WAYGROUND formerly Quizizz **Worksheets**

Name

Class

Date

Decorators

Total questions: 9

Worksheet time: 5mins

1.

```
def repeat(num_times):
    def decorator_repeat(func):
        def wrapper(*args, **kwargs):
            for _ in range(num_times):
                func(*args, **kwargs)
        return wrapper
    return decorator_repeat

@repeat(3)
def say_hello():
    print("Hello")

say_hello()
```

What is the output of the following code?

a) Hello

b) Hello Hello Hello

c) Error

d) Hello Hello

2.

What is the output of the following code?

a) Function multiply called with args: {} and kwargs: (3, 5)

b) Function multiply

c) Function multiply called with args: (3, 5) and kwargs: {}

d) Error

3.

```
def uppercase_decorator(func):
    def wrapper(*args, **kwargs):
        original_result = func(*args, **kwargs)
        return original_result.upper()
    return wrapper

@uppercase_decorator
def greet(name):
    return f'Hello, {name}'

print(greet('James'))
```

What is the output of the following code?

a) Hello, JAMES

b) ERROR

c) Hello, James

d) HELLO, JAMES

4.

```
import time
def timer(func):
    def wrapper(*args, **kwargs):
        start_time = time.time()
        result = func(*args, **kwargs)
        end_time = time.time()
        print(f'{func.__name__} took {end_time - start_time} seconds')
        return result
    return wrapper

@timer
def long_running_function():
    time.sleep(2)
    return "Done"

long_running_function()
```

What is the output of the following code?

a) long_running_function took 2 seconds Done

b) long_running_function took 0 seconds

c) long_running_function took 2 seconds

d) Error

5.

```
def cache(func):
    memo = {}
    def wrapper(n):
        if n not in memo:
            memo[n] = func(n)
        return memo[n]
    return wrapper

@cache
def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n - 1) + fibonacci(n - 2)

print(fibonacci(5))
```

What is the output of the following code?

a) 8

b) 13

c) Error

d) 5

6.

```
def decorator_with_args(arg):
    def decorator(func):
        def wrapper(*args, **kwargs):
            print(f'Argument passed to decorator: {arg}')
            return func(*args, **kwargs)
        return wrapper
    return decorator

@decorator_with_args('Python')
def display_info(name):
    print(f'Displaying info for {name}')

display_info('Smith')
```

What is the output of the following code?

a) Displaying info for Smith

b) Error

c) Python

d) Argument passed to decorator: Python
Displaying info for Smith

7.

```
def check_positive(func):
    def wrapper(x):
        if x < 0:
            raise ValueError("Must be positive!")
        return func(x)
    return wrapper

@check_positive
def square(x):
    return x * x

print(square(4))
```

What is the output of the following code?

a) -16

b) ValueError: Must be positive!

c) 16

d) Error

8.

```
def ensure_not_empty(func):
    def wrapper(string):
        if not string:
            return "String is empty!"
        return func(string)
    return wrapper

@ensure_not_empty
def print_string(string):
    return string

print(print_string(""))
```

What is the output of the following code?

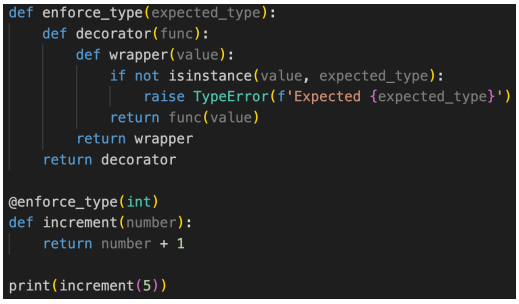
a) String is empty!

b) ''

c) String is not empty

d) Error

9.



```
def enforce_type(expected_type):
    def decorator(func):
        def wrapper(value):
            if not isinstance(value, expected_type):
                raise TypeError(f'Expected {expected_type}')
            return func(value)
        return wrapper
    return decorator

@enforce_type(int)
def increment(number):
    return number + 1

print(increment(5))
```

What is the output of the following code?

a) TypeError

b) Error

c) 5

d) 6

Answer Keys

1. b) Hello Hello Hello
2. c) Function multiply called with args: (3, 5) and kwargs: {}
3. d) HELLO, JAMES
4. c) long_running_function took 2 seconds
5. d) 5
6. d) Argument passed to decorator: Python Displaying info for Smith
7. c) 16
8. a) String is empty!
9. d) 6