SETS --> Python set is a well defined collection of distinct objects.These are stored in curlly brackets i.e.{}.They are independent of order and immutable in nature.There are two types of sets like frozen sets.They can be constructed by simple form i.e. using {} and by using constructor.

Operations--> 1.Intersection 2.Intersection_update 3.difference 4.differnce_update 5.symmetric_difference 6.symmetric_difference_update 7.union 8.Update 9.issubset 10.issuperset 11.isdisjoint 12.isproperset(<)

```
#intersection

set1={1,2,3,4,5}
set2={4,5,6,7,8}
print(set1.intersection(set2))

#update

set1.update(set2)
print(set1)

#intersection_update
set3={9,7,3,4}
set4={4,7,6,3}
set3.intersection_update(set4)
print(set3)

#symmetric_differnce

set1={1,2,3,4,5}
set2={4,5,6,7,8}
print(set1.symmetric_difference(set2))


#symmetric_difference_update

set1={1,2,3,4,5}
set2={4,5,6,7,8}
set1.symmetric_difference_update(set2)
print(set1)


#difference

set1={1,2,3,4,5}
set2={4,5,6,7,8}
print(set1.difference(set2))

#differene_update
```

```
set1={1,2,3,4,5}
set2={4,5,6,7,8}
set1.difference_update(set2)
print(set1)


#union

set1={1,2,3,4,5}
set2={4,5,6,7,8}
print(set1.union(set2))


#issuperset
set1={1,2,3,4}
set2={1,2,3,4,5,6}
print(set2.issuperset(set1))

#issubset
set1={1,2,3,4}
set2={1,2,3,4,5,6}
print(set1.issubset(set2))

#isdisjoint
set1={1,2,3,4}
set2={5,6,7,8}
print(set1.isdisjoint(set2))

#isproperset

set1={1,2,3,4}
set2={1,2,3,4,5,6}
print(set1<set2)

#add
set1={1,2,3}
set1.add(5)
print(set1)

#discard
set1.discard(5)
print(set1)

#remove
set1={1,2,3,4,5}
set1.remove(5)
print(set1)
```

```
{4, 5}
{1, 2, 3, 4, 5, 6, 7, 8}
```

```
{3, 4, 7}
{1, 2, 3, 6, 7, 8}
{1, 2, 3, 6, 7, 8}
{1, 2, 3}
{1, 2, 3}
{1, 2, 3, 4, 5, 6, 7, 8}
True
True
True
True
{1, 2, 3, 5}
{1, 2, 3}
{1, 2, 3, 4}
{frozenset({3, 4}), frozenset({1, 2})}
```

1. Create a set with the first five prime numbers.

```
set1={2,3,5,7,11}
print(set1)

{2, 3, 5, 7, 11}
```

1. Add the number 7 to the set.

```
set1={2,3,4,5}
set1.add(7)
print(set1)

{2, 3, 4, 5, 7}
```

1. Remove the number 3 from the set.

```
set1={2,3,4,5}
set1.remove(3)
print(set1)

{2, 4, 5}
```

1. Check if the set is a subset of {2, 3, 5, 7, 11}.

```
set1={2,3,4,5}
print(set2.issubset( {2, 3, 5, 7, 11}))

False
```

1. Find the union of the set with {7, 11, 13}.

```
set1={2,3,4,5}
set2={7,11,13}
print(set1.union(set2))

{2, 3, 4, 5, 7, 11, 13}
```

1. Create a frozen set from the original set.

```
set1={2,3,4,5}
print(frozenset(set1))

frozenset({2, 3, 4, 5})
```

Check if the set has any common elements with {1, 4, 9}.

```
print({1,2,3}.isdisjoint({1,4,9}))

False
```

Remove all elements from the set.

```
set1={2,3,4,5}
set1.clear()
print(set1)

set()
```

1. Create a set of your favorite fruits and find the intersection with { 'apple', 'banana', 'orange' }.

```
set1={'apple','banana','orange'}
set2={'apple','mango','orange'}
print(set1.intersection(set2))

{'orange', 'apple'}
```

Use set comprehension to create a set of squares for numbers 1 to 10.

```
squares = {x**2 for x in range(1, 11)}
print(squares)

{64, 1, 4, 36, 100, 9, 16, 49, 81, 25}
```

Dictionaries-->Python dictionaries are set of key value pairs and are unordered in nature.It can be implemented by {}. Keys can be used to identify specified values .A representation of {} without any element in it represent dictionary.Mapping in python means dictionary in python.

