lesson_5

August 23, 2022

1 Lesson 5

1.1 Scopes

[1]:
```
n = 42

def foo():
    n += 1
    print(n)

foo()
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
Input In [1], in <cell line: 7>()
      4     n += 1
      5     print(n)
----> 7 foo()

Input In [1], in foo()
      3 def foo():
----> 4     n += 1
      5     print(n)

UnboundLocalError: local variable 'n' referenced before assignment
```

[3]:
```
l = [42, "test"]


def foo():
    l.append(False)
    print(l)


foo()
```

```
[42, 'test', False]
```

1.2 Decorators Contd.

[4]:
```
from time import sleep
```

[5]:
```
sleep(3)
```

[6]:
```
n = int(input("Please enter the n: "))
sleep(3)
print(n**2)
```

```
Please enter the n:  3
```

```
9
```

[7]:
```
from time import time
```

[14]:
```
start_time = time()
sleep(3)
end_time = time()
print(end_time - start_time)
```

```
3.0050437450408936
```

[15]:
```
from functools import wraps
```

[21]:
```
def timer(func):
    @wraps(func)
    def wrapper(*args, **kwargs):
        start_time = time()
        res = func(*args, **kwargs)
        end_time = time()
        print(f"Duration: {end_time - start_time}")
        return res
    return wrapper
```

[22]:
```
@timer
def foo(x):
    sleep(3)
    return x**2
```

[23]:
```
foo(2)
```

```
Duration: 3.003056764602661
```

[23]:
```
4
```

1.3 Missing Integer Problem

[24]:
```
from random import randint
```

[30]:
```
nums = []
while len(nums) < 90:
    r = randint(1, 100)
    if r not in nums:
        nums.append(r)
```

[39]:
```
@timer
def minpositive(arr):
    if 1 not in arr:  # O(n)
        return 1
    else:
        maxArr = max(arr) #  O(n)
        c1 = set(range(2, maxArr+2)) #  O(n)
        c2 = c1 - set(arr)  #  O(n)
        return f"the smallest positive number which is missing is {min(c2)}"

print(minpositive(nums))
```

```
Duration: 2.002716064453125e-05
the smallest positive number which is missing is 21
```

[40]:
```
@timer
def minpositive(arr):
    maxArr = max(arr) #  O(n)
    c1 = set(range(1, maxArr+2)) #  O(n)
    c2 = c1 - set(arr)  #  O(n)
    return f"the smallest positive number which is missing is {min(c2)}"

print(minpositive(nums))
```

```
Duration: 1.7881393432617188e-05
the smallest positive number which is missing is 21
```

[63]:
```
@timer
def minpositive(arr):
    maxArr = max(arr) #  O(n)
    c1 = set(range(1, maxArr+2)) #  O(n)
    c2 = c1 - set(arr)  #  O(n)
    if not c2:
        return 1
    return f"the smallest positive number which is missing is {min(c2)}"

print(minpositive([-2, -1, 1]))
```

```
Duration: 1.4781951904296875e-05
the smallest positive number which is missing is 2
```

[64]:
```python
@timer
def missing_nums(x):  # O(n^2)
    max_list = max(x)  # O(n)
    for y in range(1,max_list):  # O(n)
        if y not in x:  # O(n)
            return y
    else:
        return max_list+1

missing_nums(nums)
```

```
Duration: 2.9087066650390625e-05
```

[64]:
```
21
```

[65]:
```python
@timer
def missing_number(arr):
    s = set(arr)  # O(n)
    i = 1
    while True:  # O(n)
        if i not in s: # O(1)
            return i
        i += 1


missing_number(nums)
```

```
Duration: 1.0013580322265625e-05
```

[65]:
```
21
```

[66]:
```python
import timeit
```

[96]:
```python
def minpositive(arr):
    maxArr = max(arr) #  O(n)
    c1 = set(range(1, maxArr+2)) #  O(n)
    c2 = c1 - set(arr)  #  O(n)
    return f"the smallest positive number which is missing is {min(c2)}"


timeit.timeit('minpositive(nums)', number=10000, setup="from __main__ import␣
 ↪minpositive, nums")
```

[96]:
```
0.06545312499929423
```

[97]:
```
def missing_nums(x):  # O(n^2)
    max_list = max(x)  # O(n)
    for y in range(1,max_list):  # O(n)
        if y not in x:  # O(n)
            return y
    else:
        return max_list+1

timeit.timeit('missing_nums(nums)', number=10000, setup="from __main__ import␣
↪missing_nums, nums")
```

[97]:
```
0.11957237499973417
```

[98]:
```
def missing_number(arr):
    s = set(arr)  # O(n)
    i = 1
    while True:  # O(n)
        if i not in s: # O(1)
            return i
        i += 1

timeit.timeit('missing_number(nums)', number=10000, setup="from __main__ import␣
↪missing_number, nums")
```

[98]:
```
0.027627374999610765
```

[ ]:
```

```