Practical-1

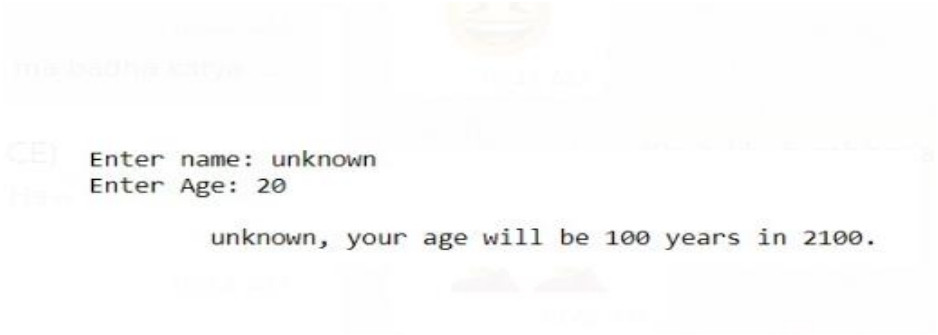
Practical 1.1

Aim: Create a program that asks the user to enter their name and their age. Printout a message addressed to them that tells them the year that they will turn100 years old.

Code:

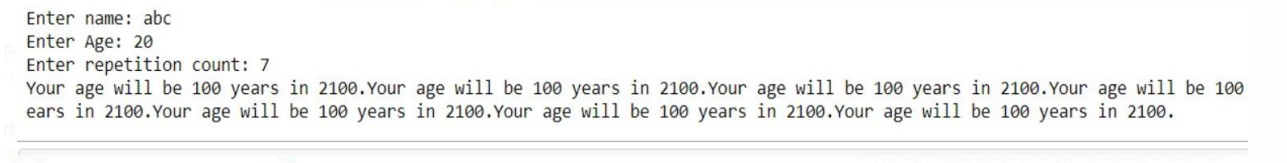
```
import datetime
name = str(input("Enter name: "))
age = int(input("Enter Age: "))
total_age = (datetime.datetime.now().year - age)+ 100
print(f'\n\t{name}, your age will be 100 years in {total_age}.')
```

Output:



Extra:

1. **Aim:** Add on to the previous program by asking the user for another number and printing out that many copies of the previous message.
   Code:
   ```
   name = str(input("Enter name: "))
   age = int(input("Enter Age: "))
   times = int(input("Enter repetition count: "))
   total_age = (datetime.datetime.now().year - age +100)
   print(f'Your age will be 100 years in {total_age}.'*times)
   ```
   Output:

2. **Aim:** Print out that many copies of the previous message on separate lines.
   Code:
   ```
   name = str(input("Enter name: "))
   age = int(input("Enter Age: "))
   times = int(input("Enter repetition count: "))
   total_age = (datetime.datetime.now().year - age +100)
   print(f'Your age will be 100 years in {total_age}.\n'*times)
   ```

   Output:

```
Enter name: user
Enter Age: 20
Enter repetition count: 7
Your age will be 100 years in 2100.
Your age will be 100 years in 2100.
Your age will be 100 years in 2100.
Your age will be 100 years in 2100.
Your age will be 100 years in 2100.
Your age will be 100 years in 2100.
Your age will be 100 years in 2100.
```

Practical 1.2

Aim: Ask the user for a number. Depending on whether the number is even orodd, print out an appropriate message to the user.

Code:

```
num = int(input("Input number: "))
if num%2 == 0:
   print(f'{num} is even')
else:
   print(f'{num} is odd')
```

Output:

```
Input number: 10
10 is even

Input number: 13259
13259 is odd
```

Extra :

1. **Aim:** If the number is a multiple of 4, print out a different message.

   Code:

   ```
   num = int(input("Input number: "))
   if (num%4) == 0:
           print(f'{num} is multiple of 4')
   else:
           print(f'{num} is not a multiple of 4')
   ```

   Output:

   ```
   Input number: 20
   20 is multiple of 4
   ```

2. **Aim:** Ask the user for two numbers: one number to check (call it num) and one number to divide by (check). If check divides evenly into num, tell that to the user. If not, print a different appropriate message.

