Untitled

February 27, 2023

Q1. You are writing code for a company. The requirement of the company is that you create a python function that will check whether the password entered by the user is correct or not. The function should take the password as input and return the string "Valid Password" if the entered password follows the below-given password guidelines else it should return "Invalid Password". Note: 1. The Password should contain at least two uppercase letters and at least two lowercase letters. 2. The Password should contain at least a number and three special characters. 3. The length of the password should be 10 characters long.

Here's an implementation of the function that checks the validity of the password:

python

[2]:
```
import re

def check_password(password):
    if len(password) != 10:
        return "Invalid Password"

    uppercase_count = len(re.findall(r'[A-Z]', password))
    lowercase_count = len(re.findall(r'[a-z]', password))
    digit_count = len(re.findall(r'\d', password))
    special_count = len(re.findall(r'[!@#$%^&*()-+]', password))

    if uppercase_count < 2 or lowercase_count < 2 or digit_count < 1 or␣
 ↪special_count < 3:
        return "Invalid Password"
    else:
        return "Valid Password"
```

Explanation:

The function takes a single argument password. It first checks whether the length of the password is 10 or not. If it's not 10, it returns "Invalid Password". It then uses regular expressions to count the number of uppercase letters, lowercase letters, digits, and special characters in the password. Finally, it checks whether the password meets the given criteria (at least two uppercase letters, at least two lowercase letters, at least one digit, and at least three special characters). If the password does not meet the criteria, it returns "Invalid Password". Otherwise, it returns "Valid Password".

Q2. Solve the below-given questions using at least one of the following: 1. Lambda functioJ 2. Filter functioJ 3. Zap functioJ 4. List ComprehensioI B Check if the string starts with a particular

letterY B Check if the string is numericY B Sort a list of tuples having fruit names and their quantity. [(“mango”,99),(“orange”,80), (“grapes”, 1000)- B Find the squares of numbers from 1 to 10Y B Find the cube root of numbers from 1 to 10Y B Check if a given number is evenY B Filter odd numbers from the given list. [1,2,3,4,5,6,7,8,9,10- B Sort a list of integers into positive and negative integers lists. [1,2,3,4,5,6,-1,-2,-3,-4,-5,0]

Check if the string starts with a particular letter: Using lambda and startswith() method:

[3]:
```
check_startswith = lambda string, letter: string.startswith(letter)
print(check_startswith("Hello World", "H"))  # True
print(check_startswith("Hello World", "W"))  # False
```

```
True
False
```

Sort a list of tuples having fruit names and their quantity: Using sorted() function with key parameter and lambda function:

[6]:
```
fruits = [("mango", 99), ("orange", 80), ("grapes", 1000)]
sorted_fruits = sorted(fruits, key=lambda fruit: fruit[1])
print(sorted_fruits)  # [("orange", 80), ("mango", 99), ("grapes", 1000)]
```

```
[('orange', 80), ('mango', 99), ('grapes', 1000)]
```

Find the squares of numbers from 1 to 10: Using list comprehension:

[7]:
```
squares = [x**2 for x in range(1, 11)]
print(squares)  # [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Find the cube root of numbers from 1 to 10: Using list comprehension and pow() function:

[8]:
```
cube_roots = [pow(x, 1/3) for x in range(1, 11)]
print(cube_roots)  # [1.0, 1.2599210498948732, 1.4422495703074083, 1.
↪5874010519681994, 1.7099759466766968, 1.8171205928321397, 1.912931182772389,␣
↪2.0, 2.080083823051904, 2.154434690031884]
```

```
[1.0, 1.2599210498948732, 1.4422495703074083, 1.5874010519681994,
1.7099759466766968, 1.8171205928321397, 1.912931182772389, 2.0,
2.080083823051904, 2.154434690031884]
```

Check if a given number is even: Using lambda and % operator:

[9]:
```
check_even = lambda num: num % 2 == 0
print(check_even(4))  # True
print(check_even(5))  # False
```

```
True
False
```

Filter odd numbers from the given list: Using filter() function and lambda function:

[10]:
```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
odd_numbers = list(filter(lambda num: num % 2 != 0, numbers))
print(odd_numbers)  # [1, 3, 5, 7, 9]
```

```
[1, 3, 5, 7, 9]
```

Sort a list of integers into positive and negative integers lists: Using list comprehension and sorted() function with key parameter:

[11]:
```
numbers = [1, 2, 3, 4, 5, 6, -1, -2, -3, -4, -5, 0]
positive_numbers = sorted([num for num in numbers if num > 0])
negative_numbers = sorted([num for num in numbers if num < 0])
print(positive_numbers)  # [1, 2, 3, 4, 5, 6]
print(negative_numbers)  # [-5, -4, -3, -2, -1]
```

```
[1, 2, 3, 4, 5, 6]
[-5, -4, -3, -2, -1]
```

[ ]:
```

```