Python Coding Questions for Data Engineer Interviews

Prime Number or not

In [1]:
```
n=int(input('enter n value: '))
if n<=1:
    print(f'{n} is not a prime number')
else:
    for i in range(2,n):
        if n%i ==0:
            print(f'{n} is not a prime number')
            break
    else:
        print(f'{n} is a prime number')
```

```
enter n value: 5
5 is a prime number
```

In [23]:
```
import time
import sys
import math

def isprime(x):
    if x<=1:
        return False
    else:
        for i in range(2,x):
            if x%i==0:
                return False
        else:
            return True
x=int(input('enter x value:'))
start_time = time.time()
if isprime(x):
    print(f'{x} is prime number')
else:
    print(f'{x} is not a prime number')
end_time = time.time()
print(f"Execution time: {end_time - start_time:.5f} seconds")
```

```
enter x value:481
481 is not a prime number
Execution time: 0.00106 seconds
```

In [24]:
```
def isprime(x):
    if x<=1:
        return False
    else:
        for i in range(2,int(x**0.5+1)):
            if x%i==0:
                return False
        else:
            return True
x=int(input('enter x value:'))
start_time = time.time()
if isprime(x):
    print(f'{x} is prime number')
else:
    print(f'{x} is not a prime number')
end_time = time.time()
print(f"Execution time: {end_time - start_time:.5f} seconds")
```

```
enter x value:481
481 is not a prime number
Execution time: 0.00000 seconds
```

Return a number is prime or not from a range of numbers

In [39]:
```
n=int(input('enter n value:'))
for i in range(1,n+1):
    def isprime(i):
        if i<=1:
            return False
        else:
            for j in range(2,i):
                if i%j ==0:
                    return False
            return True
    if isprime(i):
        print(f'{i} is prime number')
    else:
        print(f'{i} is not prime number')
```

```
enter n value:5
1 is not prime number
2 is prime number
3 is prime number
4 is not prime number
5 is prime number
```

Only return prime numbers range of given number from 1

In [68]:
```
n=int(input('enter n value:'))
for i in range(1,n+1):
    def isprime(i):
        if i<=1:
            return False
        else:
            for j in range(2,i):
                if i%j ==0:
                    return False
            return True
    start_time = time.time()
    if isprime(i):
        print(i, end=" ")
    end_time = time.time()
print(end='\n')
print(f'{end_time - start_time:.5f}')
```

```
enter n value:100
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
0.00000
```

In [69]:
```
n=int(input('enter n value:'))
count=0
for i in range(1,n+1):
    def isprime(i):
        if i<=1:
            return False
        else:
            for j in range(2,int(i**0.5+1)):
                if i%j ==0:
                    return False
            return True
    start_time = time.time()
    if isprime(i):
        count+=1
        print(f'{i} -> {count}')
    end_time = time.time()
print(end='\n')
print(f'{end_time - start_time:.5f}')
```

```
enter n value:100
2 -> 1
3 -> 2
5 -> 3
7 -> 4
11 -> 5
13 -> 6
17 -> 7
19 -> 8
23 -> 9
29 -> 10
31 -> 11
37 -> 12
41 -> 13
43 -> 14
47 -> 15
53 -> 16
59 -> 17
61 -> 18
67 -> 19
71 -> 20
73 -> 21
79 -> 22
83 -> 23
89 -> 24
97 -> 25

0.00000
```

Count the number of occurances a number exists in a list

In [93]:
```
L=[2,3,4,3,4,5,6,8,87,8,0,8]
dic={}
for i in L:
    if i in dic:
        dic[i]+=1
    else:
        dic[i]=1
print(dic)
```

```
{2: 1, 3: 2, 4: 2, 5: 1, 6: 1, 8: 3, 87: 1, 0: 1}
```

Check a given string is Palidrome or not

In [6]:
```
def is_palindrome(x):
    if x==x[::-1]:
        return True
    else:
        return False
x=input('enter a string:')
if is_palindrome(x):
    print(f'{x} is a palindrome')
else:
    print(f'{x} is not a palindrome')
```

```
enter a string:umrmu
umrmu is a palindrome
```

find the factorial of a number

In [26]:
```
n=int(input('enter n value:'))
import sys
import time
fact=1
start_time =time.time()
for i in range(1,n+1):
    fact*=i
end_time = time.time()
print(fact)
print(f'{end_time - start_time:.5f}')
```

```
enter n value:8
40320
0.00000
```

Return factorial values from 1 to a given number range

In [35]:
```
n=int(input('enter n value:'))
for i in range(1,n+1):
    fact=1
    for j in range(1,i+1):
        fact*=j
    print(fact)
```

```
enter n value:11
1
2
6
24
120
720
5040
40320
362880
3628800
39916800
```

In [38]:
```
def fact(n):
    fact=1
    for i in range(1,n+1):
        fact*=i
    return fact
```

In [39]:
```
fact(9)
```

Out[39]:
```
362880
```

In [48]:
```
def fact(n):
    for i in range(1,n+1):
        fact = 1
        for j in range(1,i+1):
            fact*=j
        print(fact)
```

