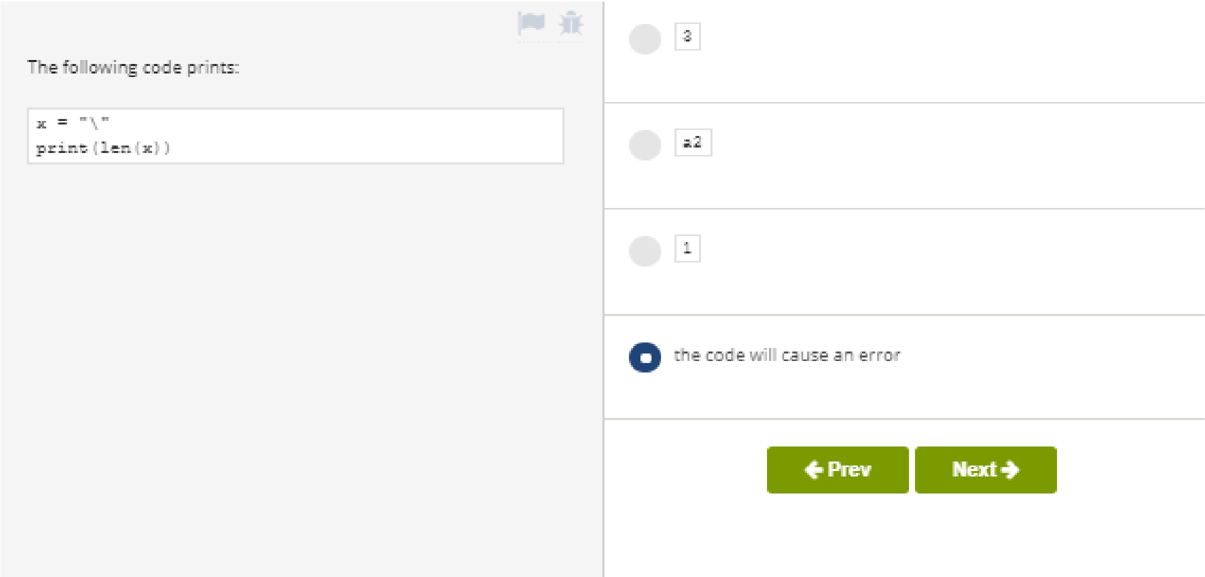
The meaning of a *keyword argument* is determined by:

○ its connection with existing variables

● the argument's name specified along with its value

○ its value

○ its position within the argument list



The following code prints:

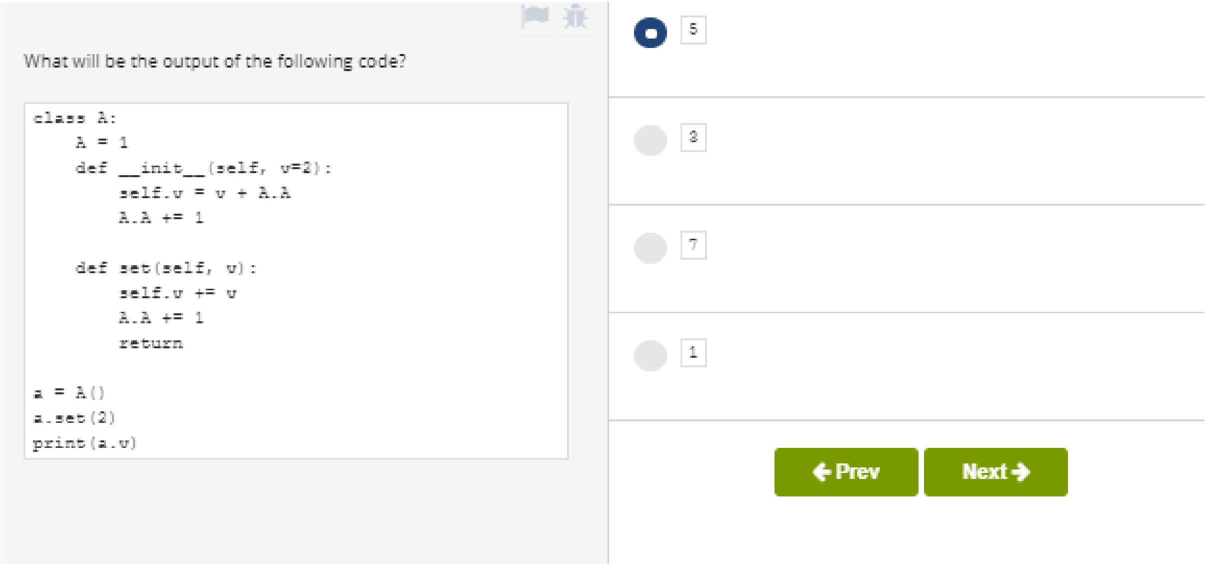
```
x = "\"
print(len(x))
```

○ 3

○ a2

○ 1

● the code will cause an error



What will be the output of the following code?

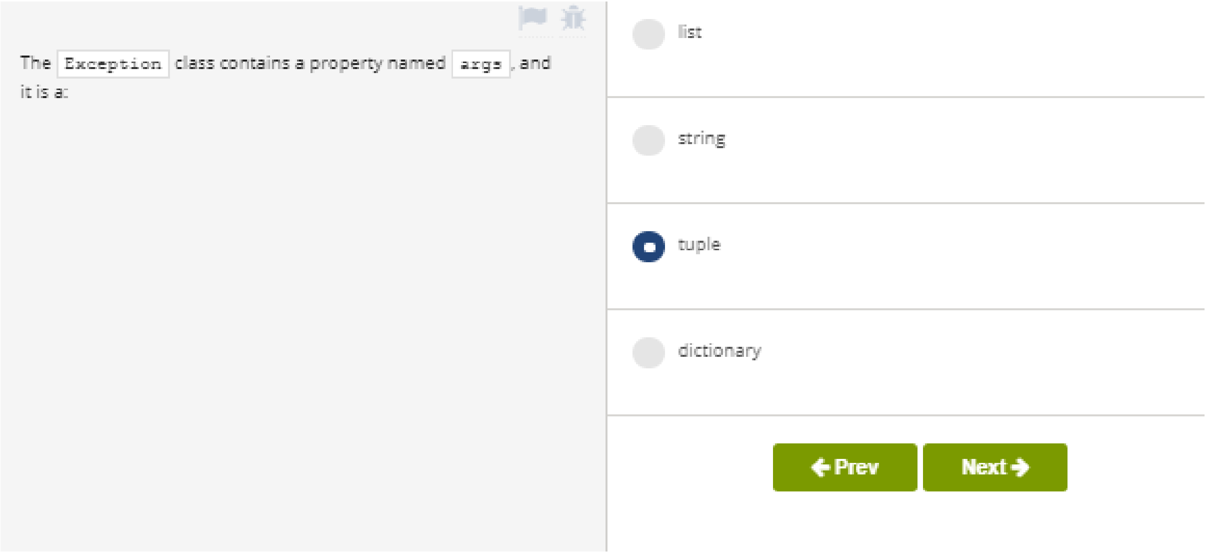
```
class A:
    A = 1
    def __init__(self, v=2):
        self.v = v + A.A
        A.A += 1

    def set(self, v):
        self.v += v
        A.A += 1
        return

a = A()
a.set(2)
print(a.v)
```

● 5

○ 3

○ 7

○ 1



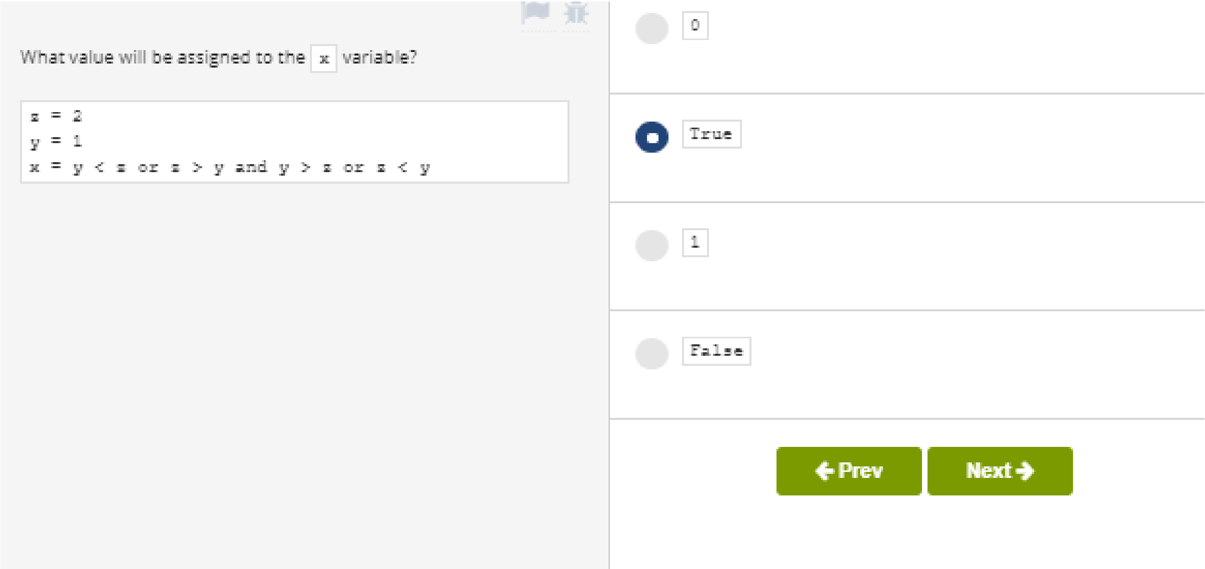
The `Exception` class contains a property named `args`, and it is a:

○ list

○ string

● tuple

○ dictionary



What value will be assigned to the `x` variable?

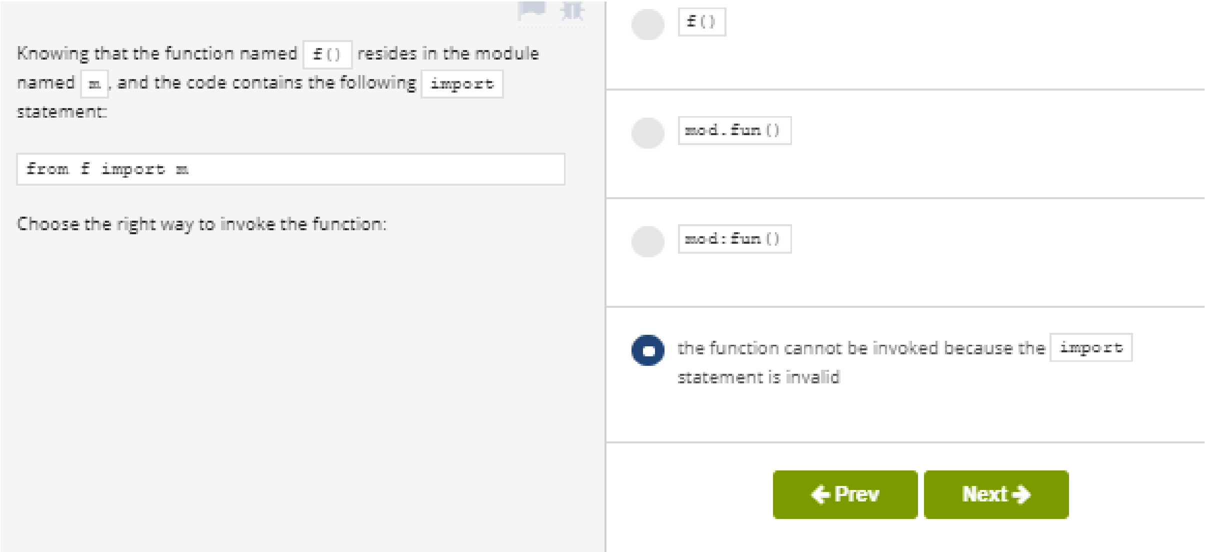
```
z = 2
y = 1
x = y < z or z > y and y > z or z < y
```

○ `0`

● `True`

○ `1`

○ `False`



Knowing that the function named `f()` resides in the module named `m`, and the code contains the following `import` statement:

```
from f import m
```

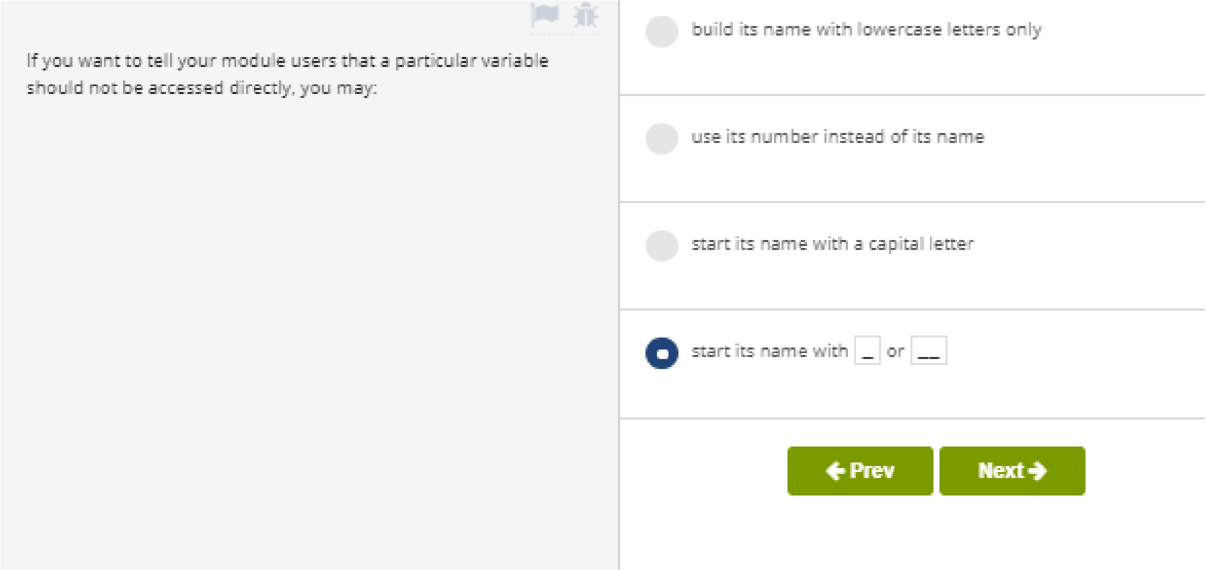
Choose the right way to invoke the function:

○ `f()`

○ `mod.fun()`

○ `mod:fun()`

● the function cannot be invoked because the `import` statement is invalid



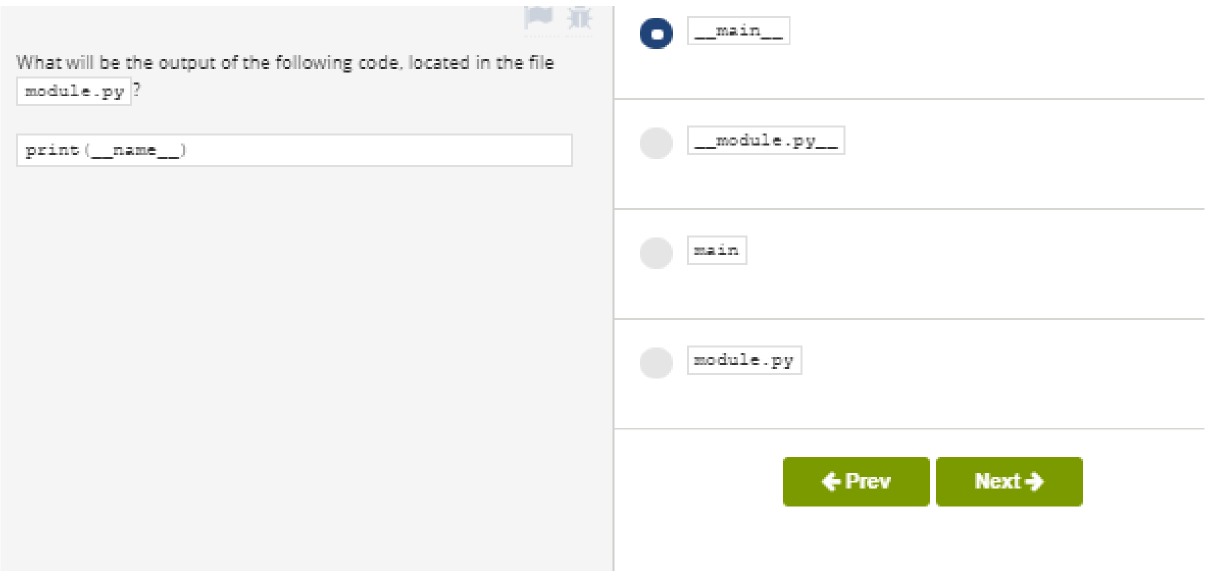
If you want to tell your module users that a particular variable should not be accessed directly, you may:

○ build its name with lowercase letters only

○ use its number instead of its name

○ start its name with a capital letter

● start its name with `_` or `__`



What will be the output of the following code, located in the file `module.py`?

