3rd feb func pythone assignment

August 9, 2023

3rd feb assignment

Q1. Which keyword is used to create a function? Create a function to return a list of odd numbe

range of 1 to 25

[25]:
```
odd_num=[]
def odd1():
    for num in range(1,25):
        if num % 2 !=0:
            odd_num.append(num)
odd1 = result
print(result)
```

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

[ ]:
```

```

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

ANS) In Python, *args and **kwargs are used to pass a variable number of arguments to a function. They allow you to work with an arbitrary number of positional and keyword arguments, respectively.

Here's an example demonstrating the use of *args and **kwargs in functions:

[28]:
```
def print_args(*args):
    print("Positional arguments:")
    for arg in args:
        print(arg)

# Demonstrate the use of *args
print_args(1, 2, 3, "hello", [4, 5])
```

```
Positional arguments:
1
2
3
```

```
hello
[4, 5]
```

[ ]:

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].

To initialize an iterator object, you use the iter() method, and for iteration, you use the next() method.

[29]:
```
my_list = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
my_iterator = iter(my_list)

for _ in range(5):
    element = next(my_iterator)
    print(element)
```

```
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

ans) A generator function in Python is a special type of function that produces an iterator, allowing you to iterate over a sequence of values without the need to create the entire sequence in memory at once. Generator functions use the yield keyword to yield values one at a time, maintaining their internal state between calls.

The yield keyword is used in a generator function to temporarily pause the function's execution and yield a value to the caller. The function's state is saved, and when the generator is iterated again, it resumes execution from where it was paused, continuing to generate and yield values until the function exits or raises a StopIteration exception.

Here's an example of a generator function that generates a sequence of squares:

[30]:
```
def square_generator(n):
    for i in range(n):
        yield i ** 2

# Create a generator object
squares = square_generator(5)

# Iterate and print squares
```

```
for square in squares:
    print(square)
```

```
0
1
4
9
16
```

[ ]:

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

[31]:
```
def is_prime(num):
    if num <= 1:
        return False
    if num <= 3:
        return True
    if num % 2 == 0 or num % 3 == 0:
        return False
    i = 5
    while i * i <= num:
        if num % i == 0 or num % (i + 2) == 0:
            return False
        i += 6
    return True

def prime_generator(limit):
    num = 2
    count = 0
    while count < limit:
        if is_prime(num):
            yield num
            count += 1
        num += 1

# Create a generator object
prime_gen = prime_generator(20)

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

```
2
3
5
```

```
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
```

[ ]:

Q6. Write a python program to print the first 10 Fibonacci numbers using a while loop

def fibonacci(n): fib_series = [] a, b = 0, 1 count = 0

```
while count < n:
    fib_series.append(a)
    a, b = b, a + b
    count += 1

return fib_series
```

1 Print the first 10 Fibonacci numbers using the fibonacci function

fibonacci_numbers = fibonacci(10) for number in fibonacci_numbers: print(number)

[ ]:

Q7. Write a List Comprehension to iterate through the given string: 'pwskills'.

Expected output: ['p', 'w', 's', 'k', 'i', 'l', 'l', 's']

[34]:
```
input_string = 'pwskills'
output_list = [char for char in input_string if char in 'pwskills']
print(output_list)
```

```
['p', 'w', 's', 'k', 'i', 'l', 'l', 's']
```

[ ]:

Q8. Write a python program to check whether a given number is Palindrome or not using a while loop

[35]:
```
def is_palindrome(number):
    original_number = number
    reverse = 0

    while number > 0:
        digit = number % 10
        reverse = reverse * 10 + digit
        number //= 10

    return original_number == reverse

# Input a number from the user
num = int(input("Enter a number: "))

if is_palindrome(num):
    print(f"{num} is a palindrome.")
else:
    print(f"{num} is not a palindrome.")
```

```
Enter a number:  123

123 is not a palindrome.
```

[ ]:

Q9. Write a code to print odd numbers from 1 to 100 using list comprehension.

Note: Use a list comprehension to create a list from 1 to 100 and use another List comprehension to filter out odd numbers

[36]:
```
odd_numbers = [num for num in range(1, 101) if num % 2 != 0]
print(odd_numbers)
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41,
43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81,
83, 85, 87, 89, 91, 93, 95, 97, 99]
```

[ ]: