Start coding or generate with AI.

Q1. Which keyword is used to create a function? Create a function to return a list of odd numbers in the range of 1 to 25.

Answer:

In Python, the keyword used to create a function is def.

```
# Here's a function that returns a list of odd numbers in the range of 1 to 25:
def odd_numbers():
    return [num for num in range(1, 26) if num % 2 != 0]

# Call the function and print the result
print(odd_numbers())
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25]
```

Start coding or generate with AI.

Q2. Why *args and* *kwargs is used in some functions? Create a function each for *args and* *kwargs to demonstrate their use.

Answer: In Python, *args and* *kwargs are used to pass a variable number of arguments to a function.

args allows to pass a variable number of non-keyword arguments to a function. *kwargs allows to pass a variable number of keyword arguments (i.e., named arguments) to a function.

```
def sum_of_numbers(*args):
    return sum(args)

# Example usage
print(sum_of_numbers(1, 2, 3))      # Output: 6
print(sum_of_numbers(4, 5, 6, 7))   # Output: 22
```

```
6
22
```

```
def print_user_details(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")

# Example usage
print_user_details(name="John", age=30, location="New York")
```

```
name: John
age: 30
location: New York
```

Q3. What is an iterator in python? Name the method used to initialise the iterator object and the method used for iteration. Use these methods to print the first five elements of the given list [2, 4, 6, 8, 10, 12, 14, 16, 18, 20].

Answer:

An iterator in Python is an object that contains a countable number of values and can be traversed through, one element at a time. Iterators implement two methods:

iter() to initialize the iterator.

next() to get the next value from the iterator.

An iterator can only go forward and is exhausted once all elements are accessed.

Methods:

iter(): This method is used to initialize an iterator object.

next(): This method is used to iterate through the next elements of the collection.

```
# Given list
my_list = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

# Get an iterator object from the list
iterator = iter(my_list)

# Use the next() method to print the first five elements
for _ in range(5):
    print(next(iterator))
```

```
2
4
6
8
10
```

Q4. What is a generator function in python? Why yield keyword is used? Give an example of a generator function.

Answer:

A generator function in Python is a function that returns an iterator, where the values are generated lazily (one at a time) instead of returning all values at once. This makes generator functions more memory-efficient, especially when dealing with large datasets, because they yield values one by one rather than storing them in memory.

The yield keyword is used in a generator function to:

1. Return a value from the function temporarily.
2. Pause the function execution, maintaining its state (local variables and the point where it was paused).
3. Resume execution when the next value is requested (using next()).

Unlike the return keyword, which ends the function's execution, yield allows the function to be paused and resumed, making it ideal for producing sequences of values.

```
# Example:
def even_numbers():
    for num in range(2, 12, 2):  # Generates numbers from 2 to 10
        yield num

# Using the generator function
even_gen = even_numbers()

# Printing values from the generator
for even in even_gen:
    print(even)
```

```
2
4
6
8
10
```

Q5. Create a generator function for prime numbers less than 1000. Use the next() method to print the first 20 prime numbers.

Answer:

Here’s a generator function to generate prime numbers less than 1000, and an example of how to use the next() method to print the first 20 prime numbers.

```
def prime_numbers():
    for num in range(2, 1000):  # Start checking from 2 to 999
        is_prime = True
        for i in range(2, int(num**0.5) + 1):
            if num % i == 0:
                is_prime = False
                break
        if is_prime:
            yield num  # Yield the prime number

# Using the generator function to print the first 20 prime numbers
prime_gen = prime_numbers()

# Printing the first 20 prime numbers
for _ in range(20):
    print(next(prime_gen))
```

```
2
3
5
7
```

```
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
```

Thank You!