Code:

```
num = int(input("Enter number: "))
div = int(input("Enter divisor : "))
if (num%div) == 0:
   print(f'{num} is divisible of {div}')
else:
   print(f'{num} is not a divisible of {div}')
```

Output:

```
Enter number: 20
Enter divisor : 2
20 is divisible of 2

Enter number: 25
Enter divisor : 6
25 is not a divisible of 6
```

Practical 1.3

Aim: Take a list, say for example this one: a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] and write a program that prints out all the elements of the list that are less than 5.

Code:

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
n = ""
for i in a:
   if int(i) < 5:
      n += str(i) + ", "
         print(f'{n}')
```

Output:

```
[1]
[1, 1]
[1, 1, 2]
[1, 1, 2, 3]
[1, 1, 2, 3]
[1, 1, 2, 3]
[1, 1, 2, 3]
[1, 1, 2, 3]
[1, 1, 2, 3]
[1, 1, 2, 3]
[1, 1, 2, 3]
```

Extra :

1. **Aim:** Instead of printing the elements one by one, make a new list that has all the elements less than 5 from this list in it and print out this new list.

   Code:

   ```
   a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
   ```

```
n = []
for i in a:
   if int(i) < 5:
      n.append(i)
print(f'{n}')
```

Output:

```
[1, 1, 2, 3]
```

2. **Aim:**Write this in one line of Python.

Code:

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
x=[i for i in a if i <5]
print(x)
```

Output:

```
[1, 1, 2, 3]
```

3. **Aim:** Ask the user for a number and return a list that contains only elements from the original list that are smaller than that number.

Code:

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
n = int(input("Input number: "))
new_list = []
for i in a:
   if int(i) < n:
      new_list.append(i)
print(f'{new_list}')
```

Output:

```
Input number: 10
[1, 1, 2, 3, 5, 8]
```

Conclusion: In this practical we learned how to take input from the user and apply basic operations on that input.

Practical-2

Practical 2.1

Aim: Create a program that asks the user for a number and then prints out a list of all the divisors of that number. (If you don't know what a divisor is, it is a number that divides evenly into another number. For example, 13 is a divisor of 26 because 26 / 13 has no remainder.)

Code:

```
n = int(input("Enter n: "))
for i in range(n):
    if n%(i+1) == 0:
        print(f'{i+1} is divisor of {n}')
```

Output:

```
Enter n: 12
1 is divisor of 12
2 is divisor of 12
3 is divisor of 12
4 is divisor of 12
6 is divisor of 12
12 is divisor of 12
```

Practical 2.2

Aim: Take two lists, say for example these two:

a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

and write a program that returns a list that contains only the elements thatare common between the lists (without duplicates). Make sure your programworks on two lists of different sizes.

Code:

```
lst1 = []
n = int(input("Enter number of elements : "))
for i in range(0, n):
    e1 = int(input())
    lst1.append(e1)
print(lst1)
lst2 = []
m = int(input("Enter number of elements : "))
for i in range(0, m):
    e2 = int(input())
    lst2.append(e2)
print(lst2)
list_1 = (lst1)
list_2 = (lst2)
output = []
for i in list_1:
    for j in list_2:
        if int(i) == int(j):
            output.append(i)
print("Common elements list: ",output)
```

Output:

```
Enter number of elements : 7
1
1
2
5
9
10
12
[1, 1, 2, 5, 9, 10, 12]
Enter number of elements : 5
1
2
5
6
8
[1, 2, 5, 6, 8]
Common elements list:  [1, 1, 2, 5]
```

Extra

1. **Aim:**Randomly generate two lists to test this
   Code:

```
import random
a = [random.randint(2,40) for i in range(10)]
b = [random.randint(1,40) for i in range(10)]
print(a)
print(b)
output = []
for i in a:
    for j in b:
        if int(i) == int(j):
            output.append(i)
print("Common elements list: ",output)
```

   Output:

```
[34, 40, 38, 37, 20, 26, 3, 4, 7, 32]
[36, 28, 19, 31, 37, 22, 20, 20, 34, 33]
Common elements list:  [34, 37, 20, 20]
```

---

2. **Aim:** Write this in one line of Python.
   Code:

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
result = [i for i in a if i in b]
print(result)
```

   Output:

```
[1, 1, 2, 3, 5, 8, 13]
```

Practical 2.3

Aim: Ask the user for a string and print out whether this string is a palindrome ornot. (A palindrome is a string that reads the same forwards and backwards.)