In [49]:
```
fact(10)
```

```
1
2
6
24
120
720
5040
40320
362880
3628800
```

In [50]:
```
def fact(n):
    if n==1 or n==0:
        return 1
    else:
        return n * fact(n-1)
```

In [52]:
```
fact(5)
```

Out[52]:
```
120
```

In [77]:
```
def fact(n):
    if n==1 or n==0:
        return 1
    else:
        return n * fact(n-1)
```

In [90]:
```
def fact_range(n):
    result = []
    for i in range(1,n+1):
        print(f' {i} => {fact(i)}')
```

In [91]:
```
fact_range(10)
```

```
 1 => 1
 2 => 2
 3 => 6
 4 => 24
 5 => 120
 6 => 720
 7 => 5040
 8 => 40320
 9 => 362880
 10 => 3628800
```

find the largest element in a list

In [92]:
```
def largest_num(L):
    max = 0
    min = 0
    for i in range(len(L)):
        if L[i]>min:
            max=L[i]
        else:
            min = L[i]
    return max
```

In [94]:
```
L=[1,2,4,5,6]
largest_num(L)
```

Out[94]:
```
6
```

Reverse a string

In [97]:
```
def rev_str(x):
    return x[::-1]
```

In [96]:
```
rev_str('hello world')
```

Out[96]:
```
'dlrow olleh'
```

In [121]:
```
def rev_str_word(x):
    a=x.split(' ')
    ans =""
    for i in a:
        ans+= i[::-1] + ' '
    return ans
```

In [123]:
```
rev_str_word('hello world welcome to the programming language python')
```

Out[123]:
```
'olleh dlrow emoclew ot eht gnimmargorp egaugnal nohtyp '
```

In [124]:
```
rev_str('hello world welcome to the programming language python')
```

Out[124]:
```
'nohtyp egaugnal gnimmargorp eht ot emoclew dlrow olleh'
```

count the frequency of numbers in a list

In [132]:
```
def count_nums(x):
    dic={}
    for i in x:
        if i in dic:
            dic[i]+=1
        else:
            dic[i]=1
    return dic
```

In [134]:
```
a=[2,3,4,5,8,4,4,6,2,9,0]
count_nums(a)
```

Out[134]:
```
{2: 2, 3: 1, 4: 3, 5: 1, 8: 1, 6: 1, 9: 1, 0: 1}
```

Program to return a common elements from two lists

In [137]:
```
def common_elements(L1,L2):
    ans=[]
    for i in L1:
        if i in L2:
            ans.append(i)
    return ans
```

In [138]:
```
L=[1,2,3,5,7,8,9]
a=[4,3,2,9]
common_elements(L,a)
```

Out[138]:
```
[2, 3, 9]
```

Second largest number in a list

In [153]:
```
def second_largest(L):
    L.sort(reverse=True)
    return L[1]
```

In [154]:
```
second_largest(L)
```

Out[154]:
```
8
```

In [171]:
```
def second_largest(L):
    largest=float('-inf')
    second_largest=float('-inf')
    for i in L:
        if i>largest:
            second_largest = largest
            largest = i
        elif i!=largest and i>second_largest:
            second_largest = i
    return second_largest
```

In [172]:
```
second_largest(L)
```

Out[172]:
```
8
```

Remove duplicates

In [174]:
```
def remove_duplicates(L):
    ans=[]
    for i in L:
        if i not in ans:
            ans.append(i)
    return ans
```

In [3]:
```
a=[1,2,3,4,5,6,9,2,2,3,4,5]
remove_duplicates(a)
```

Out[3]:
```
[1, 2, 3, 4, 5, 6, 9]
```

In [4]:
```
def remove_duplicates(L):
    return list(set(L))
```

In [5]:
```
remove_duplicates(a)
```

Out[5]:
```
[1, 2, 3, 4, 5, 6, 9]
```

In [16]:
```
b=a.copy() #shallow copy ==> created new object reference of original eleme
```

In [17]:
```
b
```

Out[17]:
```
[1, 2, 3, 4, 5, 6, 9, 2, 2, 3, 4, 5]
```

In [18]:
```
import copy
g=copy.deepcopy(a)  #deep copy ==> completely independ copy object.
```

In [19]:
```
g
```

Out[19]:
```
[1, 2, 3, 4, 5, 6, 9, 2, 2, 3, 4, 5]
```

What is a decorator in Python?

A decorator is a design pattern in Python that allows modifying the behavior of a function or class without directly changing its source code. Decorators are represented using the '@' symbol.

Explain the difference between '==', 'is', and 'in' operators.

The '==' operator compares the values of two objects, while the 'is' operator checks if two objects refer to the same memory location. The 'in' operator is used to check if a value exists within an iterable.

What is the purpose of the 'pass' statement in Python?

The 'pass' statement is a placeholder statement used when a statement is syntactically required but does not need any code execution. It is commonly used as a placeholder for future code.

