coding and theory

1. What are the key features of Python?

2. What are the Data Types in Python?

1. Numeric Types:

- **int (integers):**
  - These represent whole numbers, both positive and negative, without any decimal points.
  - Example: `10`, `5`, `1000`.
- **float (floating-point numbers):**
  - These represent real numbers with decimal points.
  - Example: `3.14`, `0.001`, `2.5`.
- **complex (complex numbers):**
  - These represent numbers with a real and an imaginary part.
  - Example: `3 + 2j`, `1j`.

2. Sequence Types:

- **str (strings):**
  - These represent sequences of characters.
  - They are immutable (cannot be changed after creation).
  - Example: `"Hello"`, `'Python'`, `"123"`.
- **list (lists):**
  - These are ordered, mutable sequences of items.
  - They can contain items of different data types.
  - Example: `[1, 2, "three"]`, `[ ]`.
- **tuple (tuples):**

    - These are ordered, immutable sequences of items.
    - They are similar to lists, but their elements cannot be modified.
    - Example: `(1, 2, "three")`, `()`.
- **range (ranges):**
    - These represent immutable sequences of numbers. They are commonly used for looping a specific number of times.

3. Mapping Type:

- **dict (dictionaries):**
    - These are unordered collections of key-value pairs.
    - Keys must be immutable, and values can be of any data type.
    - Example: `{"name": "John", "age": 30}`.

4. Boolean Type:

- **bool (booleans):**
    - These represent truth values: `True` or `False`.
    - Booleans are often the result of logical operations.

5. Set Types:

- **set (sets):**
    - These are unordered collections of unique elements.
    - They do not allow duplicate values.
    - Example: `{1, 2, 3}`, `{"apple", "banana"}`.
- **frozenset (frozen sets):**
    - These are immutable versions of sets.

6. Binary Types:

- **bytes:** Immutable sequence of bytes.
- **bytearray:** mutable sequence of bytes.

- **memoryview:** A memory view object lets you access the internal data of an object that supports the buffer protocol without copying.

3. What are local variables and global variables in Python?

1. Local Variables:

- **Definition:**
  - Local variables are defined inside a function.
  - Their scope is limited to the function in which they are created.
- **Characteristics:**
  - They exist only while the function is executing.
  - They cannot be accessed from outside the function.
  - They help encapsulate data within a specific function, reducing the risk of unintended modifications.

2. Global Variables:

- **Definition:**
  - Global variables are defined outside of any function.
  - They have a global scope, meaning they can be accessed from anywhere in the program.
- **Characteristics:**
  - They exist for the duration of the program.
  - They can be accessed by any function.
  - However, modifying global variables within a function requires the `global` keyword.

4. How do you write comments in python? And Why Comments are important?

Code Clarity and Readability:

- Comments explain the purpose of code sections, making it easier for yourself and others to understand what the code does.

- They can clarify complex logic or algorithms.

5. How to comment on multiple lines in python?

   KeyBoard shortcut   ctrl +/   to comment or uncomment

   """ """   ''' '''

6. What do you mean by Python literals?

- **Numeric Literals:**
  - **Integers:** (e.g., `10`, `5`, `0`)
  - **Floating-point numbers:** (e.g., `3.14`, `0.001`, `2.5`)
  - **Complex numbers:** (e.g., `3 + 2j`)
- **String Literals:**
  - Sequences of characters enclosed in quotes (e.g., `"Hello"`, `'Python'`, `"""Multi-line string"""`)
- **Boolean Literals:**
  - `True` and `False`
- **Literal Collections:**
  - **List literals:** (e.g., `[1, 2, 3]`)
  - **Tuple literals:** (e.g., `(1, 2, 3)`)
  - **Dictionary literals:** (e.g., `{"key": "value"}`)
  - **Set literals:** (e.g., `{1, 2, 3}`)
- **Special Literal:**
  - `None` (represents the absence of a value)

7. What are different ways to assign value to variables?

Simple Assignment:

x = 10
name = "Python"

is_valid = True

Multiple Assignments:

a = b = c = 0

x, y, z = 1, 2, 3

Augmented Assignment Operators:

x = 5
x += 2   Equivalent to x = x + 2 (x becomes 7)
x -= 3   Equivalent to x = x - 3 (x becomes 4)
x *= 4   Equivalent to x = x * 4 (x becomes 16)
x /= 2   Equivalent to x = x / 2 (x becomes 8.0)
x //= 3   Equivalent to x = x // 3 (x becomes 2)
x %= 3   Equivalent to x = x % 3 (x becomes 2)
x **= 2   equivalent to x = x ** 2 (x becomes 4)

8. What are the Escape Characters in python?

Escape characters in Python are special sequences of characters used within string literals to represent characters that are difficult or impossible to type directly. They consist of a backslash (`\`) followed by another character.

examples

\n

\t

\\

\'

\"

9. which are the different ways to perform string formatting? Explain with example.