RULES: 1.Every key must be unique 2.Every key must be hashable

It can be constructed simply by using curly braces or by using constructor.

Method: 1 .get() 2 .items() 3 .clear() 4 .keys() 5 .values() 6 .update() 7 .pop() 8 .popitem()

1. Basic Dictionary Operations: Create a dictionary named student with keys "name," "age," and "grade," and assign appropriate values. Print the value associated with the "age" key.

Update the "grade" to a new value. Add a new key-value pair for "subject" and its corresponding value.

```
dic={"name":"mehak","age":19,"grade":10}
print(dic)
print(dic["age"])
dic["age"]=20
print(dic)
dic["subject"]="Chem"
print(dic)

{'name': 'mehak', 'age': 19, 'grade': 10}
19
{'name': 'mehak', 'age': 20, 'grade': 10}
{'name': 'mehak', 'age': 20, 'grade': 10, 'subject': 'Chem'}
```

1. Dictionary Manipulation: Create two dictionaries, dict1 and dict2, with at least three key-value pairs each. Merge these dictionaries into a new dictionary called merged *dict. Remove a key from merged* dict. Check if a specific key exists in dict1.

```
dic1={"name1":"mehak","age1":19,"grade1":10}
dic2={"name2":"jacky","age2":10,"grade2":11}
dic3=dic1.copy()
print(dic3)
dic3.update(dic2)
print(dic3)
dic3.pop("grade1")
print(dic3)
key="grade2"
if key in dic3:
  print("Present")
else:
  print("error")

{'name1': 'mehak', 'age1': 19, 'grade1': 10}
{'name1': 'mehak', 'age1': 19, 'grade1': 10, 'name2': 'jacky', 'age2':
10, 'grade2': 11}
{'name1': 'mehak', 'age1': 19, 'name2': 'jacky', 'age2': 10, 'grade2':
11}
Present
```

1. Iterating Through a Dictionary: Use a loop to print all keys in the student dictionary. Use another loop to print all values in the student dictionary. Write a loop to print each key-value pair in the student dictionary.

```
dic1={"name1":"mehak","age1":19,"grade1":10}
for key,value in dic1.items():
  print(f"The specifid {key} have value {value}")
```

```
The specifid name1 have value mehak
The specifid age1 have value 19
The specifid grade1 have value 10
```

1. Dictionary Comprehension: Create a dictionary comprehension to generate a dictionary of squares from 1 to 10. Filter the above dictionary to include only even squares.

```
dic1={x**2 for x in range(1,11)}
for ele in dic1:
  if (ele%2==0):
    print(ele)
```

```
64
4
36
100
16
```

1. Nested Dictionaries: Create a dictionary named school with multiple students (nested dictionaries). Access and print information about a specific student within the school dictionary.

```
school={"child":{
    "name":"mehak",
    "age":19,
    "grade":10
},"hobby":{
        "name":"cricket",
        "level":"champion"
    },"diet":{
        "meal":"Omelete",
        "drink":"protien shake"

        }}

print(school)
```

```
{'child': {'name': 'mehak', 'age': 19, 'grade': 10}, 'hobby': {'name':
'cricket', 'level': 'champion'}, 'diet': {'meal': 'Omelete', 'drink':
'protien shake'}}
```

1. Dictionary Functions: Use the len() function to find the number of items in a dictionary. Use the keys(), values(), and items() methods on a dictionary and print the results.

```
dic1={"name":"mehak","age":"19","hobby":"coding"}
print(len(dic1))
print(dic1.keys())
print(dic1.values())
print(dic1.items())
```

```
3
dict_keys(['name', 'age', 'hobby'])
dict_values(['mehak', '19', 'coding'])
dict_items([('name', 'mehak'), ('age', '19'), ('hobby', 'coding')])
```

1. Sorting a Dictionary: Create a dictionary with unsorted keys. Use the sorted() function to print the keys in alphabetical order.

```
dic1={"name":"mehak","age":"19","hobby":"coding"}
print(sorted(dic1))

['age', 'hobby', 'name']
```