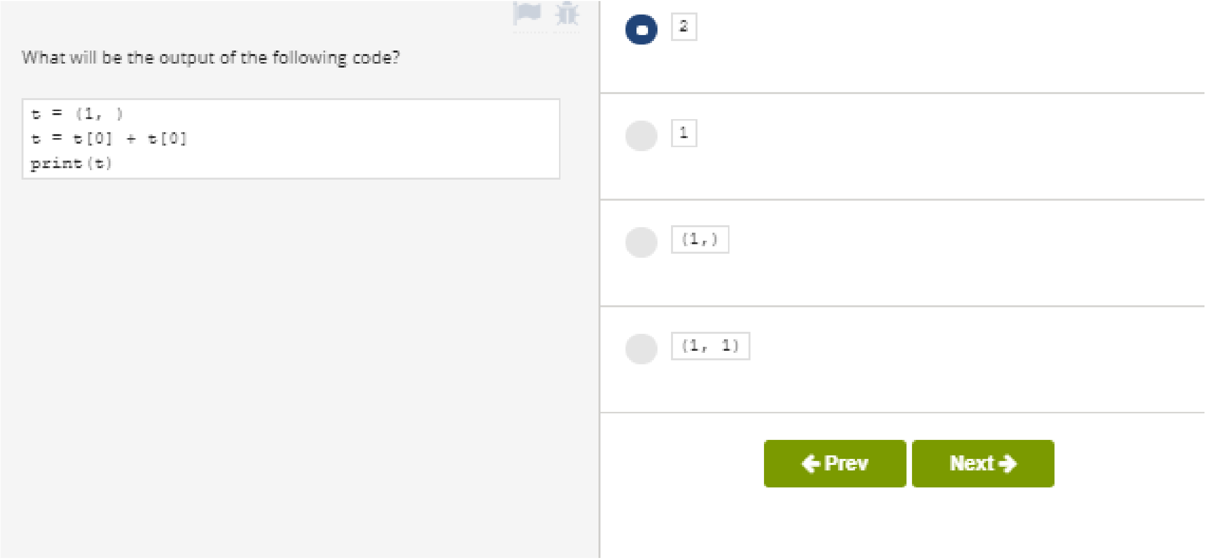
```
print(__name__)
```

● `__main__`

○ `__module.py__`

○ `main`

○ `module.py`



What will be the output of the following code?

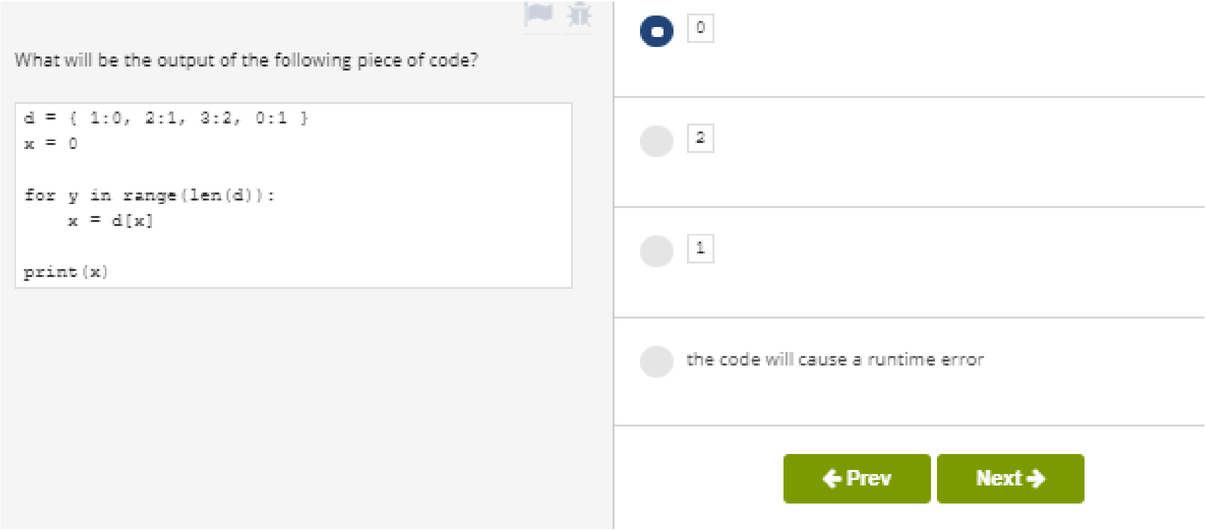
```
t = (1, )
t = t[0] + t[0]
print(t)
```

● `2`

○ `1`

○ `(1,)`

○ `(1, 1)`



What will be the output of the following piece of code?

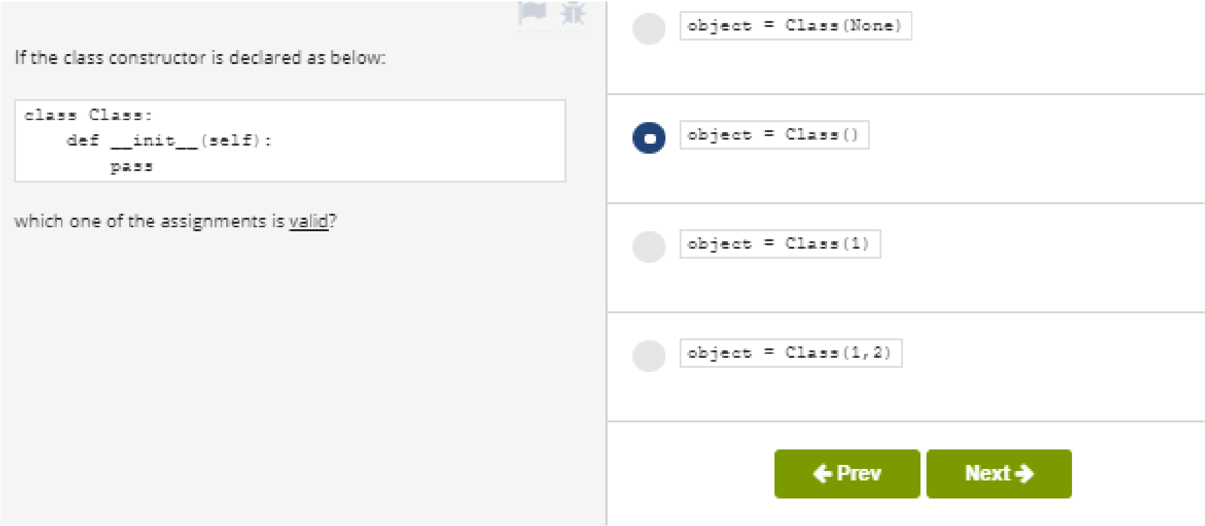
```
d = { 1:0, 2:1, 3:2, 0:1 }
x = 0

for y in range(len(d)):
    x = d[x]

print(x)
```

● `0`

○ `2`

○ `1`

○ the code will cause a runtime error



If the class constructor is declared as below:

```
class Class:
    def __init__(self):
        pass
```

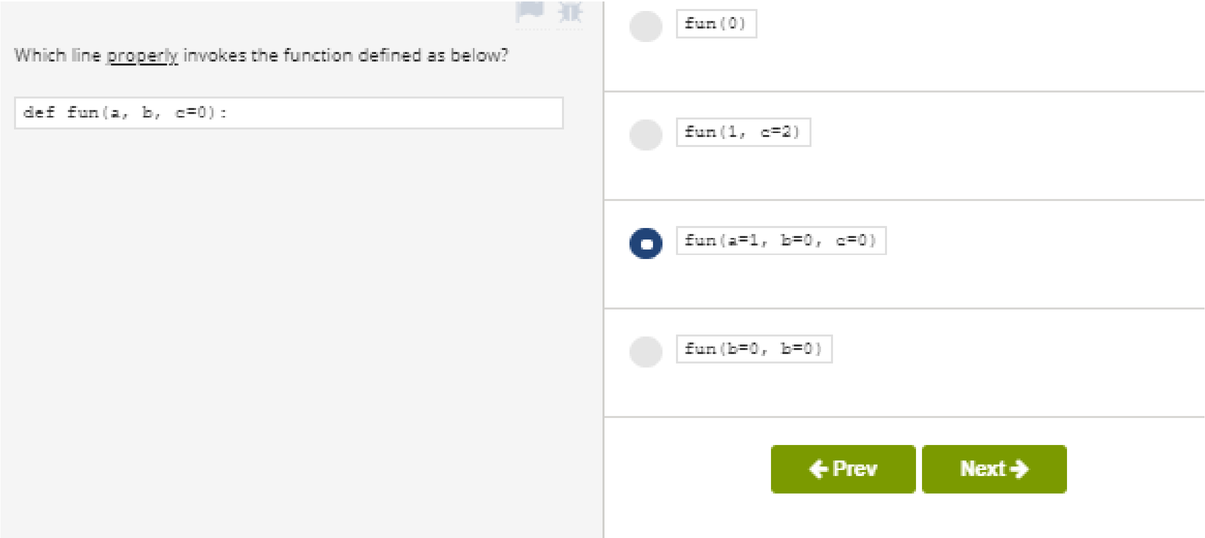
which one of the assignments is valid?

○ `object = Class(None)`

● `object = Class()`

○ `object = Class(1)`

○ `object = Class(1,2)`



Which line properly invokes the function defined as below?

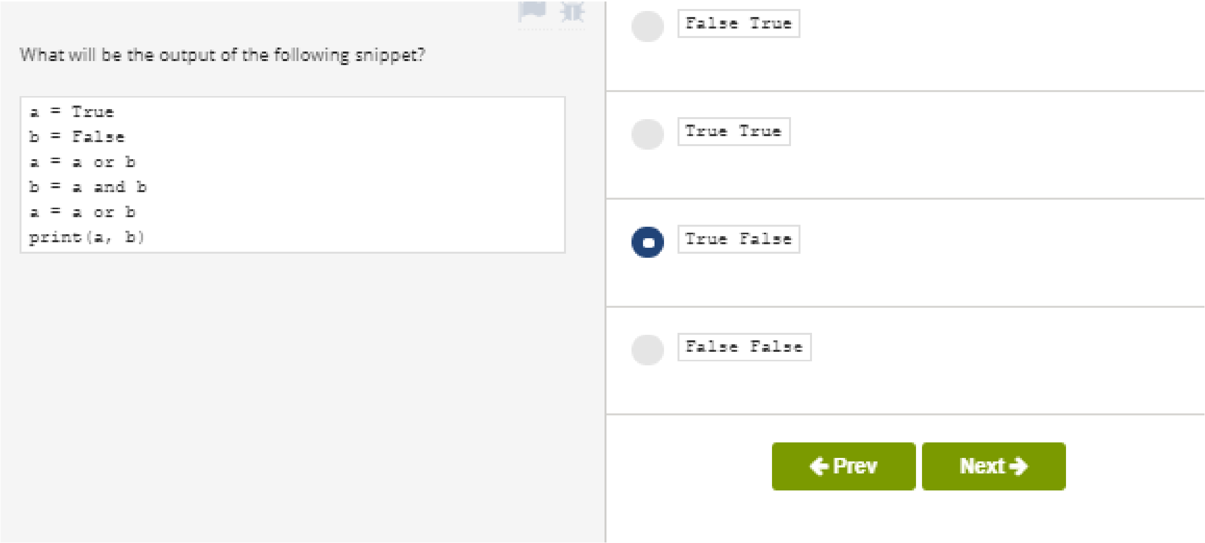
```
def fun(a, b, c=0):
```

○ `fun(0)`

○ `fun(1, c=2)`

● `fun(a=1, b=0, c=0)`

○ `fun(b=0, b=0)`



What will be the output of the following snippet?