Python offers several ways to format strings, each with its own advantages and use cases. Here are the primary methods:

1. Percent-Style Formatting (Legacy):

- This is the oldest method, similar to `printf` in C.
- It uses the `%` operator to insert values into placeholders within a string.
- While still functional, it's generally considered less readable and less powerful than newer methods.

```
name = "Alice"
age = 30
print("My name is %s and I am %d years old." % (name, age))
```

Output: My name is Alice and I am 30 years old.

2. `str.format()` Method:

- This method provides more flexibility and readability than percent-style formatting.
- It uses curly braces `{}` as placeholders, which can be implicitly or explicitly numbered.
- It also allows for keyword arguments and formatting specifications.

```
name = "Charlie"
age = 35
print("My name is {} and I am {} years old.".format(name, age))
```

Output: My name is Charlie and I am 35 years old.

3. F-strings (Formatted String Literals):

- Introduced in Python 3.6, f-strings are the most concise and readable way to format strings.
- They use an `f` or `F` prefix before the opening quote, and expressions within curly braces `{}` are evaluated at runtime.

```
name = "Eve"
age = 28
print(f"My name is {name} and I am {age} years old.")
```

Output: My name is Eve and I am 28 years old.

10. write a program to print every character of a string entered by the user in a new line using a loop

```
string = input()
for i in range(0,len(string)):
    print(string[i])
```

11. write a program to find the length of the string "machine learning" with and without using len function.

```
string = "machine learning"

# with len function
print(len(string))

# without len function
count =0
for i in string:
    count+=1
print(count)
```

12. write a program to check if the word 'orange' is present in the "This is orange juice".

```
string = "This is orange juice"
list1 = list(string.split(' '))
word = 'orang'
if word in list1:
    print(f"{word} is present in {string}")
else:
    print(f"{word} is not present in {string}")
```

13. Write a program to find the number of vowels, consonants, digits, and white space characters in a string.

```
import re
string = "This is orange juice 123"
string = string.lower()
list1 = ['a','e','i','o','u']
vowel_count = 0
digitcount = 0
spaces_count = 0
for i in string:
    if re.search(r"[0-9]", i):
        digitcount+=1
    elif i in list1:
        vowel_count+=1
    elif i == " ":
        spaces_count+=1
print(f"vowelscount : {vowel_count} \ndigitscount : {digitcount} \nspacescount :
```

14. Write a Python program to count Uppercase, Lowercase, special character, and numeric values in a given string.

```
import re
import string
string = "%This is orange juice 123 %"
uppercaseletters_count = 0
digitcount = 0
lowercaseletters_count = 0
spaces_count =0
specialchars_count = 0
for i in string:
    if re.search(r"[0-9]", i):
        digitcount+=1
    elif i.islower():
        lowercaseletters_count+=1
    elif i.isupper():
        uppercaseletters_count+=1
    elif re.search(r"[^\s]",i):
        specialchars_count+=1

print(f"lowercase : {lowercaseletters_count} \ndigitscount : {digitcount} \nupperc
```

15. Write a program to make a new string with all the consonants deleted from the string "Hello, have a good day".

```
string = "Hello, have a good day"
string = list(string)
list1 = ['a','e','i','o','u']
i= 0
while i < len(string):
    if string[i] in list1:
        i+=1
    else:
        string.remove(string[i])
string = "".join(string)
print(string)
```

16. Write a Python program to remove the nth index character from a non-empty string.

```
string = input()
n_index = int(input())
string = string[0:n_index] + string[n_index+1:]
print(string)
```

17. Write a Python program to change a given string to a new string where the first and last characters have been exchanged.

```
s = input()
print(s[len(s)-1]+s[1:len(s)-2]+s[0])
```

18. Write a Python program to count the occurrences of each word in a given sentence.

```
s = list(map(str,input().split(" ")))
word_counts ={}
for i in s:
    if i in word_counts:
        word_counts[i] +=1
    else:
        word_counts[i] =1
print(word_counts)
```

19. How do you count the occurrence of a given character in a string?

```
s = input()
char_counts ={}
for i in s:
    if i in char_counts:
        char_counts[i] +=1
    else:
        char_counts[i] =1
print(char_counts)
```

20. Write a program to find last 10 characters of a string?

```
s = input()
k = s[len(s)-11::1]
if len(s)>=10:
    print(k)
else:
    print("provide valid string with lenghth greater than 10")
```

21. WAP to convert a given string to all uppercase if it contains at least 2 uppercase characters in the first 4 characters.

```
s = input()
k = s[:5]
```

```
count = 0
if len(s)>=2:
    for i in k:
        if i.isupper():
            count+=1
    if count>=2:
        print(s.upper())
    else:
        print("First Four characters are not atleast 2 upper case letters")
else:
    print("provide valid string with lenghth greater than 1")
```