1. Dictionary Methods: Use the get() method to retrieve a value with a default if the key doesn't exist. Use the pop() method to remove and return a specific key-value pair.

```
dic1={"name":"mehak","age":"19","hobby":"coding"}
key=input("Enter the value: ")
if key in dic1:
  print(dic1.get(key))
else:
  print("error")

Enter the value: name
mehak
```

PRACTICE QUESTIONS -->>

1. You have two sets representing students enrolled in Math and Science classes. Find the set of students who are enrolled in both classes.

```
set1={1,2,3,4,5}
set2={4,5,6,7,8}
print(set1.intersection(set2))

{4, 5}
```

. Given a set of unique employee IDs, add a new employee ID to the set.

```
set1={"1234","2345","4567","6789"}
set1.add("9876")
print(set1)

{'9876', '6789', '2345', '4567', '1234'}
```

3. You are managing two different projects and have two sets representing the team members for each project. Find the set of team members who are working on either project or both projects.

```
set1={"neha","girish","falgun","jacky"}
set2={"bob","jacky","olivia","cocroach"}
print(set1.union(set2))

{'falgun', 'neha', 'jacky', 'bob', 'girish', 'cocroach', 'olivia'}
```

4. You have a set of customer IDs who have made purchases this month. Remove a customer ID from the set who has canceled their order.

```
CUSTOMER_ID={1,2,3,4,5}
CUSTOMER_ID.remove(4)
print(CUSTOMER_ID)

{1, 2, 3, 5}
```

5. Given two sets of product IDs, one representing products in stock and the other representing products that have been sold, find the set of products that are still in stock.

```
set1={1,2,3,4,5}
set2={4,5,6,7,8}
print(set1.difference(set2))

{1, 2, 3}
```

6. You have a set of registered conference attendees. Check if a particular person is registered.

```
set1={"mehak","jacky","bob","cocroach"}
key=input("Enter the value: ")
if key in set1:
  print("Present")
else:
  print("error")

Enter the value: cocroach
Present
```

7. Given two sets of book titles, one representing books available in the library and the other representing books borrowed by students, find the set of books that are not currently available in the library.

```
set1={"maths","chem","physics","biology","hindi","punjabi"}
set2={"maths","chem","physics"}
print(set1.intersection(set2))

{'physics', 'maths', 'chem'}
```

8. You have a set of unique tags assigned to a blog post. Add multiple new tags to the set at once.

```
tags = {"python", "coding", "data science"}
new_tags = {"machine learning", "deep learning"}
tags.update(new_tags)
print(tags)

{'deep learning', 'coding', 'data science', 'python', 'machine
learning'}
```

9. Given a set of employee skills and another set of skills required for a new project, find the set of skills that need to be acquired or improved.

```
skill1={"web","c++","cyber","photoshop"}
skill2={"AI","Mern","python"}
set3=skill1.difference(skill2)
print(set3)

{'web', 'photoshop', 'cyber', 'c++'}
```

10. You have two sets of favorite movies from two different friends. Find the set of movies that are favorite to at least one of them but not both.

```
friend1 = {"yo", "bo", "so", "no"}
friend2 = {"lo", "bo", "so", "go"}
not_both = friend1.symmetric_difference(friend2)
print(not_both)

{'lo', 'go', 'yo', 'no'}
```

11. Given a set of emails collected from a signup form, ensure that the set contains only unique emails by checking the length before and after adding more emails.

```
# prompt: 11. Given a set of emails collected from a signup form,
ensure that the set contains only unique emails by checking the length
before and after adding more emails.

# Create a set to store unique emails
email_set = set()

# Get the initial length of the set
initial_length = len(email_set)

# Add some new emails to the set
email_set.add("john.doe@example.com")
email_set.add("jane.doe@example.com")

# Get the length of the set after adding emails
updated_length = len(email_set)

# Check if the length has changed
```

```
if initial_length == updated_length:
    print("Duplicate emails detected.")
else:
    print("All emails are unique.")

All emails are unique.
```

12. You have a set of courses you are interested in and another set of courses you have completed. Find the set of courses you are still interested in but have not yet completed.

```
interested_courses = {"maths", "physics", "computer science"}
completed_courses = {"maths", "chemistry"}

courses_still_interested =
interested_courses.difference(completed_courses)

print(courses_still_interested)

{'physics', 'computer science'}
```