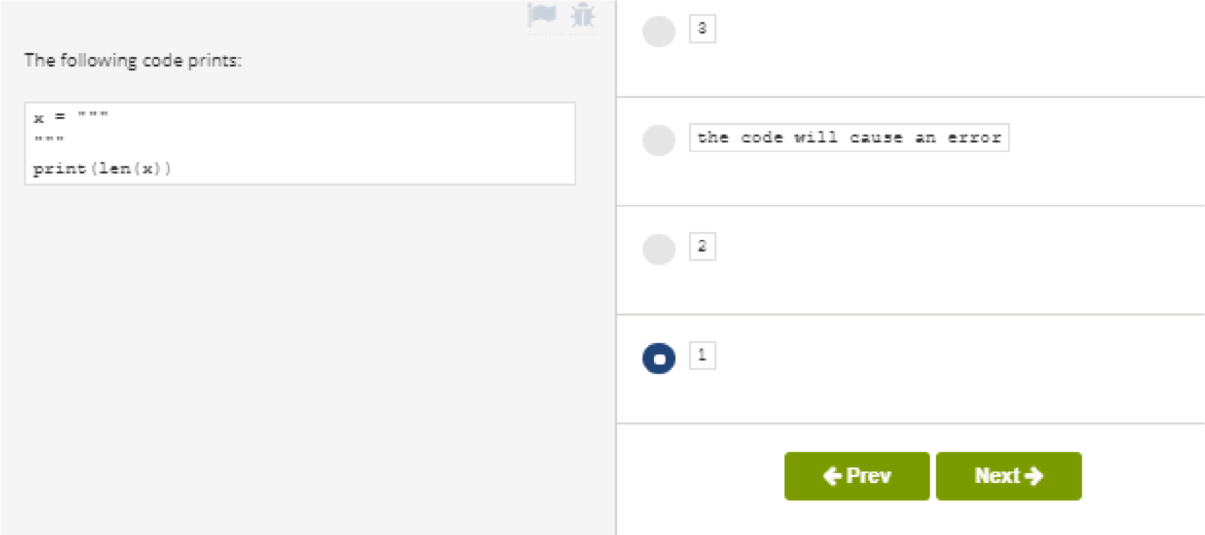
```
a = True
b = False
a = a or b
b = a and b
a = a or b
print(a, b)
```

○ `False True`

○ `True True`

● `True False`

○ `False False`



The following code prints:

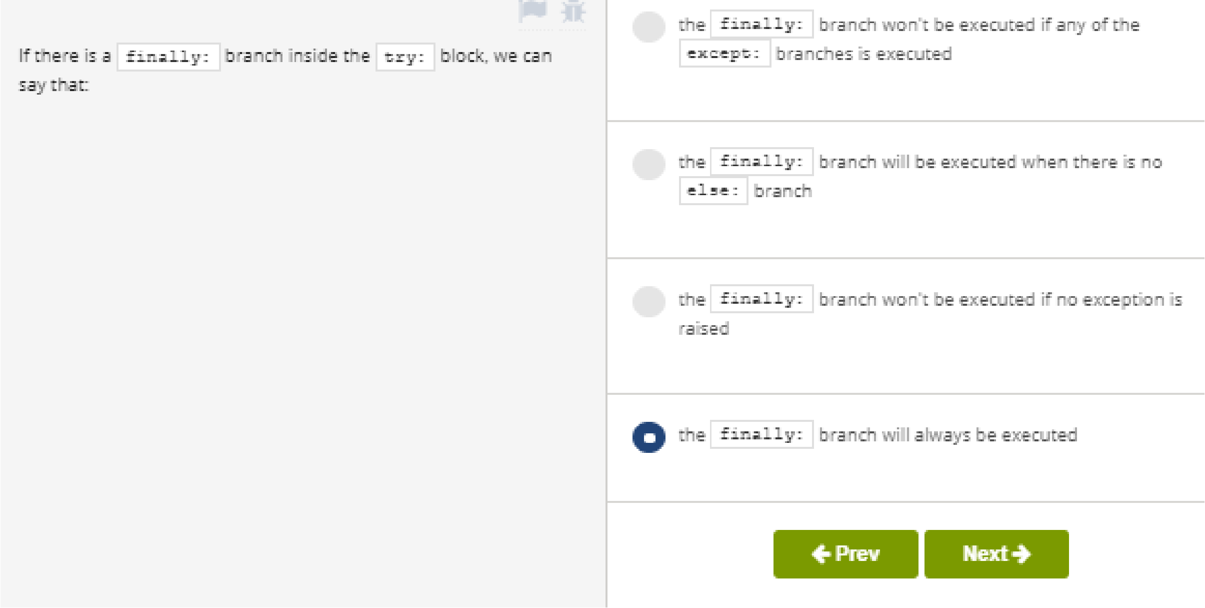
```
x = """
"""

print(len(x))
```

○ `3`

○ `the code will cause an error`

○ `2`

● `1`



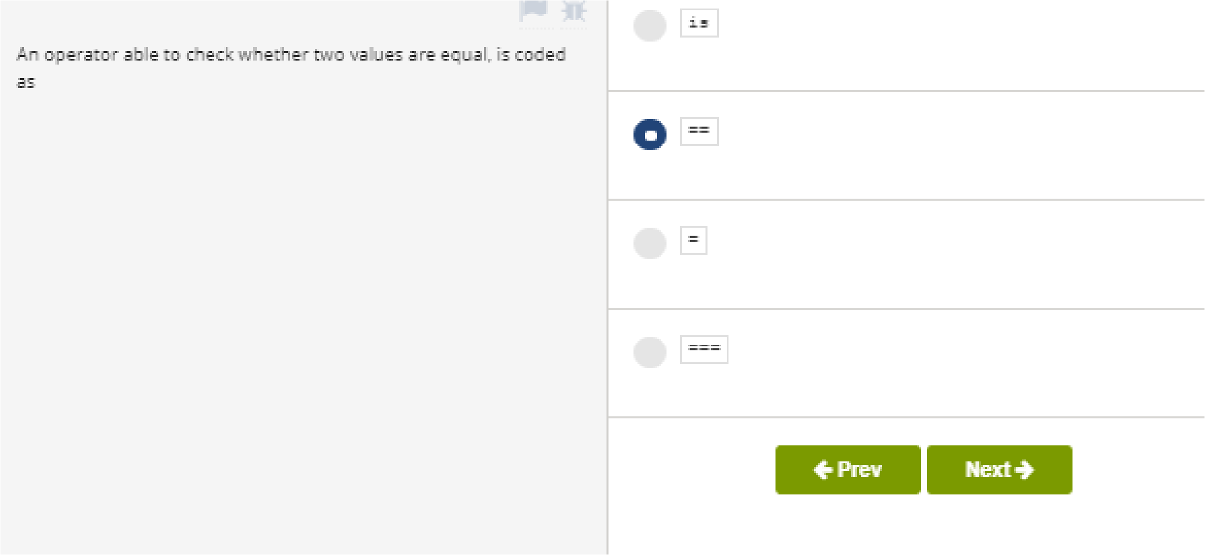
If there is a `finally:` branch inside the `try:` block, we can say that:

○ the `finally:` branch won't be executed if any of the `except:` branches is executed

○ the `finally:` branch will be executed when there is no `else:` branch

○ the `finally:` branch won't be executed if no exception is raised

● the `finally:` branch will always be executed



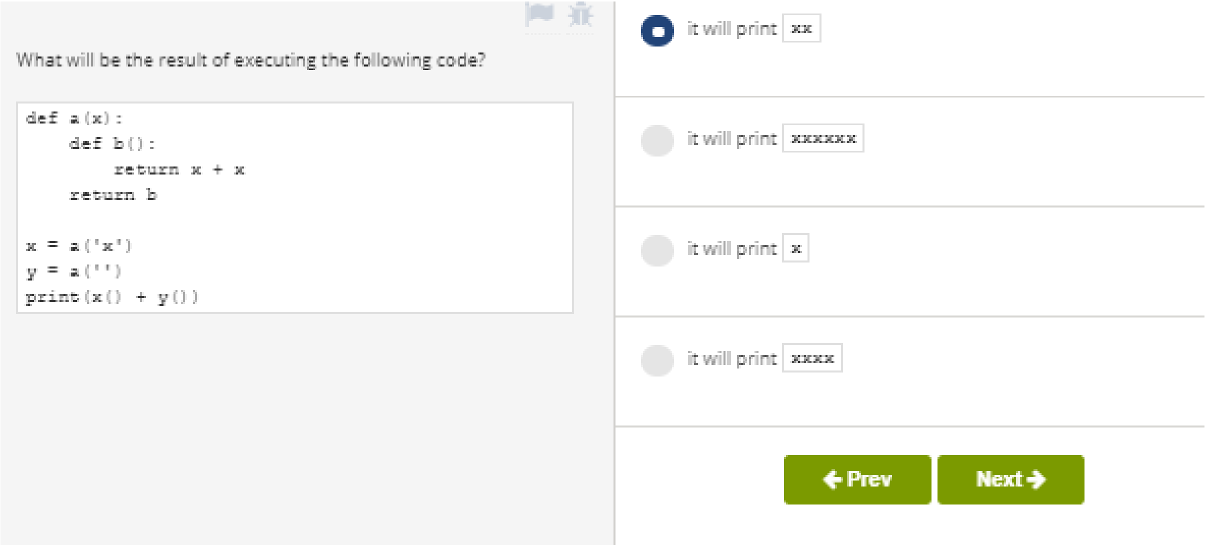
An operator able to check whether two values are equal, is coded as

○ `is`

● `==`

○ `=`

○ `===`



What will be the result of executing the following code?

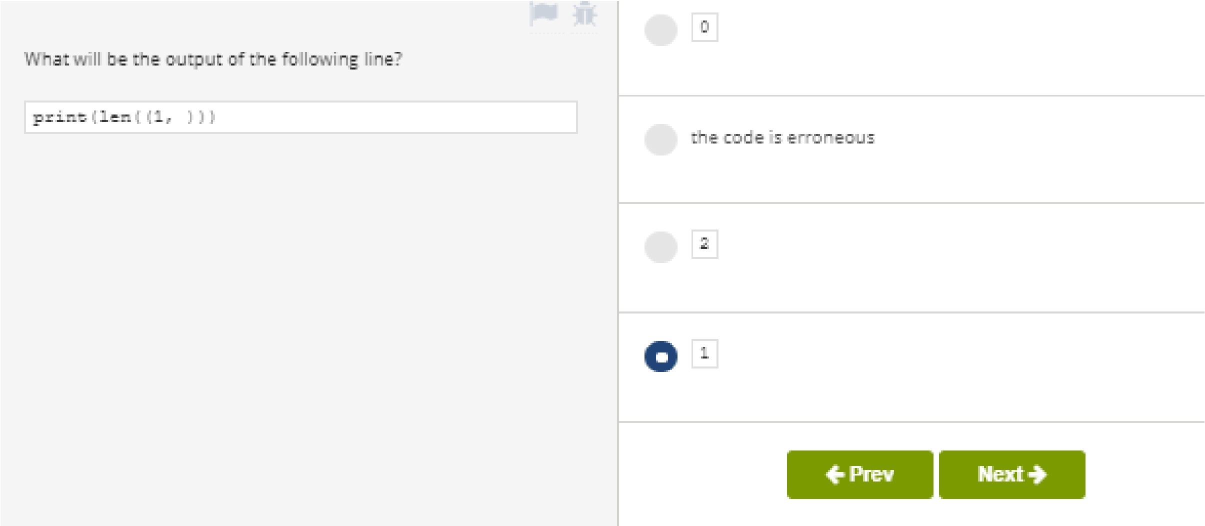
```
def a(x):
    def b():
        return x + x
    return b

x = a('x')
y = a('')
print(x() + y())
```

● it will print `xx`

○ it will print `xxxxxx`

○ it will print `x`

○ it will print `xxxx`



What will be the output of the following line?

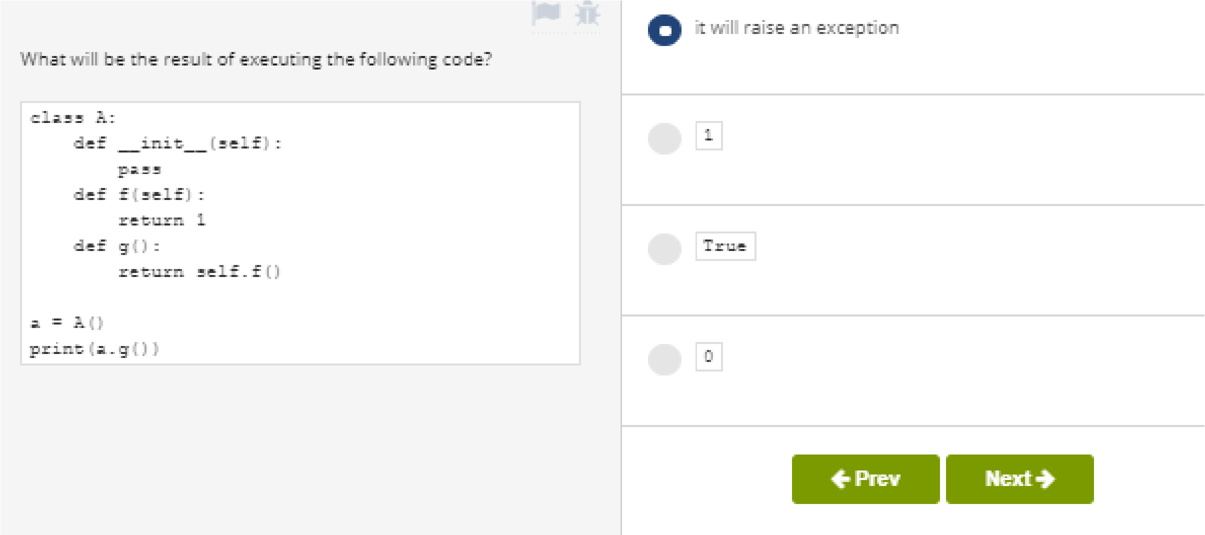
```
print(len((1, )))
```

○ `0`

○ the code is erroneous

○ `2`

● `1`



What will be the result of executing the following code?

