[CS 11] Prac 11h – Print After Return

Submit solution [CS 11] Practice 11

Points: 200 (partial)
Time limit: 8.0s
Memory limit: 1G

Author:
kvatienza (Kevin Atienza)

Problem type

Allowed languages
~~NONE~~, py3

Problem Statement

Write a decorator that decorates a function so that the return value is printed after each call.

Note:

- You can get the name of a function `f` via `f.__name__`
- Use `repr` to represent the arguments and return values as strings.

Task Details

Your task is to implement a decorator function called `print_return`. The function takes a single argument, the function `f` to be decorated.

The function must return the decorated version of `f`. It must act like `f`, but also print the return value after computing it. The return value must be printed in the following format:

`NAME(ARGS) = RETVAL`

where:

- `NAME` is the function name,
- `ARGS` is the stringified versions of the arguments separated by "`, `" (a comma followed by a space),
- `RETVAL` is the stringified version of the return value.

The stringified version of a value is its `repr`.

Restrictions

(See 11a for more restrictions)

For this problem in particular:

- The source code limit is 3000.

Example Testing

Note: This assumes that your submission has filename `prac11h.py`. Write this testing code in a separate file, say `test_prac11h.py`, and run it to test your code.

Example 1 Testing

```
from prac11h import print_return

@print_return
def fib(n):
    return n if n <= 1 else fib(n - 1) + fib(n - 2)

print(f"result: {fib(5) = }")
```

Example 1 Output

```
fib(1) = 1
fib(0) = 0
fib(2) = 1
fib(1) = 1
fib(3) = 2
fib(1) = 1
fib(0) = 0
fib(2) = 1
fib(4) = 3
fib(1) = 1
fib(0) = 0
fib(2) = 1
fib(1) = 1
fib(3) = 2
fib(5) = 5
result: fib(5) = 5
```

Example 2 Testing

```
from functools import cache

from prac11h import print_return

@cache
@print_return
def fib(n):
    return n if n <= 1 else fib(n - 1) + fib(n - 2)

print(f"result: {fib(5) = }")
assert fib(4) == 3
assert fib(5) == 5
```

Example 2 Output

```
fib(1) = 1
fib(0) = 0
fib(2) = 1
fib(3) = 2
fib(4) = 3
fib(5) = 5
result: fib(5) = 5
```

Example 3 Testing

```
from functools import cache

from prac11h import print_return

def path_count(grid):
    grid = [*map(''.join, grid)]

    print(*grid, sep='\n')

    assert grid, "grid must be nonempty"

    r, c = len(grid), len(grid[0])

    assert r >= 1 and c >= 1, "grid must be nonempty"
    assert {*map(len, grid)} == {c}, "grid rows must have the same length"

    def in_bounds(i: int, j: int) -> bool:
        return 0 <= i < r and 0 <= j < c

    def get(i: int, j: int) -> str:
        return grid[i][j] if in_bounds(i, j) else '#'

    @cache
    @print_return
    def p(i: int, j: int) -> int:
        if get(i, j) == '#':
            return 0
        elif (i, j) == (r-1, c-1):
            return 1
        else:
            return p(i + 1, j) + p(i, j + 1)

    return p(0, 0)


res = path_count((
    '.....',
    '..#..',
```

```
    '.....',
))
print(res)
assert res == 6
```

Example 3 Output

```
.....
..#..
.....
p(3, 0) = 0
p(3, 1) = 0
p(3, 2) = 0
p(3, 3) = 0
p(2, 4) = 1
p(2, 3) = 1
p(2, 2) = 1
p(2, 1) = 1
p(2, 0) = 1
p(1, 2) = 0
p(1, 1) = 1
p(1, 0) = 2
p(1, 5) = 0
p(1, 4) = 1
p(1, 3) = 2
p(0, 5) = 0
p(0, 4) = 1
p(0, 3) = 3
p(0, 2) = 3
p(0, 1) = 4
p(0, 0) = 6
6
```

Example 4 Testing

```
from prac11h import print_return

def merge(a, b):
    i, j = 0, 0
    while i < len(a) or j < len(b):
        if i < len(a) and (not (j < len(b)) or a[i] <= b[j]):
            yield a[i]; i += 1
        else:
            yield b[j]; j += 1

@print_return
def merge_sort(seq):
    if len(seq) <= 1:
        return tuple(seq)
    else:
        h = len(seq) // 2
        return tuple(merge(merge_sort(seq[:h]),
merge_sort(seq[h:])))

res = merge_sort((3, 1, 4, 1, 5, 9, 2, 6))
assert res == (1, 1, 2, 3, 4, 5, 6, 9)
```

Example 4 Output

```
merge_sort((3,)) = (3,)
merge_sort((1,)) = (1,)
merge_sort((3, 1)) = (1, 3)
merge_sort((4,)) = (4,)
merge_sort((1,)) = (1,)
merge_sort((4, 1)) = (1, 4)
merge_sort((3, 1, 4, 1)) = (1, 1, 3, 4)
merge_sort((5,)) = (5,)
merge_sort((9,)) = (9,)
merge_sort((5, 9)) = (5, 9)
merge_sort((2,)) = (2,)
merge_sort((6,)) = (6,)
merge_sort((2, 6)) = (2, 6)
merge_sort((5, 9, 2, 6)) = (2, 5, 6, 9)
merge_sort((3, 1, 4, 1, 5, 9, 2, 6)) = (1, 1, 2, 3, 4, 5, 6, 9)
```

Example 5 Testing

```
from prac11h import print_return

@print_return
def is_subseq(big, sml):
    if not sml:
        return True
    elif not big:
        return False
    elif big[0] == sml[0]:
        return is_subseq(big[1:], sml[1:])
    else:
        return is_subseq(big[1:], sml)

print(is_subseq('banana', 'anna'))
print(is_subseq('banana', 'anne'))
```

Example 5 Output

```
is_subseq('', '') = True
is_subseq('a', 'a') = True
is_subseq('na', 'na') = True
is_subseq('ana', 'na') = True
is_subseq('nana', 'nna') = True
is_subseq('anana', 'anna') = True
is_subseq('banana', 'anna') = True
True
is_subseq('', 'e') = False
is_subseq('a', 'e') = False
is_subseq('na', 'ne') = False
is_subseq('ana', 'ne') = False
is_subseq('nana', 'nne') = False
is_subseq('anana', 'anne') = False
is_subseq('banana', 'anne') = False
False
```

Example 6 Testing

```
from prac11h import print_return

@print_return
def alph(n, s):
    return '' if n == 0 else chr(ord('a') + s) + alph(n - 1, s +
1)

res = alph(5, 0)
assert res == 'abcde'
```

Example 6 Output

```
alph(0, 5) = ''
alph(1, 4) = 'e'
alph(2, 3) = 'de'
alph(3, 2) = 'cde'
alph(4, 1) = 'bcde'
alph(5, 0) = 'abcde'
```

Constraints

- The function `print_return` will be called at most 200 times.
- The function will only be called with immutable arguments.
- No keyword arguments will be passed.

Scoring

- You get 200 ❤ points if you solve all test cases.

? Clarifications

Report an issue

No clarifications have been made at this time.

proudly powered by **DMOJ** | English (en)