

Python

1. Predict output of following code:

```
def funct():  
    print("Well some functions seem useless.. but are not")  
  
print(funct.__name__)
```

2. WAP to create a higher order function that takes a function as argument and prints its name on the screen.
3. What do following pieces of code do. Check on your system:

```
import time  
print(time.time())  
  
import time  
start_time = time.time()  
[i for i in range(10000)]  
diff = time.time() - start_time  
print(diff)
```

4. Check the output of following on your system after typing in a script [don't use command line directly]:

```
import time  
start_time = time.time()  
l1 = []  
for i in range(10000):  
    l1.append(i)  
time_for = time.time() - start_time  
  
start_time = time.time()  
l2 = [i for i in range(10000)]  
time_list_compr = time.time() - start_time  
  
print("For loop: ", time_for, " List Comprehension: ", time_list_compr)
```

5. What will the following code do. Is there a decorator in the code? Also predict the output of the code.

```
def wrapper(func):  
    print(func.__name__)  
    func()  
    return func  
  
def print_my_age():  
    print("Age is relative to perception of time")  
  
f = wrapper(print_my_age)  
f()
```

6. Rewrite the above code to use a decorator. The decorator should simply provide the functionality that the name of the function should get printed before it is called.

7. Without running the following code on your system, check whether it will give runtime error or not:

```
def decorator(func):
    def wrapper(x, y):
        print("Calling: ", func)
        return func(x, y)
    return wrapper

@decorator
def adder(x, y):
    return x + y

@decorator
def squarer(x):
    return x*x
```

8. Will the following code give any error? If yes think which statements produce what error and why.

```
def decorator(func):
    def wrapper(x, y):
        return func(x, y)
    return wrapper

@decorator
def adder(a, b):
    return a + b

@decorator
def squarer(a):
    return a*a

print(adder(10, 20))
print(adder(x = 10, y = 20))
print(adder(a = 10, b = 20))
print(squarer(4))
print(squarer(x = 10))
```

9. Rewrite the above code to fix the issue. Which features of python are you using to fix the issue.
10. Remember the time module?? Write a decorator called **profiler** using the time module to print the runtime of the function that it decorates.

```
def decorator(func):
    def wrapper(x, y):
        return func(x, y)
    return wrapper

@decorator
def adder(a, b):
    return a + b

@decorator
def squarer(a):
    return a*a

print(adder(10, 20))
print(adder(x = 10, y = 20))
print(adder(a = 10, b = 20))
print(squarer(4))
print(squarer(x = 10))
```