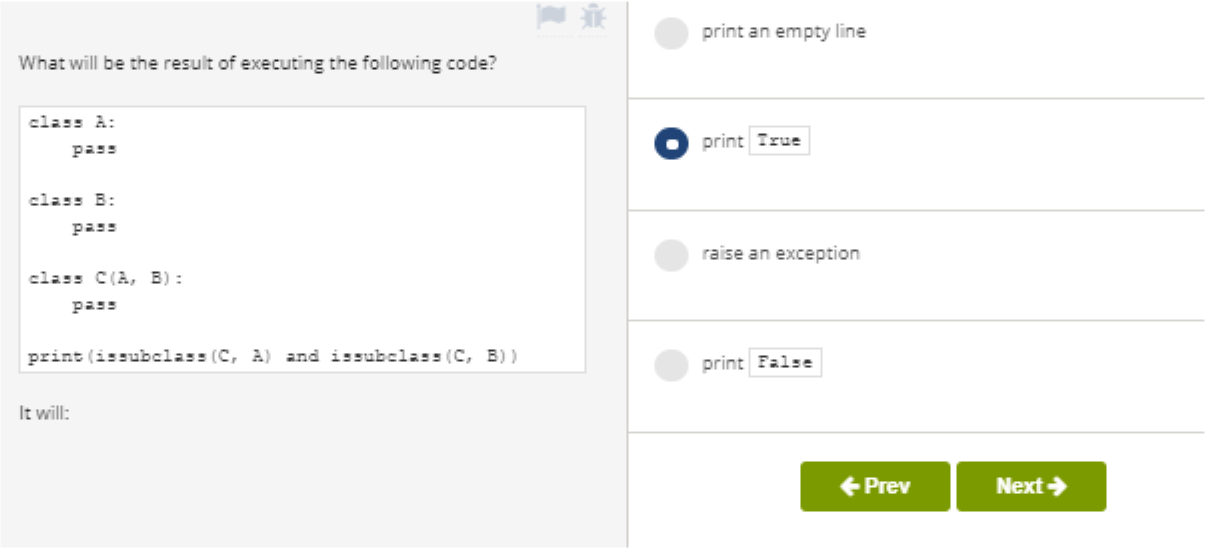
```
class A:
    def __init__(self):
        pass
    def f(self):
        return 1
    def g():
        return self.f()

a = A()
print(a.g())
```

● it will raise an exception

○ `1`

○ `True`

○ `0`



What will be the result of executing the following code?

```
class A:
    pass

class B:
    pass

class C(A, B):
    pass

print(issubclass(C, A) and issubclass(C, B))
```

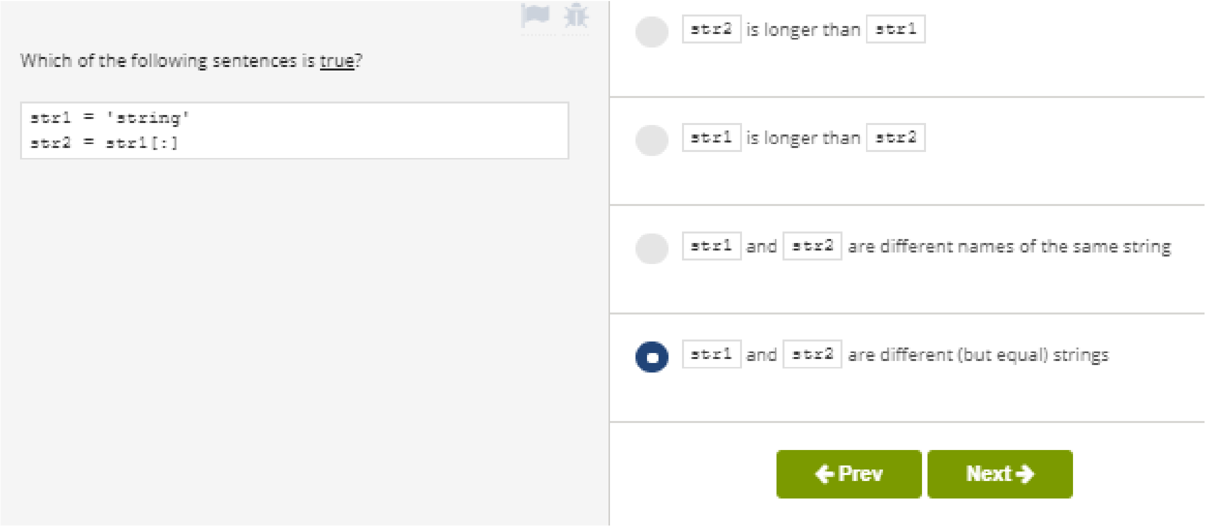
It will:

○ print an empty line

● print `True`

○ raise an exception

○ print `False`



Which of the following sentences is true?

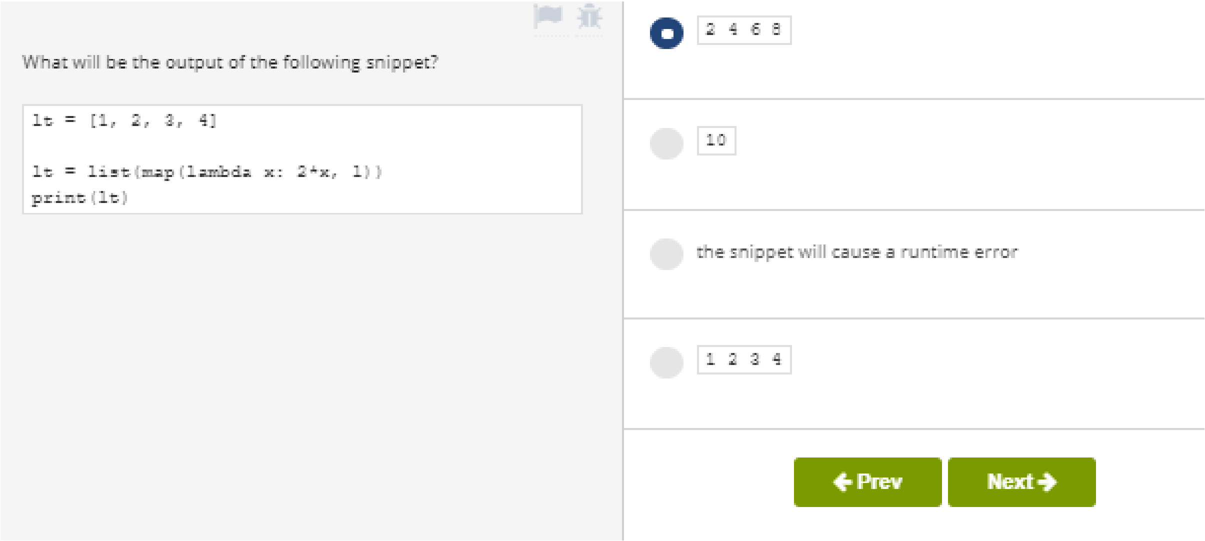
```
str1 = 'string'
str2 = str1[:]
```

○ `str2` is longer than `str1`

○ `str1` is longer than `str2`

○ `str1` and `str2` are different names of the same string

● `str1` and `str2` are different (but equal) strings



What will be the output of the following snippet?

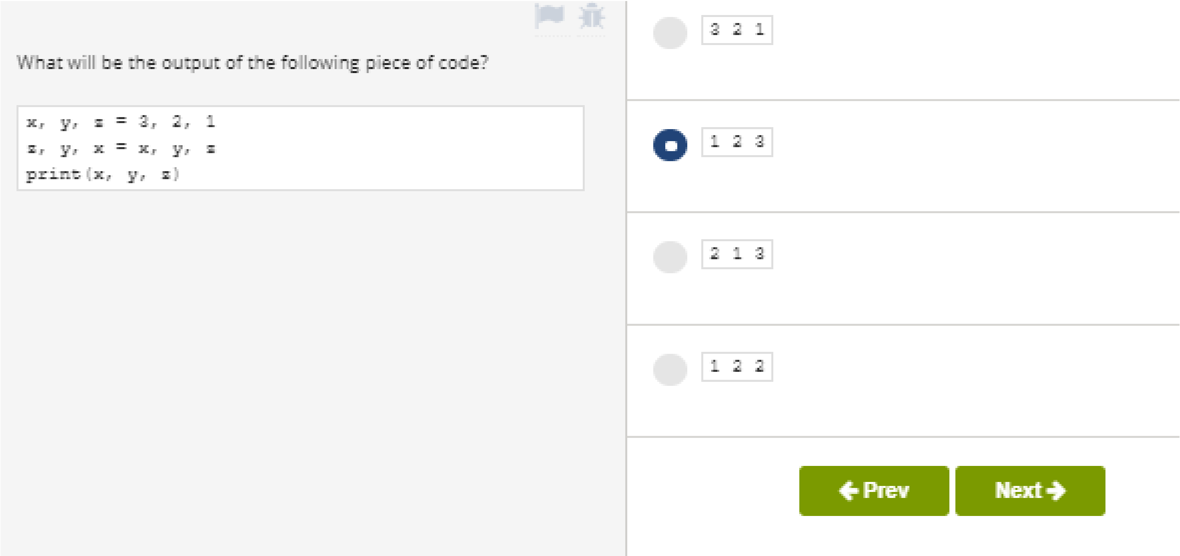
```
lt = [1, 2, 3, 4]

lt = list(map(lambda x: 2*x, 1))
print(lt)
```

● `2 4 6 8`

○ `10`

○ the snippet will cause a runtime error

○ `1 2 3 4`



What will be the output of the following piece of code?

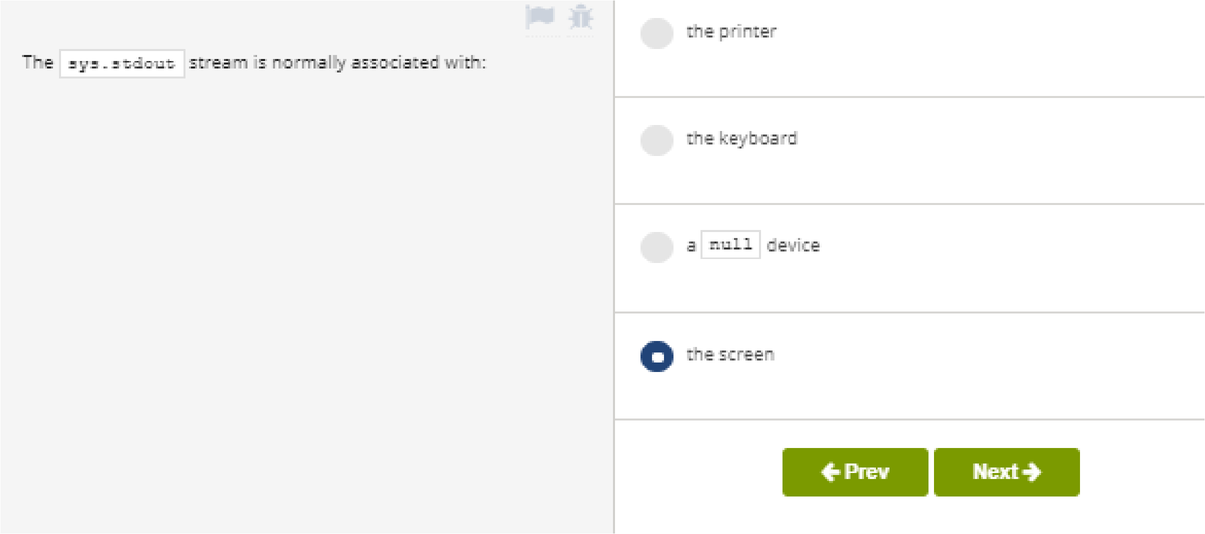
```
x, y, z = 3, 2, 1
z, y, x = x, y, z
print(x, y, z)
```

○ `3 2 1`

● `1 2 3`

○ `2 1 3`

○ `1 2 2`



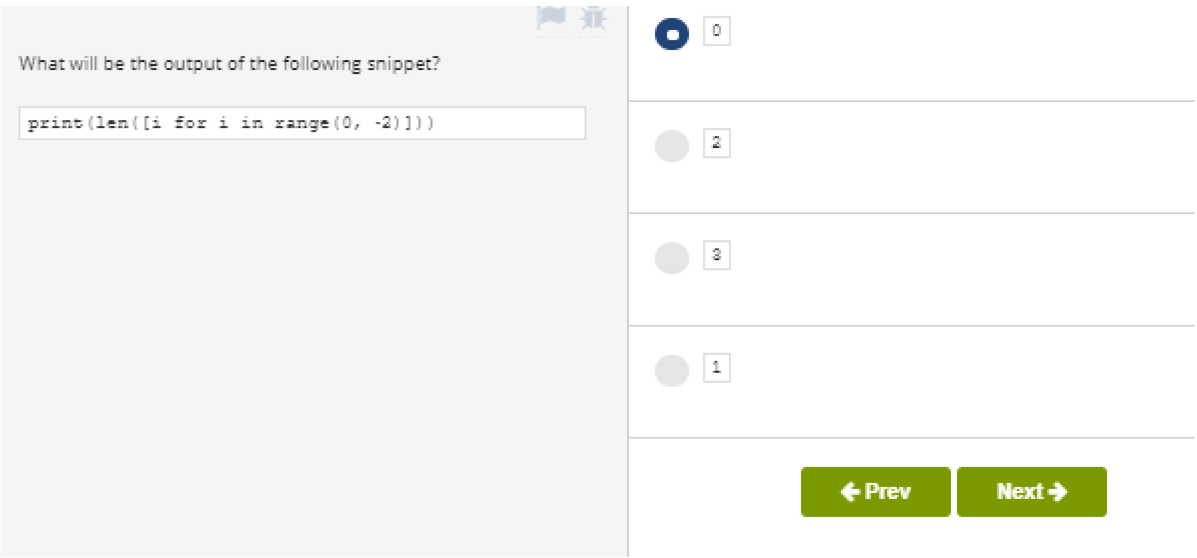
The `sys.stdout` stream is normally associated with:

○ the printer

○ the keyboard

○ a `null` device

● the screen



What will be the output of the following snippet?

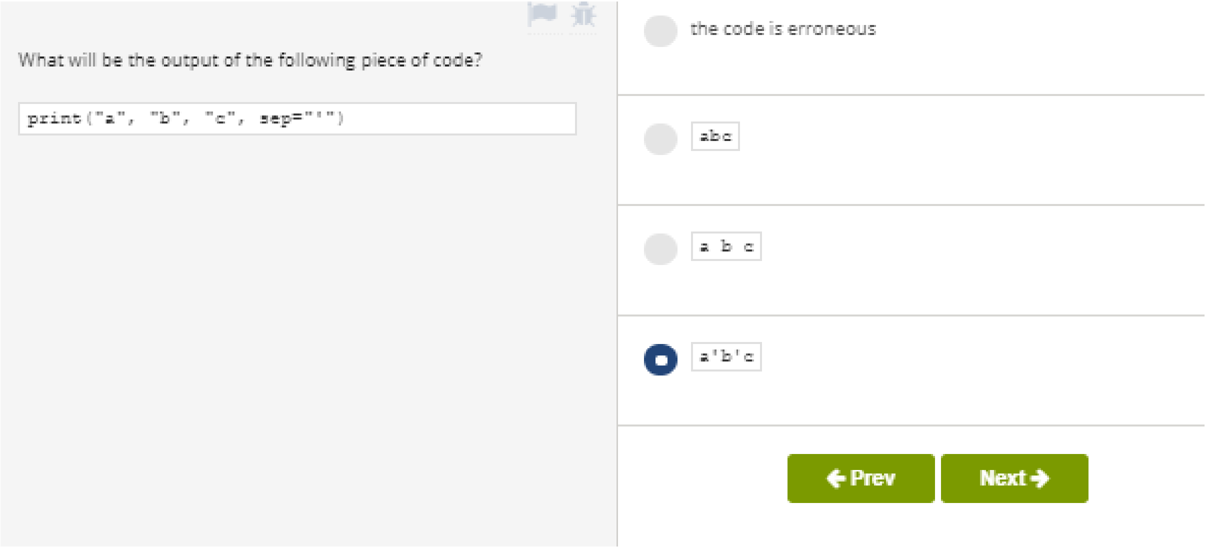
```
print(len([i for i in range(0, -2)]))
```

● 0

○ 2

○ 3

○ 1



What will be the output of the following piece of code?