13. Given a set of cities visited last year and another set of cities visited this year, find the set of cities that were visited in either year but not both.

```
last={"ludhiana","delhi","lonavala","patiala","jalandhar"}
now={"ropar","amritsar","ludhiana","patiala","moga","bathinda"}
print(last.symmetric_difference(now))

{'jalandhar', 'amritsar', 'moga', 'lonavala', 'bathinda', 'delhi',
'ropar'}
```

15. Given two sets representing two different sports clubs' members, find the set of members who are exclusive to the first club.

```

club1 = {"Alice", "Bob", "Charlie", "David", "Eve"}
club2 = {"Bob", "Charlie", "Frank", "George", "Harry"}

exclusive_to_club1 = club1.difference(club2)

print(exclusive_to_club1)

{'David', 'Alice', 'Eve'}
```

You have a set of ingridients required for a recipe. Check if a specific ingredient is missing.

```
ingridient={"haldi","dahi","masla","mirch"}
check=input("Is thhere present the ingredient: ")
if check in ingridient:
  print("Present")
```

```
else:
  print("error")

Is thhere present the ingredient: mirch
Present
```

Given a set of friends on social media and another set of mutual friends with a colleague, find the set of mutual friends.

```
set1={"neha","rahul","nisha","neha"}
set2={"joy","jacky","naina"}
print(set1.intersection(set2))

set()
```

18. You have a set of items in your inventory and another set representing items that need restocking. Find the set of items that are currently in stock and need restocking.

```
in_stock = {"apple", "banana", "orange"}
need_restock = {"banana", "grape", "mango"}

items_to_restock = in_stock.intersection(need_restock)

print(items_to_restock)

{'banana'}
```

1. Given a set of languages spoken by employees in a company and another set of languages required for a new project, find the set of languages that are both spoken by employees and required for the project.

```


spoken_languages = {"English", "Spanish", "French", "German"}
project_languages = {"English", "French", "Japanese"}

languages_needed = spoken_languages.intersection(project_languages)

print(languages_needed)

{'English', 'French'}
```

1. You have a set of completed tasks and another set of tasks for a project. Find the set of tasks that are yet to be completed.

```


completed_tasks = {"Task 1", "Task 2", "Task 3"}
all_tasks = {"Task 1", "Task 2", "Task 3", "Task 4", "Task 5"}

yet_to_complete = all_tasks.difference(completed_tasks)
```

```
print(yet_to_complete)

{'Task 4', 'Task 5'}
```

1. You have a set of students who have submitted an assignment and another set of students who have attended a class. Find the set of students who have either submitted the assignment or attended the class.

```
submitted_students = {"Alice", "Bob", "Charlie"}
attended_class_students = {"Bob", "Charlie", "David"}

students_participated =
submitted_students.union(attended_class_students)

print(students_participated)

{'Bob', 'David', 'Alice', 'Charlie'}
```

Given two sets representing the collection of books in two different libraries, find the set of books that are available in both libraries.

```
library_1 = {"Book 1", "Book 2", "Book 3", "Book 4"}
library_2 = {"Book 2", "Book 3", "Book 5", "Book 6"}

common_books = library_1.intersection(library_2)

print(common_books)

{'Book 2', 'Book 3'}
```

Practice questions-->> . You have a dictionary representing the inventory of a store with item names as keys and quantities as values. Add a new item to the inventory with a specific quantity.

```
dic1={"dahi":20,"ghee":60,"mehndi":90,"doodh":80}
dic1["masala"]=100
print(dic1)

{'dahi': 20, 'ghee': 60, 'mehndi': 90, 'doodh': 80, 'masala': 100}
```

. Given a dictionary of student names and their scores, find the score of a particular student.

```
student_scores = {"Alice": 95, "Bob": 80, "Charlie": 75}

student_name = input("Enter the name of the student: ")
```

```
if student_name in student_scores:
  score = student_scores[student_name]
  print(f"{student_name}'s score is {score}")
else:
  print(f"{student_name} is not found in the dictionary.")
```

```
Enter the name of the student: alice
alice is not found in the dictionary.
```

1. You have a dictionary of country names as keys and their capitals as values. Update the capital of a specific country.

```
country_capitals = {"India": "New Delhi", "USA": "Washington D.C.",
"UK": "London"}

country = input("Enter the country name: ")
new_capital = input("Enter the new capital: ")

if country in country_capitals:
  country_capitals[country] = new_capital
  print(f"The capital of {country} has been updated to {new_capital}")
else:
  print(f"{country} is not found in the dictionary.")
```

```
Enter the country name: India
Enter the new capital: Chandigarh
The capital of India has been updated to Chandigarh
```