Code:

```
a = input("Enter string : ")
x = list(a)
y = x[::-1]
count = 0
for i in range(len(x)):
    if str(x[i]).lower() == str(y[i]).lower():
        count += 1
    else:
        break
if count == len(x):
    print("String is Palindrome.\n")
else:
    print("String is not Palindrome.\n")
```

Output:

```
Enter string : MaDam
String is Palindrome.
```

Conclusion: In this practical we learned the implementation of palindrome, working with given list and for loop.

Practical-3

Practical 3.1

Aim: Let's say I give you a list saved in a variable: a = [1, 4, 9, 16, 25, 36, 49, 64, 81,100]. Write one line of Python that takes this list a and makes a new list thathas only the even elements of this list in it.

Code:

```
a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
even = []
for i in a:
    if (a.index(i)+1)%2 == 0:
        even.append(i)
print(even)
```

Output:

```
[4, 16, 36, 64, 100]
```

Practical 3.2

Aim: Make a two-player Rock-Paper-Scissors game. (Hint: Ask for player plays(using input), compare them, print out a message of congratulations to thewinner, and ask if the players want to start a new game)

Remember the rules:

Rock beats scissors

Scissors beats paper

Paper beats rock

Code:

```
import sys
while True:
    user1 = input("What's your name?")
    user2 = input("And your name?")
    user1_answer = input("%s, do yo want to choose rock, paper or scissors?" % user1)
    user2_answer = input("%s, do you want to choose rock, paper or scissors?" % user2)
    def compare(u1, u2):
        if u1 == u2:
            return("It's a tie!")
        elif u1 == 'rock':
            if u2 == 'scissors':
                return("Rock wins!")
            else:
                return("Paper wins!")
        elif u1 == 'scissors':
            if u2 == 'paper':
                return("Scissors win!")
            else:
                return("Rock wins!")
```

```
        elif u1 == 'paper':
            if u2 == 'rock':
                return("Paper wins!")
            else:
                return("Scissors win!")
        else:
            return("Invalid input! You have not entered rock, paper or scissors, try again.")
            sys.exit()
    print(compare(user1_answer, user2_answer))
    print("Do you want to play again? (Y/N)")
    ans = input()
    if ans == 'n' or ans == 'N':
        break
print("\nThanks for playing")
```

Output:

```
What's your name?user1
And your name?user2
user1, do yo want to choose rock, paper or scissors?rock
user2, do you want to choose rock, paper or scissors?paper
Paper wins!
Do you want to play again? (Y/N)
n

Thanks for playing
```

Practical 3.3

Aim: Generate a random number between 1 and 9 (including 1 and 9). Ask theuser to guess the number, then tell them whether they guessed too low, toohigh, or exactly right.

Code:

```
import random
number = random.randint(1,9)
guess = 0
count = 0
while guess != number and guess != "exit":
    guess = input("What's your guess?")
    if guess == "exit":
        break
    guess = int(guess)
    count += 1
    if guess < number:
```

```
    print("Too low!")
  elif guess > number:
    print("Too high!")
  else:
    print("You got it!")
    print("And it only took you",count,"tries!")
```

Output:

```
What's your guess?9
Too high!
What's your guess?5
Too high!
What's your guess?3
Too low!
What's your guess?4
You got it!
And it only took you 4 tries!

What's your guess?5
Too high!
What's your guess?4
Too high!
What's your guess?2
You got it!
And it only took you 3 tries!
```

Extra:

1. **Aim:** Keep the game going until the user types “exit”. Keep track of how many guesses the user has taken, and when the game ends&print this out.

   Code:

```
import random
guess_number = random.randint(1,10)
guess = ""
count = 0

while True:
  guess = input("\nGuess: ")
  if guess.lower() == "exit":
    print("Your count is",count)
    break
  if int(guess) == int(guess_number):
    print("You guessed the correct number.\n\tYour count is",count)
    break
  elif int(guess) > guess_number:
    print("You guess too high.")
    count += 1
  else:
    print("You guess too low.")
    count += 1
```

Output:

```
Guess: 3
You guess too low.

Guess: 1
You guess too low.

Guess: 0
You guess too low.

Guess: 6
You guess too high.

Guess: 5
You guessed the correct number.
        Your count is 4
```

```
Guess: 6
You guess too low.

Guess: exit
Your count is 1
```

Conclusion: In this practical we learned how to use if with elif and how to find factors of a number.

Practical-4

Practical 4.1

Aim: Take two lists, say for example these two:

a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

and write a program that returns a list that contains only the elements that arecommon between the lists (without duplicates). Make sure your program works ontwo lists of different sizes. Write this in one line of Python using at least one listcomprehension.

Code:

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
result = [i for i in a if i in b]
print(result)
```

Output:

```
[1, 1, 2, 3, 5, 8, 13]
```

Extra:

1. **Aim:**Randomly generate two lists to test this.

   Code:

```
import random
a = [random.randint(2,40) for i in range(10)]
b = [random.randint(1,40) for i in range(10)]
print(a)
print(b)
output = []
for i in a:
    for j in b:
        if int(i) == int(j):
            output.append(i)
print("Common elements list: ",output)
```

   Output:

```
[34, 40, 38, 37, 20, 26, 3, 4, 7, 32]
[36, 28, 19, 31, 37, 22, 20, 20, 34, 33]
Common elements list:  [34, 37, 20, 20]
```

Practical 4.2

Aim: Ask the user for a number and determine whether the number is prime ornot. (For those who have forgotten, a prime number is a number that has nodivisors.). You can use your answer to Practical 2 to help you.Take this opportunity to practice using functions.

Code:

```
def divisor(n):
    count = 0
```

```
    for i in range(2, n):
        if n % i == 0:  count += 1
    return count
n = int(input("\nInput: "))
if int(divisor(n)) > 0:
    print("Not Prime")
else:
    print("Prime")
```

Output:

```
Input: 13
Prime

 Input: 25
 Not Prime
```

Practical 4.3

Aim: Write a program that takes a list of numbers (for example, a = [5, 10, 15, 20,25]) and makes a new list of only the first and last elements of the given list.For practice, write this code inside a function.

Code:

```
num_of_elements = int(input("Enter number of elements of list: "))
a = []
print("Input numbers: ")
for i in range(num_of_elements):
    x = int(input("> "))
    a.append(x)
l1 = [a[0],a[-1]]
print(f'\tList of first & last number: {l1}\n')
```

Output:

```
Enter number of elements of list: 5
Input numbers:
> 2
> 6
> 3
> 4
> 10
        List of first & last number: [2, 10]
```

Conclusion: In this practical we learned about how to split in a list, how to append inside a list and write a comprehensive command in one line.

Practical-5

Practical 5.1

Aim: Write a program that asks the user how many Fibonnaci numbers togenerate and then generates them. Take this opportunity to think about howyou can use functions. Make sure to ask the user to enter the number ofnumbers in the sequence to generate. (Hint: The Fibonnaci seqence is asequence of numbers where the next number in the sequence is the sum ofthe previous two numbers in the sequence. The sequence looks like this: 0, 1, 1,2, 3, 5, 8, 13, …)

Code:

```
nterms = int(input("How many terms? "))
n1, n2 = 0, 1
count = 0
if nterms <= 0:
   print("Please enter a positive integer")
elif nterms == 1:
   print("Fibonacci sequence upto",nterms,":")
   print(n1)
else:
   print("Fibonacci sequence:")
   while count < nterms:
      print(n1)
      nth = n1 + n2
      n1 = n2
      n2 = nth
      count += 1
```

Output:

```
How many terms? 3
Fibonacci sequence:
0
1
1
```

```
How many terms? 9
Fibonacci sequence:
0
1
1
2
3
5
8
13
21
```

Practical 5.2

Aim: Write a program (function!) that takes a list and returns a new list thatcontains all the elements of the first list minus all the duplicates.

Code:

```
def dedupe_v1(x):
```

```
  y = []
  for i in x:
    if i not in y:
      y.append(i)
  return y
lst = []
n = int(input("Enter number of elements : "))
for i in range(0, n):
    e = int(input())
    lst.append(e)
print(dedupe_v1(lst))
```

Output:

```
Enter number of elements : 10
2
2
3
6
6
5
9
10
10
25
[2, 3, 6, 5, 9, 10, 25]
```

Extra:

1. **Aim:** Write two different functions to do this - one using a loop and constructing a list, and another using sets. Go back and do Practical 2 using sets, and write the solution for that.

   Code:

   ```
   def func_set(list):
       uniq = set(list)
       return uniq
   a_list = []
   input_range = int(input("\nEnter number of elements : "))
   for i in range(input_range):
       element = int(input())
       a_list.append(element)
   print("\nList - Dup_elements Using Set = ", func_set(a_list))
   ```

   Output:

```
Enter number of elements : 9
2
3
3
5
25
20
20
20
2

List - Dup_elements Using Set =  {2, 3, 5, 20, 25}
```

Practical 5.3

Aim: Write a program (using functions!) that asks the user for a long stringcontaining multiple words. Print back to the user the same string, except withthe words in backwards order. For example,

say I type the string:**"My name is Michele"**

Then I would see the string:**"Michele is name My"**

shown back to me.

Code:

```
def reverse_str(x):
  y = x.split()
  result = []
  for word in y:
    result.insert(0,word)
  return " ".join(result)
string = input("Enter a sentence: ")
print(reverse_str(string))
```

Output:

```
Enter a sentence: My name is Michele
Michele is name My
```

Conclusion: In this practical we learned how to how to work with different methods of list, and how to implement the functions by creating them.

Practical-6

Practical 6.1

Aim: Write a password generator in Python. Be creative with how you generatepasswords - strong passwords have a mix of lowercase letters, uppercaseletters, numbers, and symbols. The passwords should be random, generatinga new password every time the user asks for a new password. Include yourrun-time code in a main method.

Code:

```
import random
s = "abcdefghijklmnopqrstuvwxyz01234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ!@#$%^&*()?"
passlen = 8
p =  "".join(random.sample(s,passlen ))
print(p)
```

Output:

```
ZCB(0GP0
```

Extra :

1. **Aim:** Ask the user how strong they want their password to be. For weak passwords, pick a word or two from a list.

   Code:

   ```
   import string
   import random
   def pw_gen(size = 8, chars=string.ascii_letters + string.digits + string.punctuation):
       return ''.join(random.choice(chars) for _ in range(size))
   print(pw_gen(int(input('How many characters in your password?'))))
   ```

   Output:

   ```
   How many characters in your password?3
   Og9

   How many characters in your password?20
   ],/"U3"HHW_rX.0c_3-$
   ```

Practical 6.2

Aim: Use the BeautifulSoup and requests Python packages to print out a list of allthe article titles on the New York Times homepage.

Code:

```
import requests
from bs4 import BeautifulSoup
url = 'http://www.nytimes.com' r = requests.get(url)
r_html = r.text
soup = BeautifulSoup(r_html, "lxml")
for titles in soup.find_all(class_="story"): title = titles.a
print(title.string)
```

Output:

```
Listen to 'Still Processing': M.J.
'The Daily' Newsletter
Listen to 'The Argument'
Amazon's Tax Breaks and Incentives Were Big. Hudson Yards' Are Bigger.
ISIS Rises in Philippines as It Dwindles in Middle East
Bernie Sanders-Style Politics Are Defining 2020 Race, Unnerving Moderates
Klobuchar and Warren Take Their Messages to South by Southwest
In South Africa's Fabled Wine Country, White and Black Battle Over Land
U.S. Continues to Separate Migrant Families Despite Rollback of Policy
11 of Our Best Weekend Reads
Test your knowledge of the week's headlines with our news quiz.
Daylight saving time begins at 2 a.m. in the U.S. But some wonder if it's time for time to be left
alone.
Will There Be Smoking Guns in the Mueller Report?
Are You an Amazon or an Apple Family?
What Alex Trebek Is Really Like
Is Anti-Semitism Exceptional?
I Am Not Your Tinder Fantasy
'An Angel From God,' and Border Agents Took Her
```

Practical 6.3

Aim: Create a program that will play the "cows and bulls" game with the user. The gameworks like this:

Randomly generate a 4-digit number. Ask the user to guess a 4-digit number. Forevery digit that the user guessed correctly in the correct place, they have a "cow".For every digit the user guessed correctly in the wrong place is a "bull." Every timethe user makes a guess, tell them how many "cows" and "bulls" they have. Once the user guesses the correct number, the game is over. Keep track of the number ofguesses the user makes throughout teh game and tell the user at the end.Say the number generated by the computer is 1038. An example interaction couldlook like this:

Welcome to the Cows and Bulls Game!

Enter a number:

>>> 1234

2 cows, 0 bulls

>>> 1256

1 cow, 1 bull

...

Until the user guesses the number.

Code:

```
import random
n = str(random.randint(1000,9999))
nlist = []
cow = 0
for i in n:
   nlist.append(i)
while cow < 4 and exit !="x":
   x = str(input("Choose a 4 digit number, x to exit: "))
   xlist = []
   cow = 0
   bull = 0
   if x!= "x":
      for i in x:
         xlist.append(i)
```

```
        for i in nlist:
            if i in xlist and nlist.index(i) == xlist.index(i):
                cow +=1
            if i in xlist and nlist.index(i) != xlist.index(i):
                bull +=1
        print(cow, "cow(s)", bull, "bull(s)")
    else:
        exit = "x"
print(nlist, xlist)
```

Output:

```
Choose a 4 digit number, x to exit: 1100
0 cow(s) 2 bull(s)
Choose a 4 digit number, x to exit: 2322
0 cow(s) 0 bull(s)
Choose a 4 digit number, x to exit: 4015
3 cow(s) 0 bull(s)
Choose a 4 digit number, x to exit: 2021
0 cow(s) 2 bull(s)
```

Conclusion: In this practical we learned how to generate random numbers and use them for different purposes as well as how to get text from URL using Beautiful Soap.

Practical-7

Practical 7.1

Aim: Using the requests and BeautifulSoup Python libraries, print to the screen the fulltext of the article on this website: any news website.The article will be too long, so it is split up between 4 pages. Your task is to print outthe text to the screen so that you can read the full article without having to click anybuttons. This will just print the full text of the article to the screen. It will not make iteasy to read, so next exercise we will learn how to write this text to a .txt file.

Code:

```
import urllib.request
from bs4 import BeautifulSoup
url = "https://in.mashable.com/tech/2135/honor-view-20-review-i-dont-miss-my-oneplus-6t- anymore"
with urllib.request.urlopen(url) as uri: html = uri.read()
soup = BeautifulSoup(html)
# kill all script and style elements for script in soup(["script", "style"]):
script.extract()# rip it out
# get text
text = soup.get_text()
# break into lines and remove leading and trailing space on each lines = (line.strip() for line in text.splitlines())
# break multi-headlines into a line each
chunks = (phrase.strip() for line in lines for phrase in line.split(" ")) # drop blank lines

text = '\n'.join(chunk for chunk in chunks if chunk)
print(text)
```

Output:

```
Honor View 20 Review: I don't miss my OnePlus 6T anymore - tech
Skip to main content
Mashable
Mashable Australia
Mashable Benelux
Mashable India
Mashable Italia
Mashable Middle East
Mashable Pakistan
Mashable SE Asia
Mashable UK
CoronavirusEntertainmentCultureTechScienceMobilitySocial Good
Mashable
Mashable Australia
Mashable Benelux
Mashable India
Mashable Italia
Mashable Middle East
Mashable Pakistan
Mashable SE Asia
Mashable UK
Search
+
tech
Honor View 20 Review: I don't miss my OnePlus 6T anymore
+
By
Aakash Jhaveri
1 year, 1 month
With a new year in tow, we have exciting new technologies and innovation to look forward to, and smartphones get to experience these first hand. 2018 was all about the OnePlus 6T, with it becoming the best-seller in the premium segment. After using that phone extensively, and making it my daily driver, I have no doubts over why it did. With the View 20, Honor is tryin g to grab some of the market from OnePlus, with a phone that makes extremely bold decisions. Here’s why you need to pay attention to the gorgeous Honor View 20.
The Honor View 20 is priced at INR 37,999 for the 6GB RAM+128GB variant, and 45,999 for the 8GB RAM+256GB variant. At that price, it competes with the fan favorite OnePlus 6T, as well a s other older flagships that have faced a price cut. Its predecessor, the Honor View 10 was also aimed at the then champion, the OnePlus 5T but barely close to it. But with the View 20, Honor has done an excellent job at making a real competitor that even tips the 6T and brings features that were previously unheard of.
Love at first sight
One of the first things you will notice upon unboxing, which was seconded by dozens of my co-commuters, is that this phone is a sexy looker. And if you opt for the Red or the Blue varia nt, phew! It's gorgeous. On the back, you get these sharp lines that shine in neon colors when light strikes to reveal an interesting V-pattern. Honestly, it looks more classy than tack y, and more premium than the price tag would suggest. The glass is not as slippery as you would believe, having a look at the images. If you are scared of scratching up the non-Gorilla glass-covered back panel, use the included case for some added peace of mind.
A display that will captivate you
The front is predominantly covered by the screen, which is one of the major talking points of this phone. The Honor View 20 is one of the only commercially available phones, for now, to sport a punch-hole display, where instead of a notch, the front camera takes up only the bare minimum amount of space needed. It looks cool and functionally frees up a lot of space, off ering almost just the display. I had concerns over the scaling and adaptability of this abyss in my screen, but had no issues for the most part. I'm glad to report that PUBG scaled perf ectly too, giving some much coveted extra screen on the sides.
As for the viewing experience, the LCD panel with a resolution of 2310x1080 was more than adequate. Sharpness and color accuracy were high, but if you are coming from an OLED display, t his LCD might look a little unsaturated. In absolute terms, there's nothing to complain about. The screen has a brilliant coating which lends a perfect balance between friction and smoo thness. Swiping across this 6.4-inch slab of glass was a joy. The display also stays reasonably bright, and I found no need to crank it up to the max.
```

Practical 7.2

Aim: Write a function that takes an ordered list of numbers (a list where theelements are in order from smallest to largest) and another number. Thefunction decides whether or not the given number is inside the list andreturns (then prints) an appropriate boolean.

Code:

```
def in_list(list,s):
    min=0
    max=len(list)-1
    while(min<=max):
        mid = int((min+max) / 2)
        if(list[mid] == s):
            return True
        if list[mid] < s:
            min = mid+1
        else:
            max = mid-1
    return False
print (in_list([1,2,3,4,5,8],4))
print (in_list([1,2,3,4,5,8],7))
print (in_list([1,2,3,4,5,8],2))
print (in_list([1,2,3,4,5,8],9))
```

Output:

```
True
False
True
False
```

Extra : Use Binary Search

Code :

```
def find(ordered_list, element_to_find):
        start_index = 1
        end_index = len(ordered_list) - 1
        while True:
                middle_index = (end_index - start_index) / 2
        if middle_index < start_index or middle_index > end_index or middle_index < 0:
                return False
        middle_element = ordered_list[middle_index]
        if middle_element == element_to_find:
                return True
        elif middle_element < element_to_find:
                end_index = middle_index
        else:
                start_index = middle_index
```

```
if __name__=="__main__":
        l = [2, 4, 6, 8, 10, 12]
        print(find(l, 5))
        print(find(l, 10))
        print(find(l, -1))
        print(find(l, 2))
```

Output:

```
False
True
False
True
```

Practical 7.3

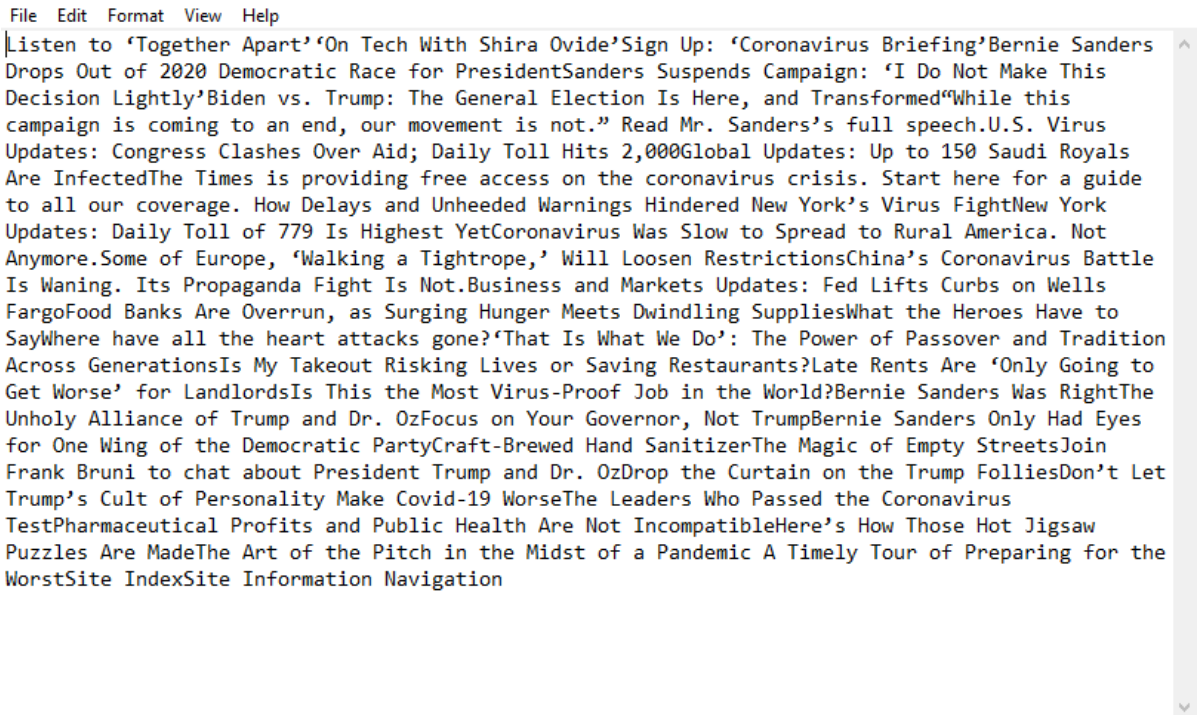
Aim: Take the code from the How to Decode A Website exercise, and instead ofprinting the results to a screen, write the results to a txt file. In your code,just make up a name for the file you are saving to.(Extra: Ask the user to specify the name of the output file that will be saved.)

Code:

```
import requests
from bs4 import BeautifulSoup
source = requests.get("https://www.nytimes.com").text
def get_title(text):
   n=input(text)
   return str(n)

soup = BeautifulSoup(source, 'lxml')
with open(get_title('What do you want to name the file?'), 'w') as open_file:
   for article in soup.find_all('h2'):
      open_file.write(str(article.text))
```

Output:



Practical 7.4

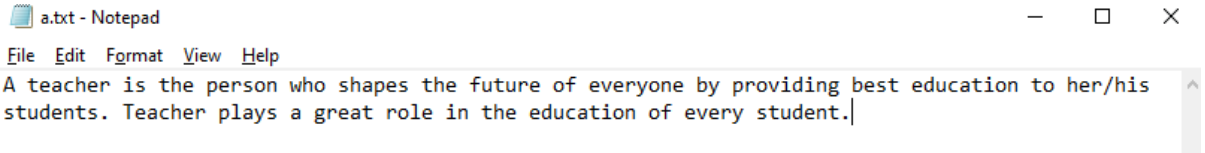
Aim: Given a .txt file that has a list of a bunch of names, count how many of eachname there are in the file, and print out the results to the screen.

Code:

```
count = dict()
with open("a.txt",'r') as f :
    x=f.read()
    y=x.split()
    for i in y :
        count[i]=0
    for i in y :
        count[i]+=1


for key,val in count.items():
    print (key, "=>", val)
```

Output:



```
A => 1
teacher => 1
is => 1
the => 3
person => 1
who => 1
shapes => 1
future => 1
of => 2
everyone => 1
by => 1
providing => 1
best => 1
education => 2
to => 1
her/his => 1
students. => 1
Teacher => 1
plays => 1
a => 1
great => 1
role => 1
in => 1
every => 1
student. => 1
```

Extra: Instead of using the .txt file from above (or instead of, if you want the challenge), take this .txt file, and count how many of each "category" of each image there are. This text file is actually a list of files corresponding to the SUN database scene recognition database, and lists the file directory hierarchy for the images. Once you take a look at the first line or two of the file, it will be clear which part represents the scene category. To do this, you're going to have to remember a bit about string parsing in Python 3. I talked a little bit about it in this post.

Code:

```
counter_dict = {}
with open('Desktop\Training.txt') as f:
    line = f.readline()
    while line:
        line = line[3:-26]
        if line in counter_dict:
            counter_dict[line] += 1
        else:
            counter_dict[line] = 1
        line = f.readline()


for key in counter_dict:
```

```
    print(key + ":" + str(counter_dict[key]))
```

Output:

```
abbey:50
airplane_cabin:50
airport_terminal:50
alley:50
amphitheater:50
amusement_arcade:50
amusement_park:50
anechoic_chamber:50
apartment_building/outdoor:50
apse/indoor:50
aquarium:50
aqueduct:50
arch:50
archive:50
arrival_gate/outdoor:50
art_gallery:50
art_school:50
art_studio:50
assembly_line:50
```

Conclusion: In this practical we learned different concepts regarding the scrapping of text from the website, use of function for finding the elements in a list. We also learned the different concepts of how to write the contents of a website into a text file and also how to calculate the number of letter of a text