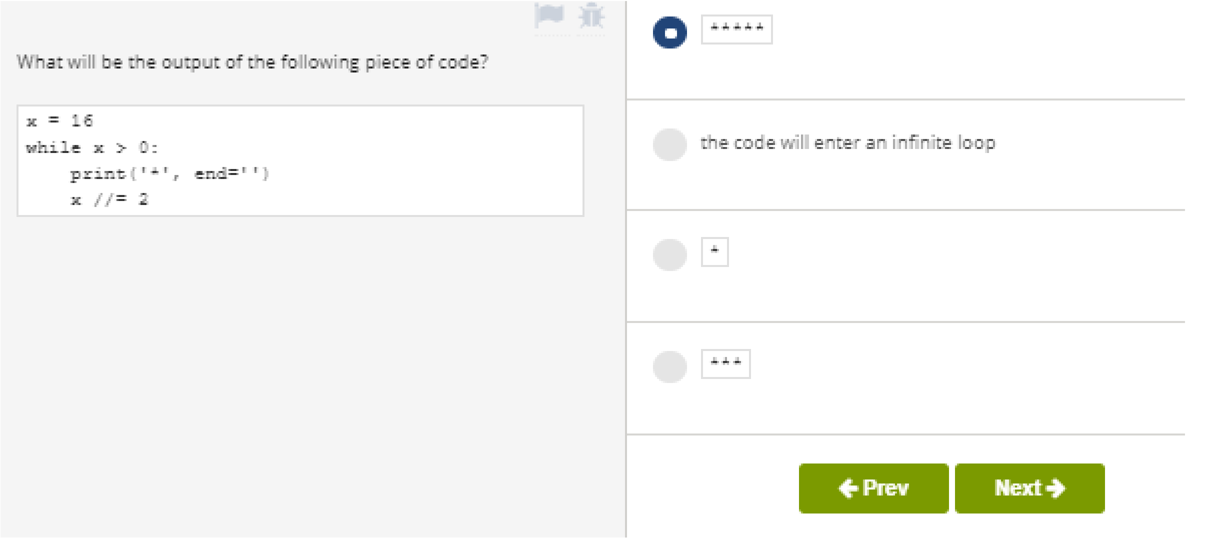
```
print("a", "b", "c", sep="'")
```

○ the code is erroneous

○ `abc`

○ `a b c`

● `a'b'c`



What will be the output of the following piece of code?

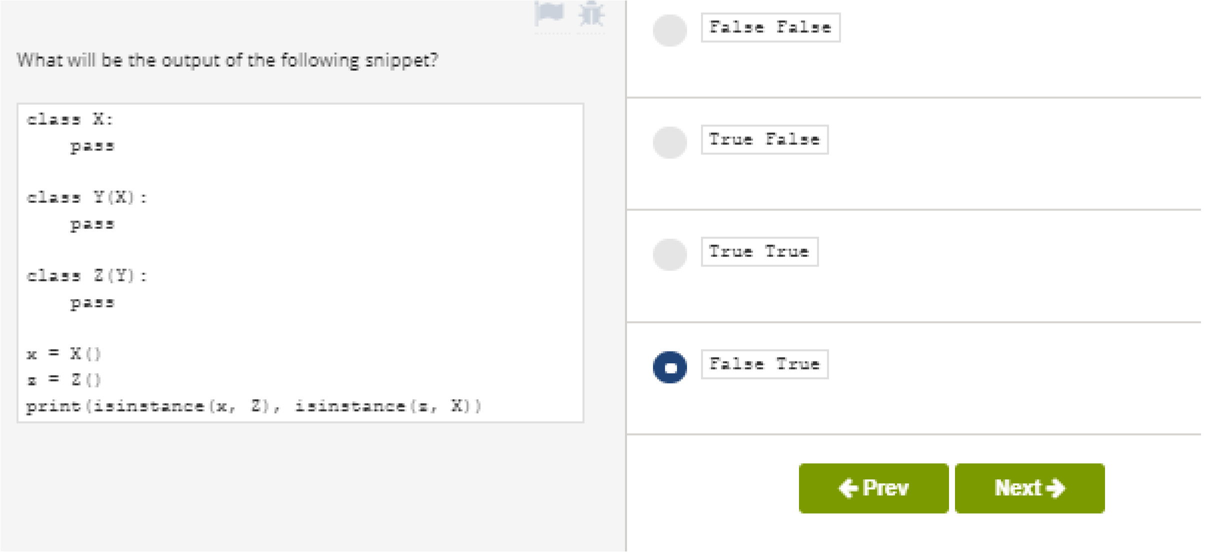
```
x = 16
while x > 0:
    print('*', end='')
    x //= 2
```

● `*****`

○ the code will enter an infinite loop

○ `*`

○ `***`



What will be the output of the following snippet?

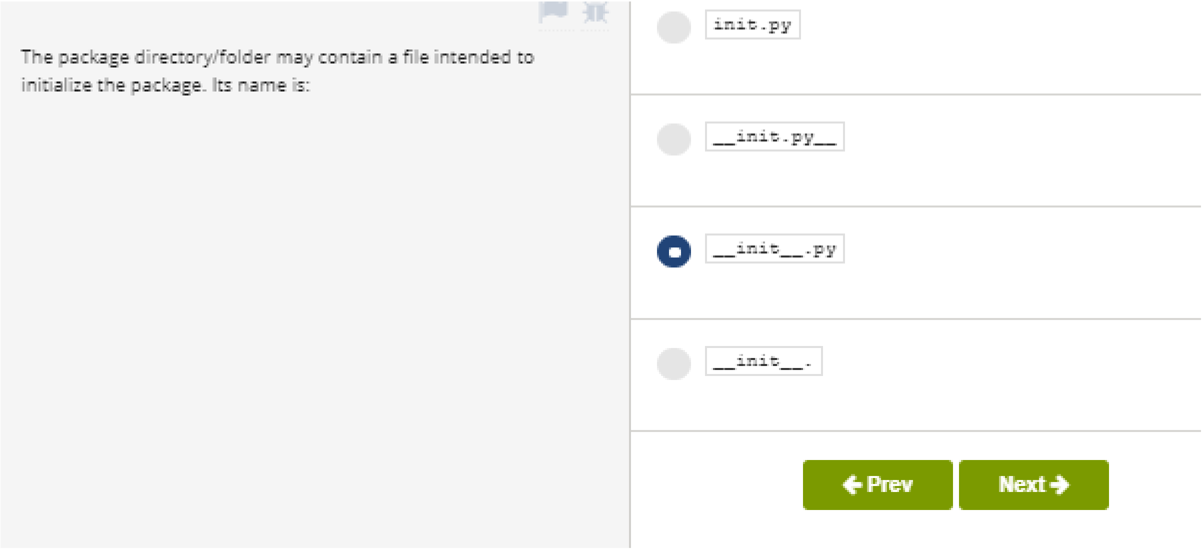
```
class X:
    pass

class Y(X):
    pass

class Z(Y):
    pass

x = X()
z = Z()
print(isinstance(x, Z), isinstance(z, X))
```

○ `False False`

○ `True False`

○ `True True`

● `False True`



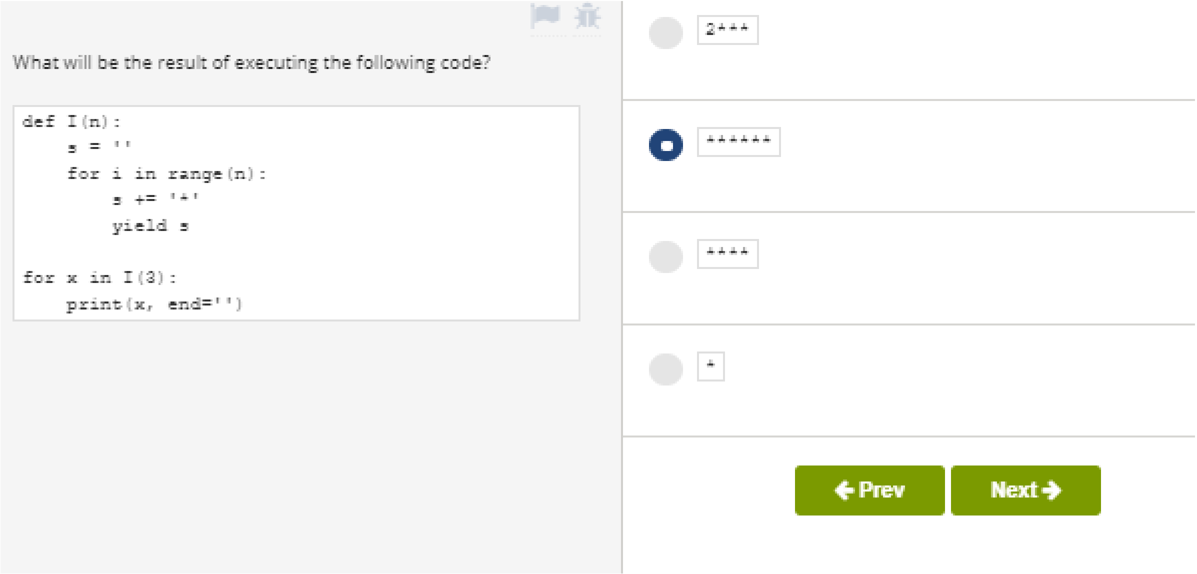
The package directory/folder may contain a file intended to initialize the package. Its name is:

○ `init.py`

○ `__init.py__`

● `__init__.py`

○ `__init__.`



What will be the result of executing the following code?

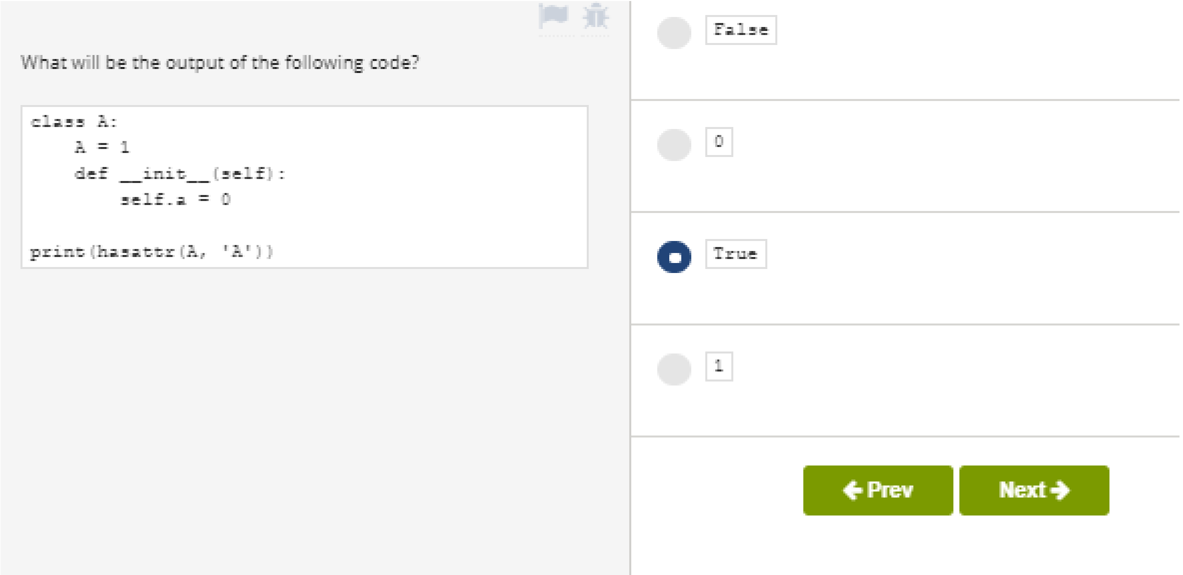
```
def I(n):
    s = ''
    for i in range(n):
        s += '*'
        yield s

for x in I(3):
    print(x, end='')
```

○ `2***`

● `******`

○ `****`

○ `*`



What will be the output of the following code?

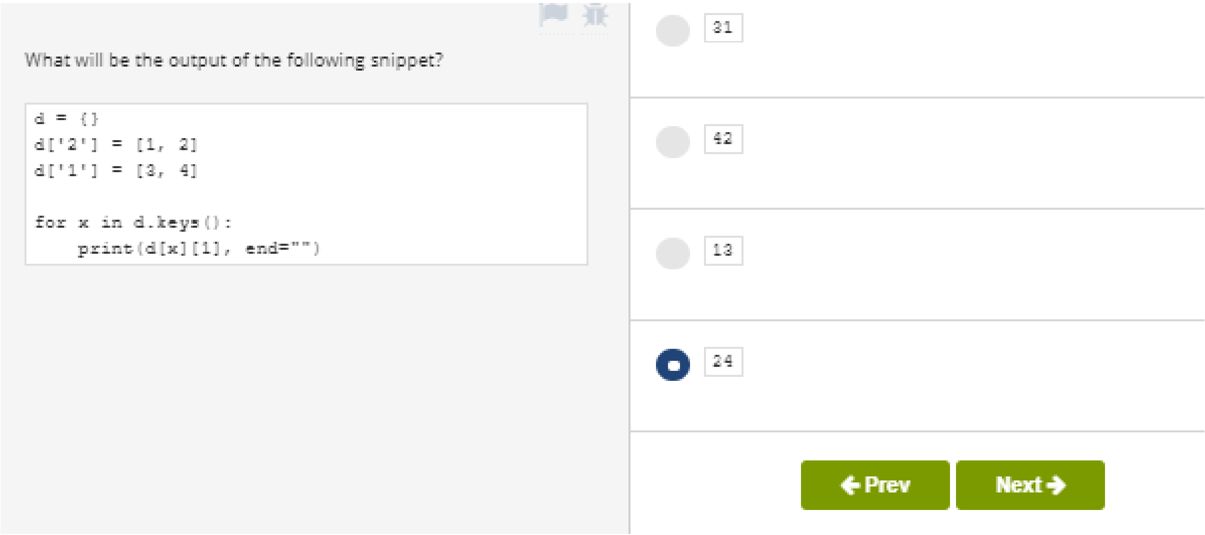
```
class A:
    A = 1
    def __init__(self):
        self.a = 0

print(hasattr(A, 'A'))
```

○ `False`

○ `0`

● `True`

○ `1`



What will be the output of the following snippet?

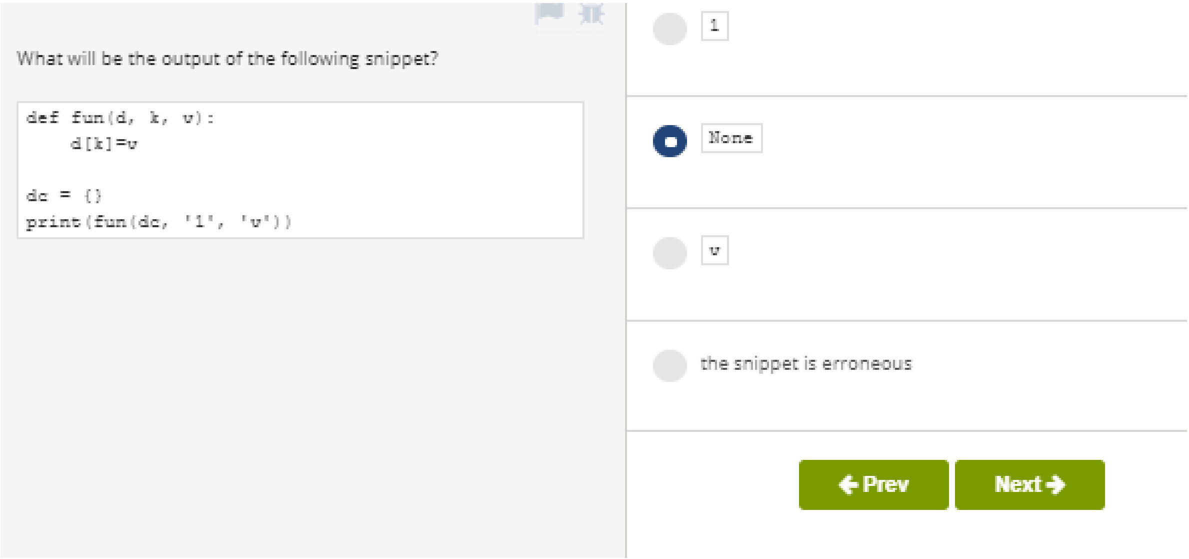
```
d = {}
d['2'] = [1, 2]
d['1'] = [3, 4]

for x in d.keys():
    print(d[x][1], end="")
```

○ 31

○ 42

○ 13

● 24



What will be the output of the following snippet?