1. Given a dictionary of product prices with product names as keys, remove a product from the dictionary.

```
dic1={"dahi":20,"ghee":60,"mehndi":90,"doodh":80}
dic1.pop("dahi")
print(dic1)
```

```
{'ghee': 60, 'mehndi': 90, 'doodh': 80}
```

1. You have two dictionaries representing two different departments with employee names as keys and their salaries as values. Merge these two dictionaries into one.

```
library_1 = {"Book 1":1, "Book 2":2, "Book 3":3, "Book 4":4}
library_2 = {"Book 2":5, "Book 3":6, "Book 5":7, "Book 6":8}
library_1.update(library_2)
print(library_1)
```

```
{'Book 1': 1, 'Book 2': 5, 'Book 3': 6, 'Book 4': 4, 'Book 5': 7,
'Book 6': 8}
```

1. Given a dictionary of city names and their populations, check if a particular city is in the dictionary.

```
city_populations = {"New York City": 8336817, "Los Angeles": 3971883,
"Chicago": 2746388}

city_name = input("Enter the name of the city: ")

if city_name in city_populations:
  print(f"{city_name} is in the dictionary with a population of
{city_populations[city_name]}")
else:
  print(f"{city_name} is not found in the dictionary.")
```

```
Enter the name of the city: delhi
delhi is not found in the dictionary.
```

1. You have a dictionary of book titles as keys and their authors as values. Add multiple new books to the dictionary at onc

```
library_1 = {"Book 1":1, "Book 2":2, "Book 3":3, "Book 4":4}
library_2 = {"Book 2":5, "Book 3":6, "Book 5":7, "Book 6":8}
library_1.update(library_2)
print(library_1)
```

```
{'Book 1': 1, 'Book 2': 5, 'Book 3': 6, 'Book 4': 4, 'Book 5': 7,
'Book 6': 8}
```

1. Given a dictionary of students' names and their grades, find the student with the highest grade.

```
students_grades = {
    "Alice": 85,
    "Bob": 92,
    "Charlie": 87,
    "Diana": 100,
    "Eve": 78
}

max_grade=-1
for grade in students_grades.values():
  if grade>max_grade:
    max_grade=max(max_grade,grade)
print(max_grade)
```

```
100
```

1. You have a dictionary representing a phone book with names as keys and phone numbers as values. Remove a contact from the phone book.

```
phone_book = {"Alice": "555-1234", "Bob": "555-5678", "Charlie": "555-
9012"}

name = input("Enter the name of the contact to remove: ")

if name in phone_book:
  del phone_book[name]
  print(f"{name} has been removed from the phone book.")
  print(phone_book)
else:
  print(f"{name} is not found in the phone book.")
```

```
Enter the name of the contact to remove: Alice
Alice has been removed from the phone book.
{'Bob': '555-5678', 'Charlie': '555-9012'}
```

1. Given a dictionary of product names and their prices, calculate the total value of all the products in the dictionary.

```
products = {"apple": 2.50, "banana": 1.75, "orange": 3.00}

total_value = 0

for price in products.values():
  total_value += price

print(f"The total value of all products is ${total_value}")
```

```
The total value of all products is $7.25
```

1. You have a dictionary of movie titles as keys and their release years as values. Find all movies released before a specific year.

```
movies = {
    "The Shawshank Redemption": 1994,
    "The Godfather": 1972,
    "The Dark Knight": 2008,
    "12 Angry Men": 1957,
    "Schindler's List": 1993
}

year = int(input("Enter a year: "))
for title, release_year in movies.items():
  if release_year < year:
    print(title)
```

```
Enter a year: 1994
The Godfather
```

```
12 Angry Men
Schindler's List
```

Given a dictionary of students and their grades, create a new dictionary where the grades are keys and the values are lists of students who have that grade.

```python
students_grades = {
    "Alice": 85,
    "Bob": 92,
    "Charlie": 85,
    "Diana": 100,
    "Eve": 78
}

grade_students = {}


for student, grade in students_grades.items():

    if grade not in grade_students:
        grade_students[grade] = []

    grade_students[grade].append(student)

print(grade_students)
```

```
{85: ['Alice', 'Charlie'], 92: ['Bob'], 100: ['Diana'], 78: ['Eve']}
```