What is a lambda function?

A lambda function is a small anonymous function defined using the 'lambda' keyword. It can take any number of arguments but can only have one expression. Lambda functions are typically used when a function is required for a short period.

In [24]:
```
numbers = [1, 2, 3, 4]
squared_numbers = list(map(lambda x: x*x, numbers))
print(squared_numbers)
```

```
[1, 4, 9, 16]
```

In [37]:
```
a=list(map(lambda d:d**d, numbers))
```

In [38]:
```
a
```

Out[38]:
```
[1, 4, 27, 256]
```

List Comprehension

List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list

In [39]:
```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = []

for x in fruits:
  if "a" in x:
    newlist.append(x)

print(newlist)
```

```
['apple', 'banana', 'mango']
```

In [40]:
```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x for x in fruits if "a" in x]

print(newlist)
```

```
['apple', 'banana', 'mango']
```

In [43]:
```
#reverse a list
a
```

Out[43]:
```
[1, 4, 27, 256]
```

In [44]:
```
a[::-1]
```

Out[44]:
```
[256, 27, 4, 1]
```

What is the difference between ‘pop()’ and ‘remove()’ methods for lists?

The ‘pop()’ method removes and returns the last element from a list or the element at a specified index, while the ‘remove()’ method removes the first occurrence of a specified element from a list.

In [46]:
```
a = [1, 2, 3]
a.remove(2)         #referes to element removals
a               # [1, 3]

a = [1, 2, 3]
del a[1]
a               # [1, 3]

a = [1, 2, 3]
a.pop(1)        # 2   #pop reference to indexed value removal
a               # [1, 3]
```

Out[46]:
```
[1, 3]
```

Explain the difference between ‘os’ and ‘sys’ modules in Python.

The ‘os’ module provides functions for interacting with the operating system, such as file operations and directory manipulations. The ‘sys’ module provides functions and variables related to the Python interpreter itself.

How can you check if a file exists in Python?

You can use the ‘os.path.exists()’ function to check if a file exists. It returns ‘True’ if the file exists, and ‘False’ otherwise.

How can you convert a list of strings to a single string in Python?

You can use the ‘join()’ method to concatenate a list of strings into a single string. For example, ‘“”.join(string_list)’.

In [60]:
```
a=['mahi','mahendra','sam','lokesh','sai']
ans=''.join(a)
```

In [61]:
```
ans
```

Out[61]: 'mahimahendrasamlokeshsai'

In [74]:
```
ans=''
for i in a:
    ans=ans+i+' '
print(ans)
```

```
mahi mahendra sam lokesh sai
```

In [75]:
```
len(ans)
```

Out[75]: 29

In [76]:
```
ans.upper()
```

Out[76]: 'MAHI MAHENDRA SAM LOKESH SAI '

In [77]:
```
ans.lower()
```

Out[77]: 'mahi mahendra sam lokesh sai '

Explain the difference between a generator and a list in Python.

A generator is an iterator that generates values on-the-fly, only when requested. It is memory-efficient as it generates values one at a time. In contrast, a list stores all the values in memory at once.

In [78]:
```
a
```

Out[78]: ['mahi', 'mahendra', 'sam', 'lokesh', 'sai']

In [80]:
```
l=[2,3,5,55,6,7,88,5,33,22,221,453,229]
```

In [84]:
```
l.min()
l.max()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[84], line 1
----> 1 l.min()
      2 l.max()

AttributeError: 'list' object has no attribute 'min'
```

In [82]:
```
min(l)
```

Out[82]: 2

In [83]:
```
max(l)
```

Out[83]: 453

Explain the purpose of the ‘random’ module in Python.

The ‘random’ module provides functions for generating random numbers and performing random selections. It is commonly used for simulations, games, and cryptographic applications.

What is the difference between .py and .pyc files?

.py files contain Python source code, while .pyc files contain the bytecode that the Python interpreter generates from the .py files for faster execution.

Explain the difference between map() and filter() functions.

map() applies a function to all items in an iterable and returns a list of results. filter() returns a list of items for which a function returns True.

Decorator Example

In [85]:
```python
def square_decorator(func):
    def wrapper(num):
        result = func(num)
        return result ** 2
    return wrapper

@square_decorator
def add_five(x):
    return x + 5

print(add_five(3))  # Output: 64 (because (3 + 5)² = 64)
```

```
64
```

What is recursion in Python?

Recursion is a function calling itself to solve a smaller instance of the same problem. Example: calculating the factorial of a number.

How does memory management work in Python?

Memory management in Python involves private heap space, managed by Python's memory manager and supported by garbage collection.

What is monkey patching in Python?

Monkey patching refers to dynamic modifications of a class or module at runtime.

What is the Global Interpreter Lock (GIL)?

The GIL is a mutex that protects access to Python objects, preventing multiple threads from executing Python bytecodes simultaneously.

What is polymorphism in Python?

Polymorphism allows methods to do different things based on the object it is acting upon, enabling different classes to be treated through the same interface.

In [ ]:
```python

```