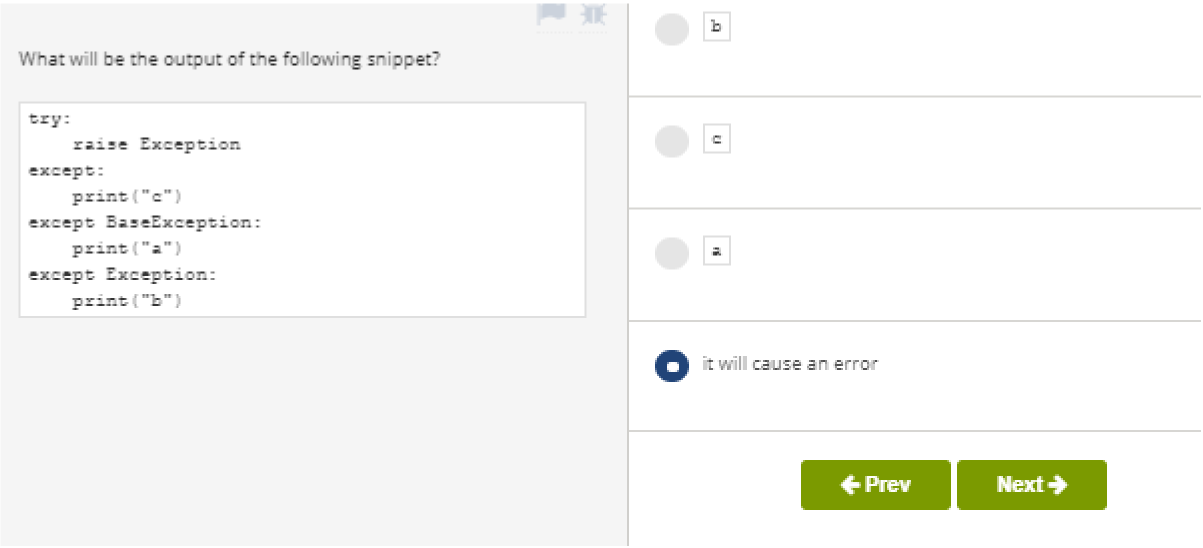
```
def fun(d, k, v):
    d[k]=v

dc = {}
print(fun(dc, '1', 'v'))
```

○ 1

● None

○ v

○ the snippet is erroneous



What will be the output of the following snippet?

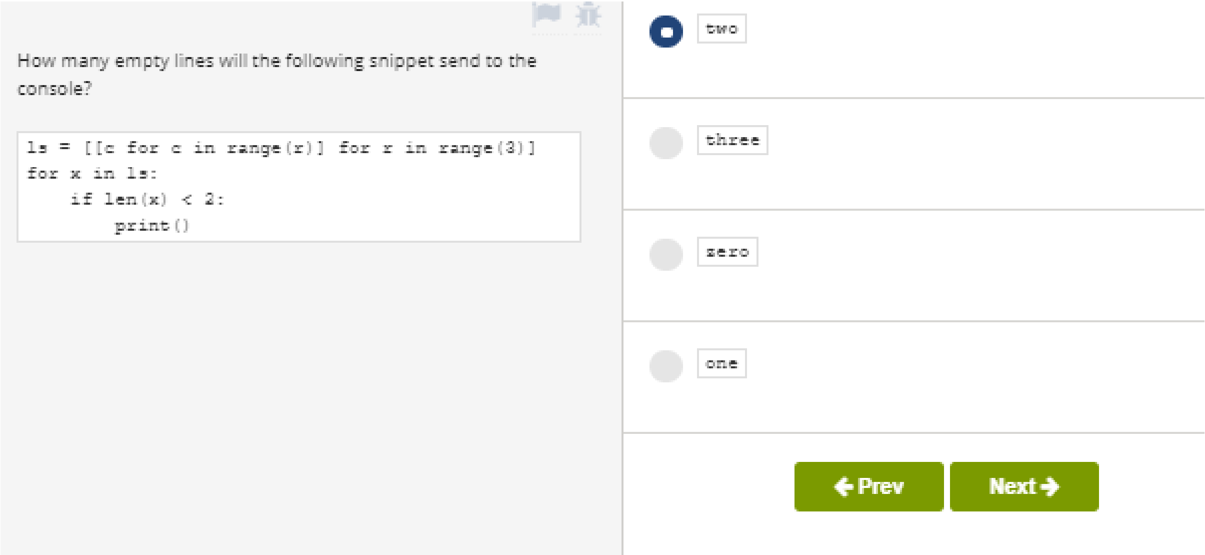
```
try:
    raise Exception
except:
    print("c")
except BaseException:
    print("a")
except Exception:
    print("b")
```

○ b

○ c

○ a

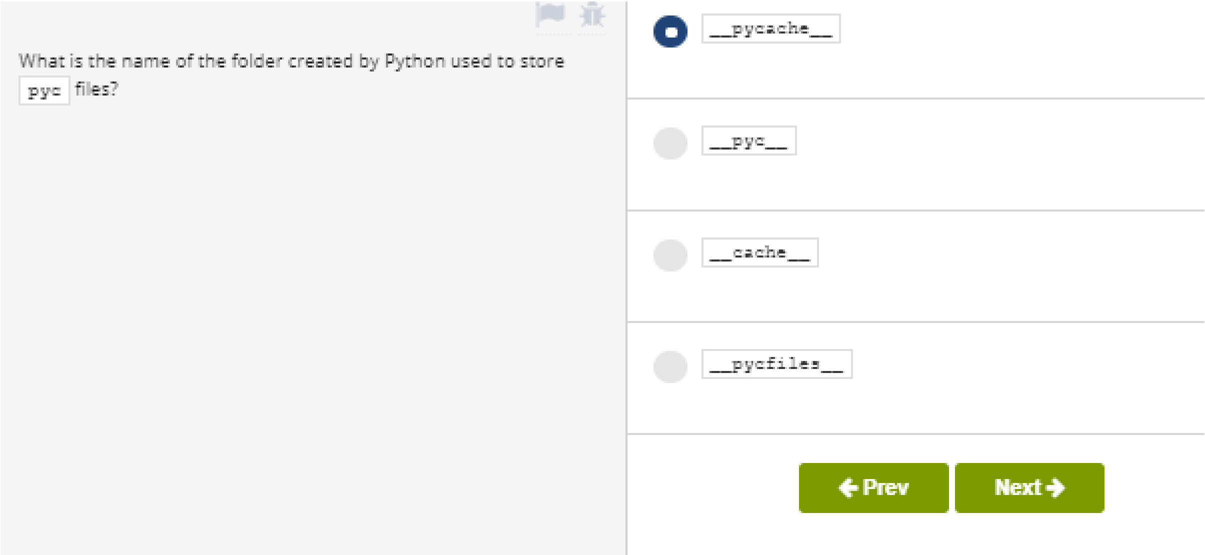
● it will cause an error



How many empty lines will the following snippet send to the console?

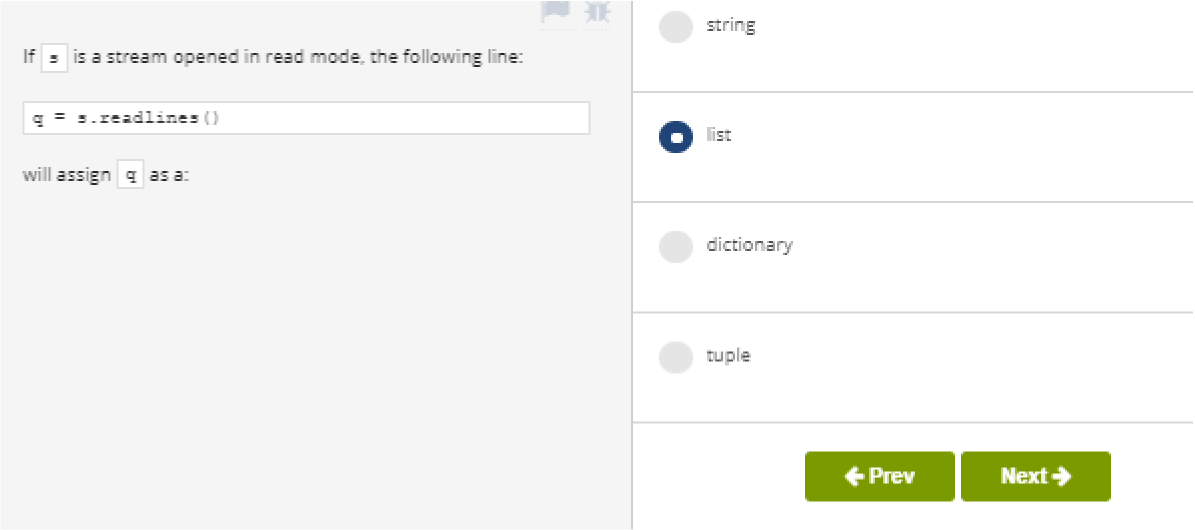
```
ls = [[c for c in range(r)] for r in range(3)]
for x in ls:
    if len(x) < 2:
        print()
```

- ● `two`
- ○ `three`
- ○ `zero`
- ○ `one`



What is the name of the folder created by Python used to store `pyc` files?

- ● `__pycache__`
- ○ `__pyc__`
- ○ `__cache__`
- ○ `__pycfiles__`

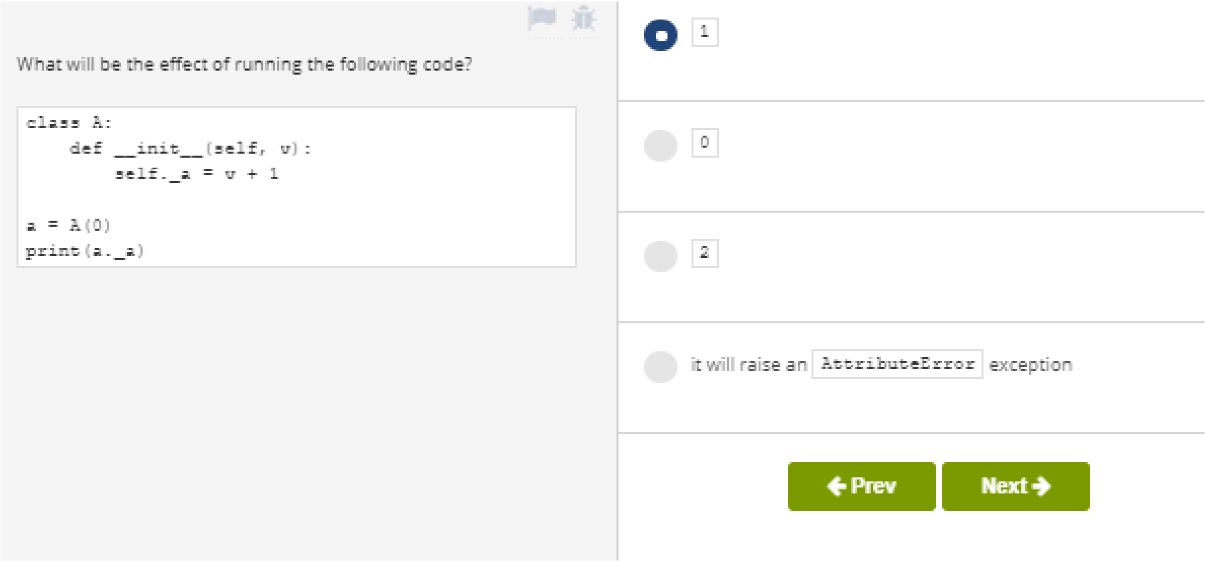


If `s` is a stream opened in read mode, the following line:

```
q = s.readlines()
```

will assign `q` as a:

- ○ string
- ● list
- ○ dictionary
- ○ tuple



What will be the effect of running the following code?

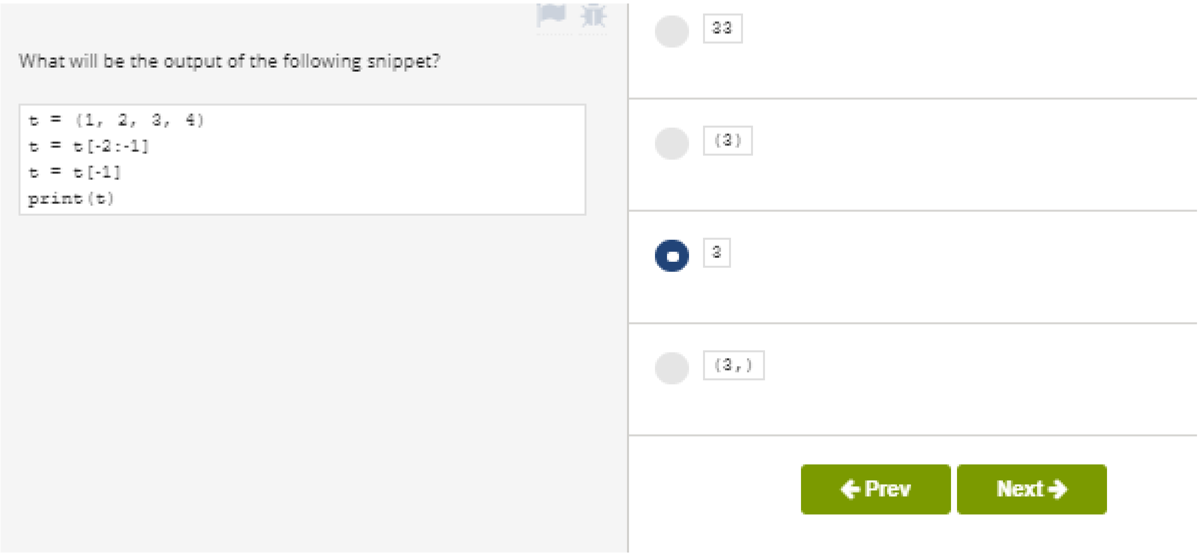
```
class A:
    def __init__(self, v):
        self._a = v + 1

a = A(0)
print(a._a)
```

● `1`

○ `0`

○ `2`

○ it will raise an `AttributeError` exception



What will be the output of the following snippet?