Given a dictionary of employee names and their department names, find all employees in a specific department.

```python
employees = {
    "John": "Marketing",
    "Jane": "Sales",
    "Bob": "Marketing",
    "Alice": "Finance",
    "Tom": "Sales"
}

department = "Marketing"

department_employees = []

for employee, department_name in employees.items():
  if department_name == department:
    department_employees.append(employee)

print(department_employees)
```

```
['John', 'Bob']
```

You have a dictionary of course names as keys and the number of enrolled students as values. Increase the number of enrolled students for a specific course by a given number.

```
courses = {
    "Math": 20,
    "Science": 15,
    "English": 25,
    "History": 30
}

course_name = "Math"
increase_by = 5

if course_name in courses:
    courses[course_name] += increase_by
    print(f"The value of course maths increases to 
{courses[course_name]}")
else:
    print(f"{course_name} is not found in the dictionary.")
```

```
The value of course maths increases to 25
```

Given a dictionary of book titles and their prices, create a list of all book titles that cost less than a specified amount.

```
book= {
    "Math": 20,
    "Science": 15,
    "English": 25,
    "History": 30
}
book_list=[]
for key,value in book.items():
  if value<30:
    book_list.append(key)
print(book_list)
```

```
['Math', 'Science', 'English']
```

You have a dictionary of country names as keys and their population as values. Sort the dictionary by population in ascending order.

```
country_population = {
    "China": 1_439_323_776,
    "India": 1_380_004_385,
    "United States": 331_002_651,
    "Indonesia": 273_523_615,
    "Pakistan": 220_892_340
}

sorted_country_population=dict(sorted(country_population.items(),key=l
ambda item:item[1]))
print(sorted_country_population)
```

```
{'Pakistan': 220892340, 'Indonesia': 273523615, 'United States':
331002651, 'India': 1380004385, 'China': 1439323776}
```

Given a dictionary of employee names and their salaries, create a new dictionary with only those employees who earn more than a specified amount.

```
employee = {
    "Alice": 8500,
    "Bob": 9200,
    "Charlie": 8500,
    "Diana": 10000,
    "Eve": 7800
}
new_dic={}
for key,value in employee.items():
  if value>8000:
    new_dic[key]=value
print(new_dic)
```

```
{'Alice': 8500, 'Bob': 9200, 'Charlie': 8500, 'Diana': 10000}
```

You have a dictionary of product names as keys and their quantities as values. Check if the inventory is empty and print an appropriate message.

```
dic1={"dahi":20,"ghee":60,"mehndi":90,"doodh":80}
if len(dic1)==0:
  print("inventory is empty")
else:
  print("inventory is not empty")
```

```
inventory is not empty
```

You have a dictionary of student names and their test scores. Create a new dictionary where the keys are the test scores and the values are lists of students who achieved those scores.

```
students_grades = {
    "Alice": 85,
    "Bob": 92,
    "Charlie": 85,
    "Diana": 100,
    "Eve": 78
}

grade_students = {}
for key,value in students_grades.items():
  if value not in grade_students:
    grade_students[value]=[]
  grade_students[value].append(key)
print(grade_students)
```

```
{85: ['Alice', 'Charlie'], 92: ['Bob'], 100: ['Diana'], 78: ['Eve']}
```

Given a dictionary of items and their prices, calculate the average price of all items.

```
dic1={"dahi":20,"ghee":60,"mehndi":90,"doodh":80}
price=0
avg=0
for key,value in dic1.items():
  price+=dic1[key]
  avg=price/len(dic1)
print(avg)
```

```
62.5
```

You have a dictionary of employee names and their respective years of experience. Find the employee with the least experience.

```
employees = {
    "Alice": 8,
    "Bob": 9,
    "Charlie": 8,
    "Diana": 1,
    "Eve": 7
}

least_experience = None
least_experienced_employee = None

for employee, experience in employees.items():
    if least_experience is None or experience < least_experience:
        least_experience = experience
        least_experienced_employee = employee

print(f"Employee with least experience: {least_experienced_employee}
with {least_experience} years")
```

```
Employee with least experience: Diana with 1 years
```