Q.1 str='Kavya' output:avK

```
str="Kavya"

str[-1::-2]
```

```
'avK'
```

Q.2 // and / difference

```
a=5/2
a
```

```
2.5
```

```
b=5//2
b
```

```
2
```

Q.3 Swapcase

```
sc="ASjdsfefef"
sc.swapcase()
```

```
'asJDSFEFEF'
```

Q.4 Can we do tuple Comprehension?If not why?

Ans:No,Tuple comprehension isn't possible in Python because tuples are immutable, means once they're created, their elements cannot be changed.

Q.5 sort and sorted Difference

Ans: sort:- sort will sort the original list

sorted:-sorted will create the new list and will sort that new list

Q.6 Current time Stamp

```
import time
import datetime
date=datetime.datetime.now()
time.time()
```

```
1716533357.72789
```

```
date
```

```
datetime.datetime(2024, 5, 24, 12, 19, 17, 727890)
```

Q.7 Can function be call as a argument to another function?

Ans:Yes function can be call as a argument to another function and it is called as Higher Order Functions.

Q.8 format string and raw string example?

```
a='rahul'

#Format string
print(f'{a}')
```

```
rahul
```

```
#Raw String
print(r"{a}")
```

```
{a}
```

Q.9 Generators vs Decorators?

Ans: Decorators are functions that modify the behavior of other functions or methods.

Generators are often used to efficiently generate large sequences of values without loading them all into memory at once.use for large datasets.

Q.10 Function Annotations?

Ans:Function annotations in Python allow you to specify the types of function parameters and return values.

```
#Example
def greet(name: str) -> str:
    return "Hello, " + name

print(greet("Alice"))
print(greet.__annotations__)
```

```
Hello, Alice
{'name': 'Kavya', 'return': 'Kavya'}
```

Q.11 Walrus Operation?

Ans:

":=" it is use for assignment operation.

```
if (x := 5) > 3:
    print("x is greater than 3")
```

```
x is greater than 3
```

Q.12 Initialize the Dictionary with default value?

Ans: my_dict = {}

default_value = 0

value = my_dict.get('key', default_value)

Q.13 Pandas Head Vs Tail?

Ans: head() returns the first n rows i.e top rows from the dataframe

tail() returns the last n rows i.e bottom rows from the dataframe

Q.14 Pandas Multiple Indexes?

Ans:Pandas supports multiple indexes, known as MultiIndex, which enables working with higher-dimensional data in DataFrame objects.

```
import pandas as pd

df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5,
6]},index=pd.MultiIndex.from_tuples([('a', 1), ('a', 2), ('b', 1)],
names=['letter', 'number']))

df
```

```
               A  B
letter number
a      1       1  4
       2       2  5
b      1       3  6
```

Q.15 Pandas reindex?

Ans:Pandas reindex is a method used to change the row or column labels of a DataFrame.

#Example: me df_reindexed = df.reindex(['a', 'b', 'c])

Q.16 Pandas Merge vs Join?

Ans: Merge: Flexible function allowing different types of joins based on specified key columns.
join : Specific case of merg where the join operation is based on DataFrame indexes.

Q.17 How to Optimize in Pandas?

Ans:Using techniques like vectorization, using appropriate data types, avoiding unnecessary copying of data, and utilizing built-in Pandas methods for common tasks.

Q.18 What is TimeDelta in Pandas?

Ans:Timedelta in Pandas represents differences in times, allowing for operations on durations.

timedelta()

Q.19 Pandas Concat vs Append?

Ans: concat : Combines multiple DataFrames along a specified axis, offering flexibility in concatenation options.

Example:a=firstname,b=lastname

pd.concat([a, b])w2) append :: Adds rows from one DataFrame to another, simplifying concatenation along the row

append_a = a.append(b)axis.

Q.20 Rolling Mean in Pandas?

Ans:Rolling mean in Pandas calculates the mean of a specified window of consecutive values in a Series or DataFrame, moving through the data.()

```
import pandas as pd

data = {'value': [1, 2, 3, 4, 5]}
df = pd.DataFrame(data)

rolling_mean = df['value'].rolling(window=2).mean()

print(rolling_mean)
```

```
0    NaN
1    1.5
2    2.5
3    3.5
4    4.5
Name: value, dtype: float64
```

Q.21 How to do Sum operation in Pandas?

Ans: Using sum() method.

```
import pandas as pd

data = {'A': [1, 2, 3, 4, 5]}
df = pd.DataFrame(data)

column_sum = df.sum()

row_sum = df.sum(axis=1)

print("Column sum:", column_sum)
print("Row sum:", row_sum)
```

```
Column sum: A    15
dtype: int64
Row sum: 0    1
1    2
2    3
```

```
3    4
4    5
dtype: int64
```

Q.22 Pickling In Python?

Ans: Pickling in Python is the process of serializing Python objects into a byte stream for storage or transmission.

Q.23 Pandas Group By using any column and Sort By Using Multiple Columns?

```
import pandas as pd

# Example DataFrame
data = {'group': ['A', 'B', 'A', 'B', 'A'],
        'value1': [1, 2, 3, 4, 5],
        'value2': [6, 7, 8, 9, 10]}
df = pd.DataFrame(data)

# Grouping by 'group' column and sorting by 'group' and 'value1'
sorted_group = df.groupby('group').apply(lambda x:
x.sort_values(by=['value1', 'value2'])).reset_index(drop=True)

print(sorted_group)
```