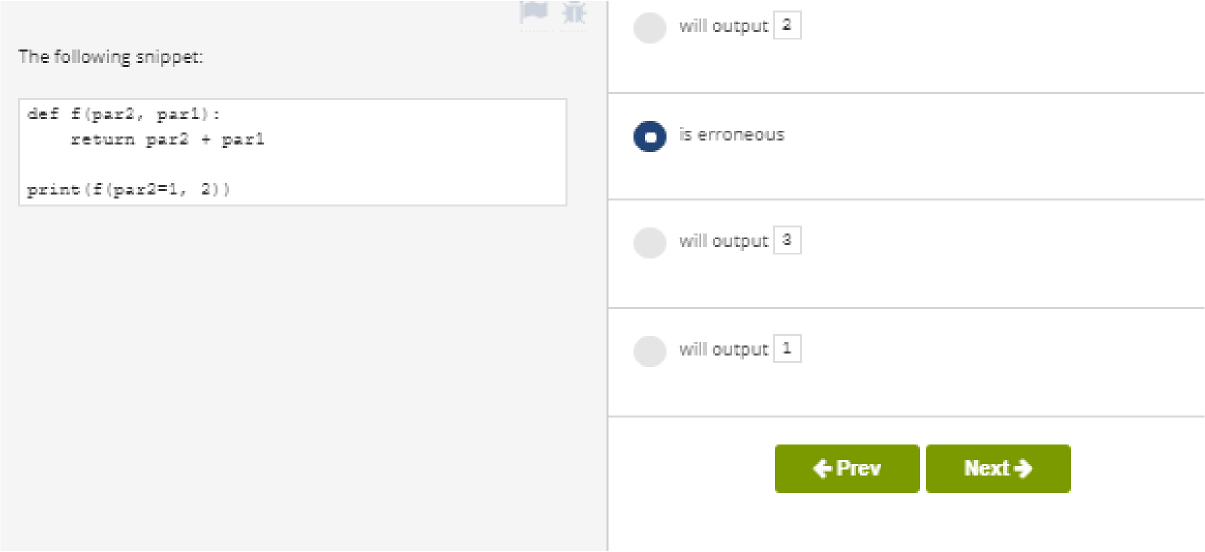
```
t = (1, 2, 3, 4)
t = t[-2:-1]
t = t[-1]
print(t)
```

○ `33`

○ `(3)`

● `3`

○ `(3,)`



The following snippet:

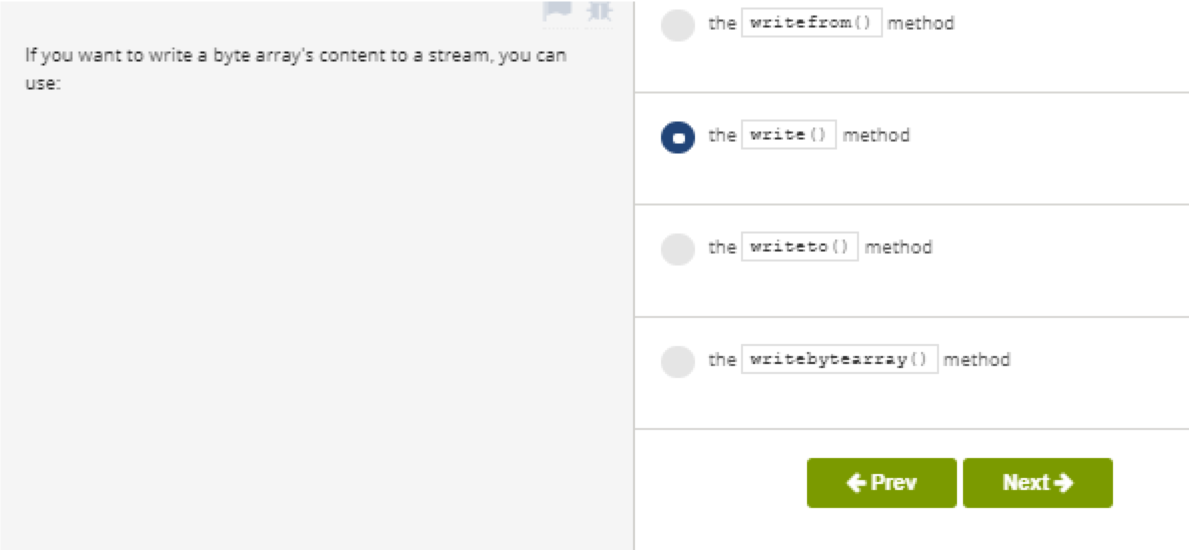
```
def f(par2, par1):
    return par2 + par1

print(f(par2=1, 2))
```

○ will output `2`

● is erroneous

○ will output `3`

○ will output `1`



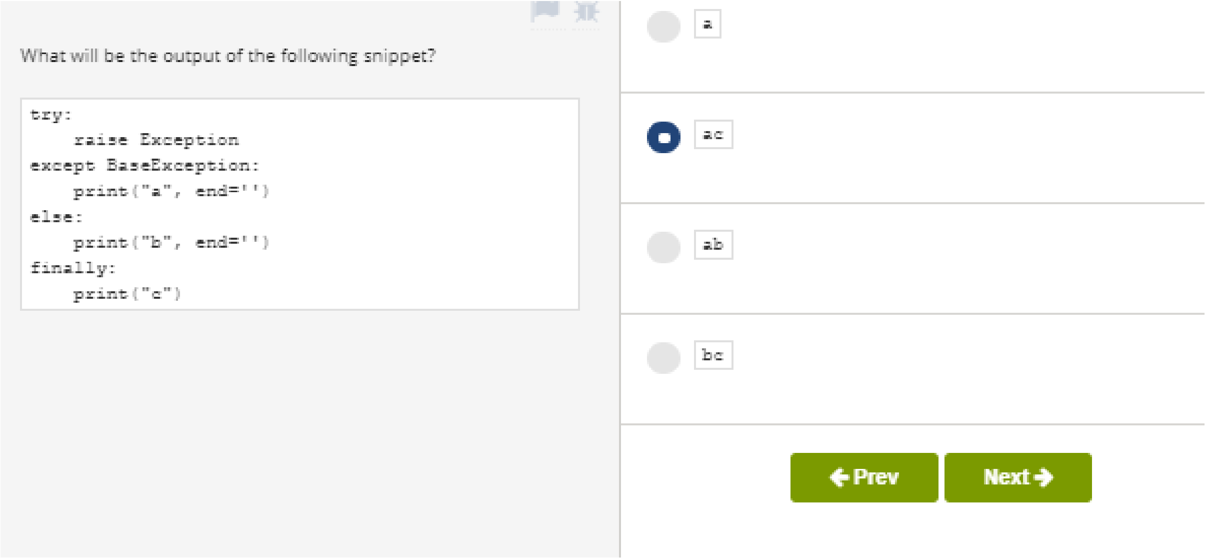
If you want to write a byte array's content to a stream, you can use:

○ the `writefrom()` method

● the `write()` method

○ the `writeto()` method

○ the `writebytearray()` method



What will be the output of the following snippet?

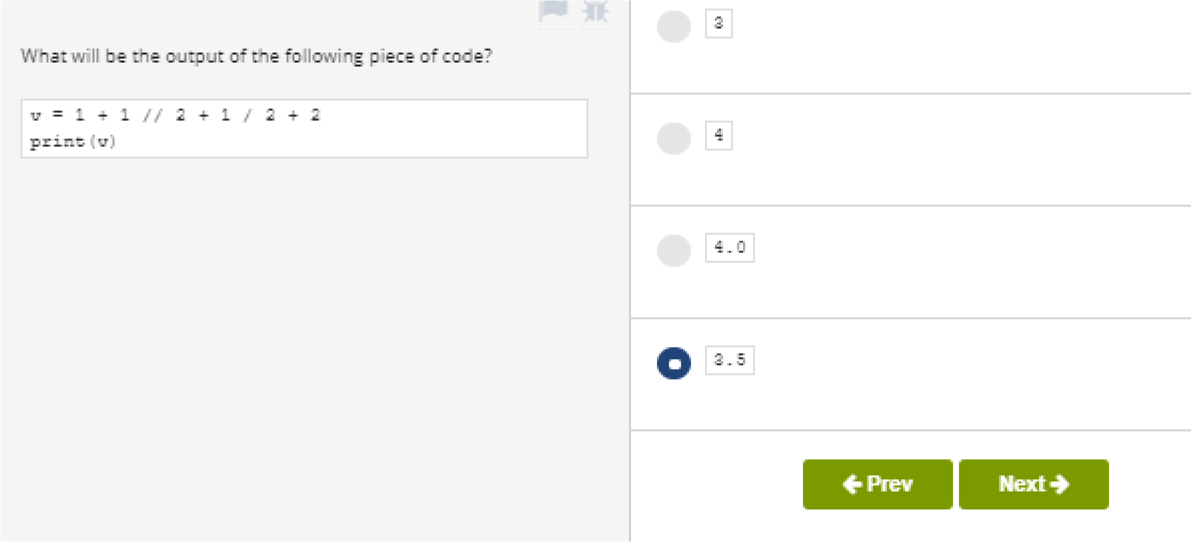
```
try:
    raise Exception
except BaseException:
    print("a", end='')
else:
    print("b", end='')
finally:
    print("c")
```

○ `a`

● `ac`

○ `ab`

○ `bc`



What will be the output of the following piece of code?

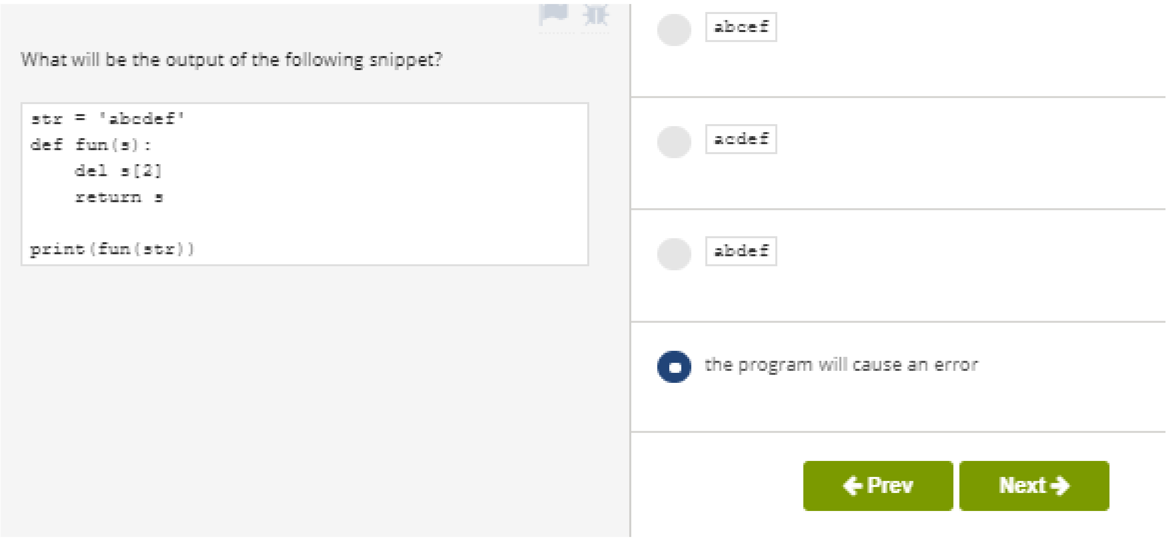
```
v = 1 + 1 // 2 + 1 / 2 + 2
print(v)
```

○ `3`

○ `4`

○ `4.0`

● `3.5`



What will be the output of the following snippet?

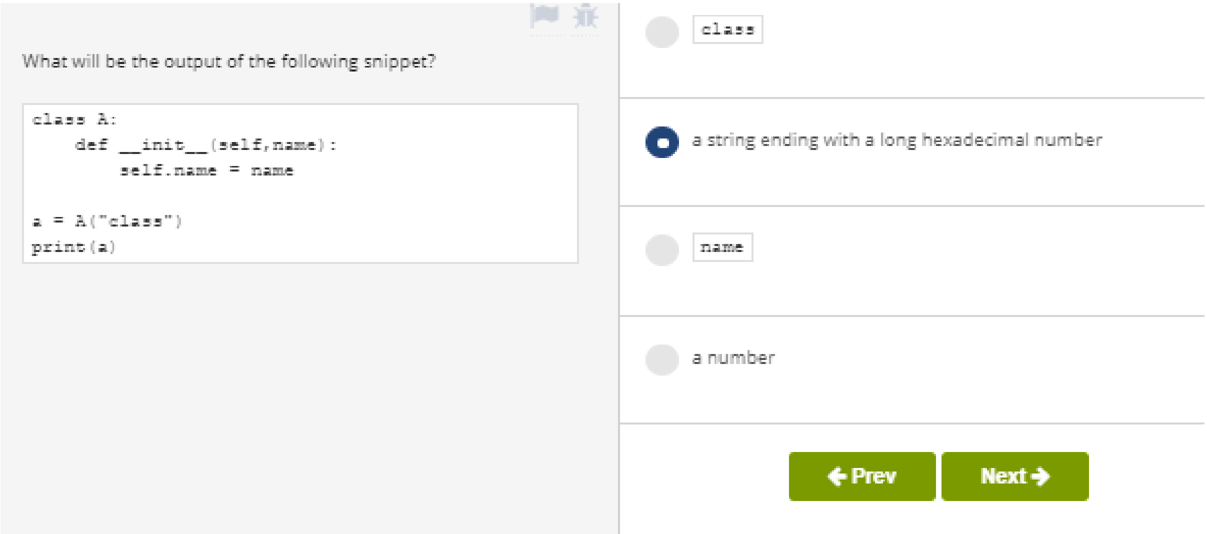
```
str = 'abcdef'
def fun(s):
    del s[2]
    return s

print(fun(str))
```

○ `abcef`

○ `acdef`

○ `abdef`

● the program will cause an error



What will be the output of the following snippet?

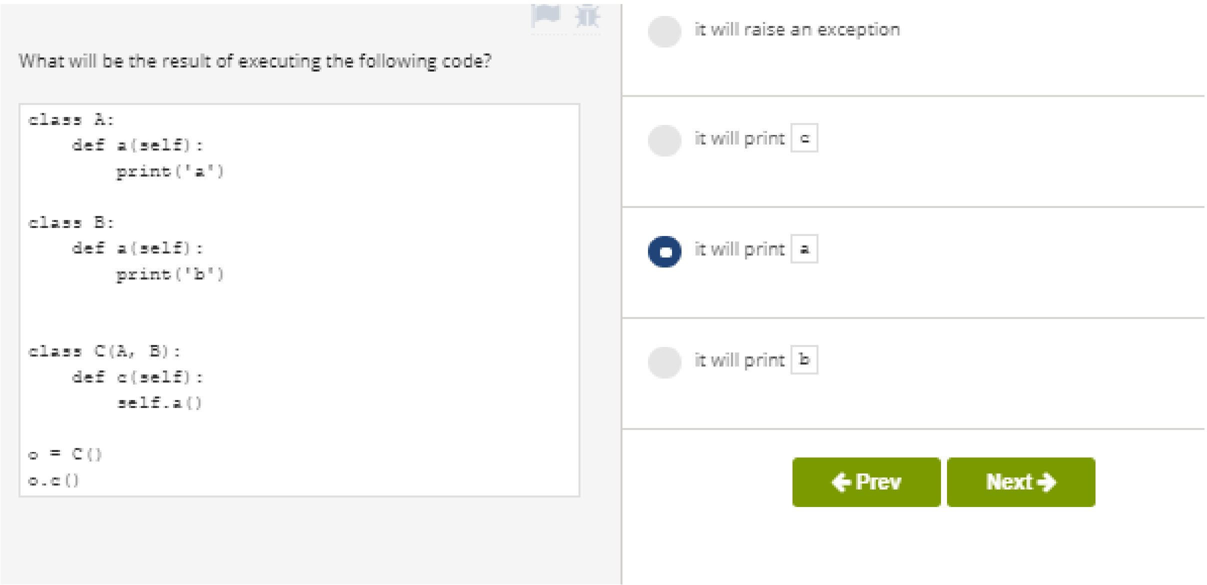
```
class A:
    def __init__(self,name):
        self.name = name

a = A("class")
print(a)
```

○ `class`

● a string ending with a long hexadecimal number

○ `name`

○ a number



What will be the result of executing the following code?

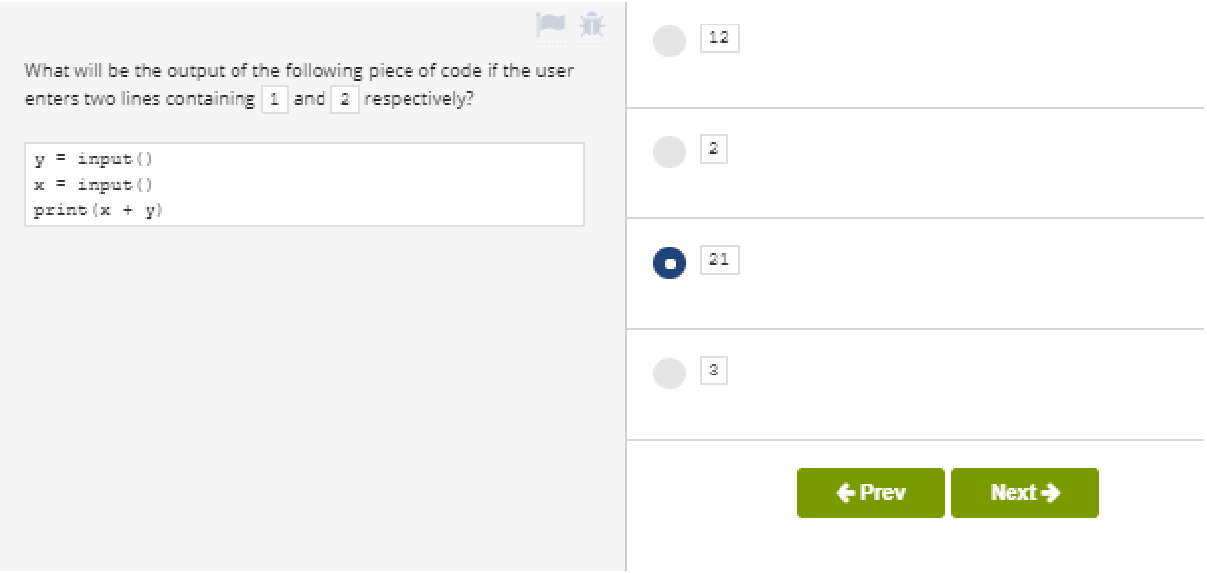
```
class A:
    def a(self):
        print('a')

class B:
    def a(self):
        print('b')


class C(A, B):
    def c(self):
        self.a()

o = C()
o.c()
```

○ it will raise an exception

○ it will print `c`

● it will print `a`

○ it will print `b`



What will be the output of the following piece of code if the user enters two lines containing `1` and `2` respectively?

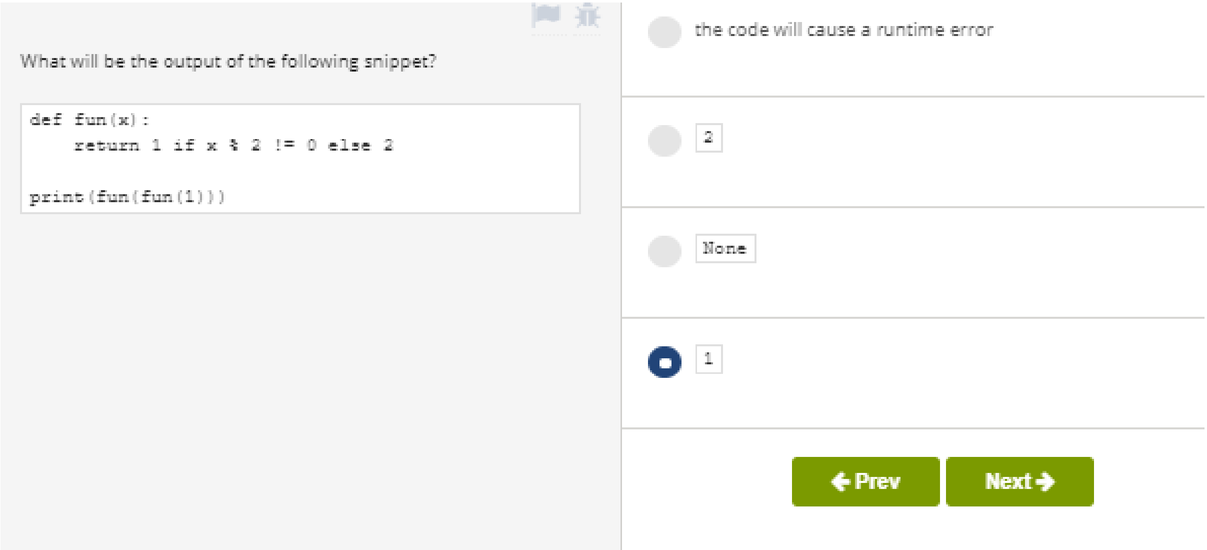
```
y = input()
x = input()
print(x + y)
```

○ `12`

○ `2`

● `21`

○ `3`



What will be the output of the following snippet?

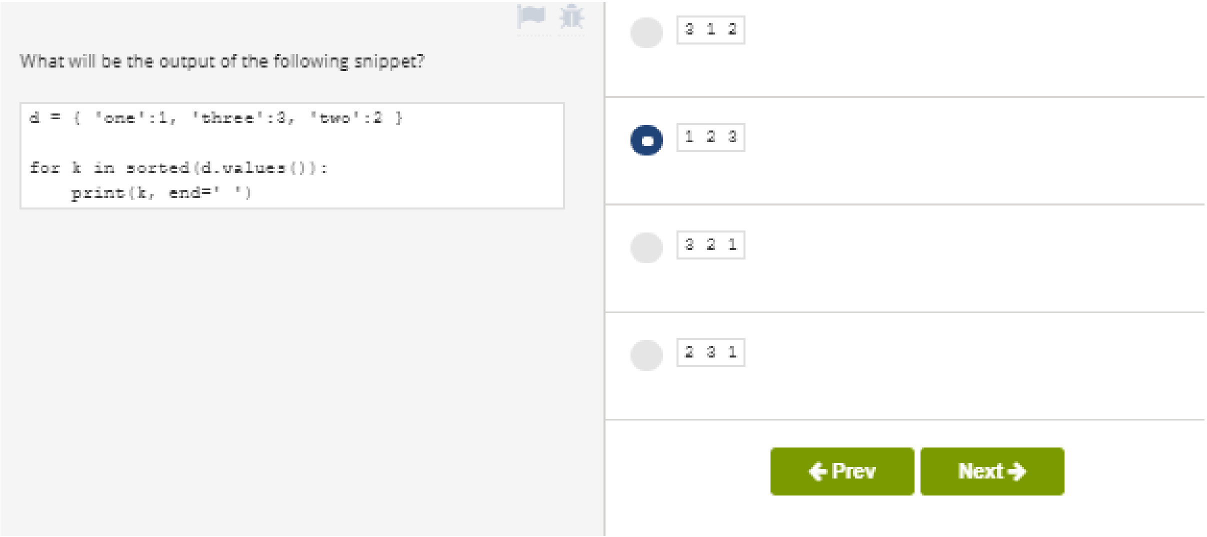
```
def fun(x):
    return 1 if x % 2 != 0 else 2

print(fun(fun(1)))
```

○ the code will cause a runtime error

○ `2`

○ `None`

● `1`



What will be the output of the following snippet?

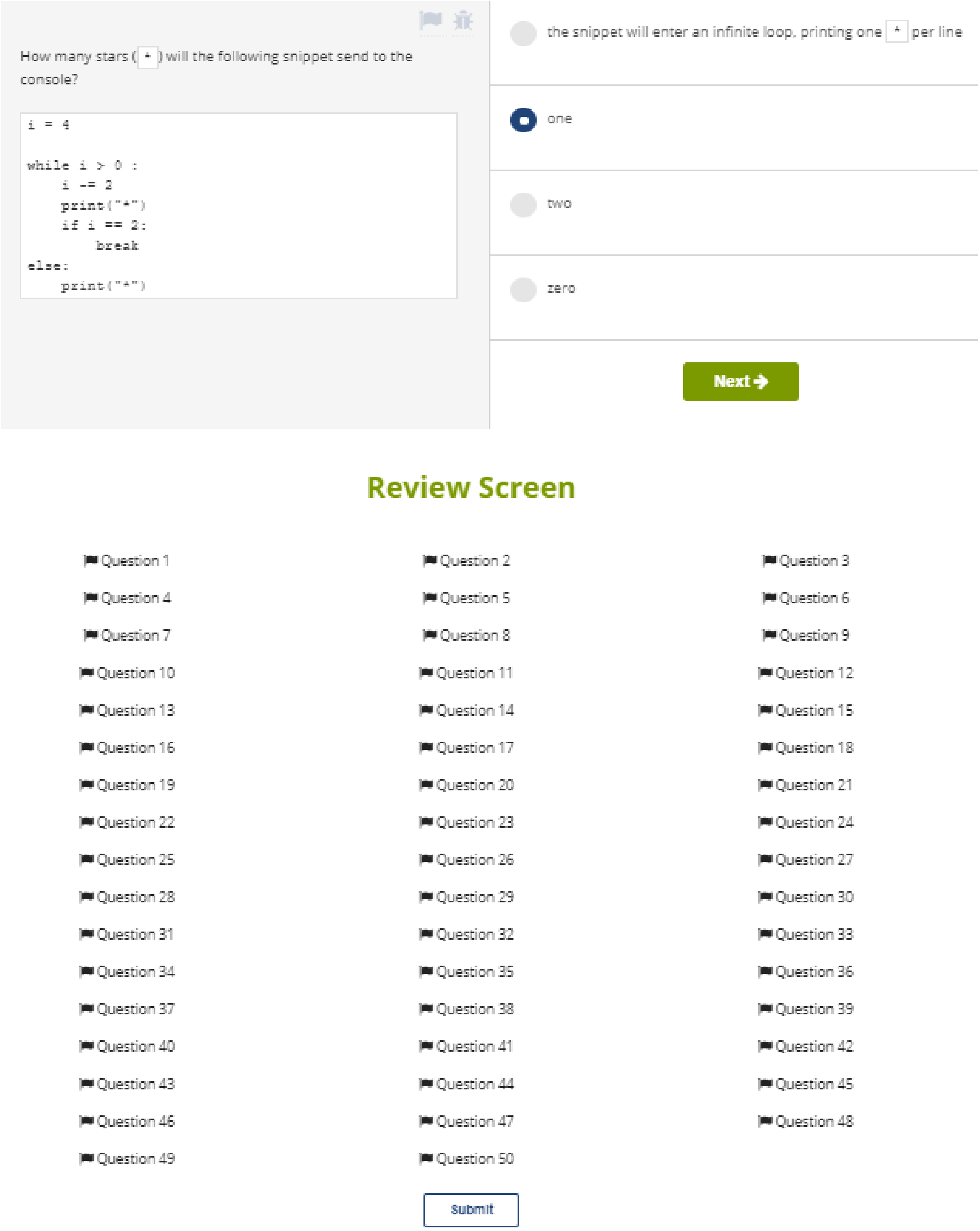
```
d = { 'one':1, 'three':3, 'two':2 }

for k in sorted(d.values()):
    print(k, end=' ')
```

○ `3 1 2`

● `1 2 3`

○ `3 2 1`

○ `2 3 1`



How many stars (*) will the following snippet send to the console?

```
i = 4

while i > 0 :
    i -= 2
    print("*")
    if i == 2:
        break
else:
    print("*")
```

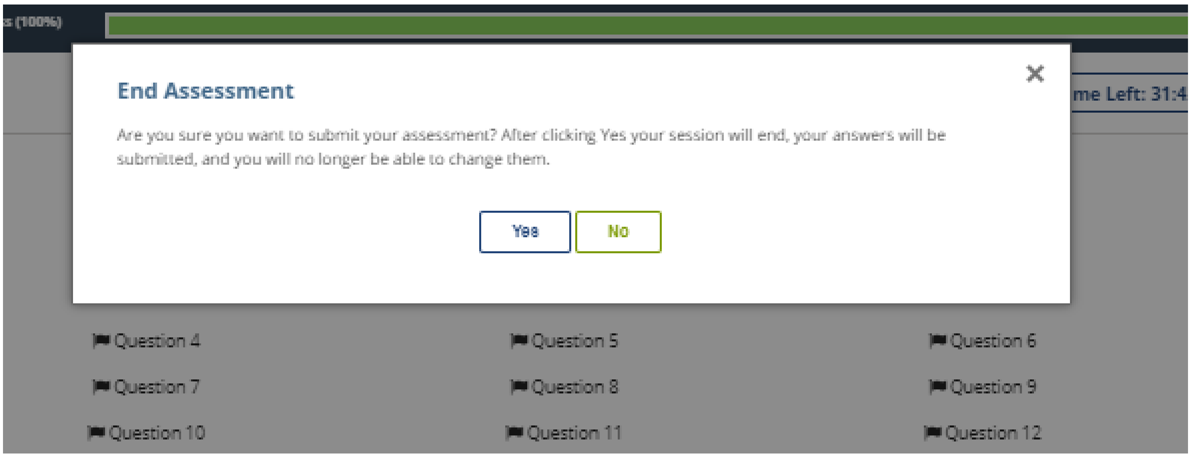
○ the snippet will enter an infinite loop, printing one * per line

● one

○ two

○ zero

Review Screen



End Assessment

Are you sure you want to submit your assessment? After clicking Yes your session will end, your answers will be submitted, and you will no longer be able to change them.