03 Feb Assignment

May 10, 2023

[ ]:
```
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.
```

[ ]:
```
ANS -
```

[4]:
```
def odd_numbers():
    return [i for i in range(1, 26) if i % 2 != 0]
```

[6]:
```
odd_numbers()
```

[6]:
```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25]
```

[ ]:

[ ]:

[ ]:
```
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 special syntax used in function definitions␣
 ↪that allow you to pass a variable number of arguments to a
function.
   *args allows you to pass a variable number of non-keyworded arguments to a␣
 ↪function as a tuple.

   **kwargs allows you to pass a variable number of keyword arguments to a␣
 ↪function as a dictionary.
```

[9]:
```
def my_fun(*args, **kwargs):
    for arg in args:
        print("hello world")
    for key, value in kwargs.items():
        print(f"{key} = {value}")
```

[ ]:

[ ]:

[ ]:
```
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].
```

[ ]:
```
ANS -
```

[ ]:
```
An iterator in Python is an object that is used to iterate over iterable␣
 ↪objects like lists, tuples, dicts, and sets. The Python iterators
object is initialized using the iter () method. It uses the next () method for␣
 ↪iteration.
```

[ ]:
```
To initialize an iterator object in Python, we use the iter() method. To␣
 ↪iterate over an iterable object like a list, we use the next()
method.
```

[11]:
```
lst = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
it = iter(lst)
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

[ ]:

[ ]:

[ ]:
```
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 function that uses the yield keyword␣
 ↪instead of return. When a generator function is called, it
returns an iterator object without executing the body of the function. The␣
 ↪iterator object can be used to iterate over the values produced by
```

```
the generator function one at a time on demand. This makes generators␣
 ↪memory-efficient and faster than normal functions when dealing with
large datasets.
```

[12]:
```
def square_numbers(n):
    for i in range(1, n+1):
        yield i * i

for square in square_numbers(5):
    print(square)
```

```
1
4
9
16
25
```

[ ]:

[ ]:

[ ]:
```
Q5. Create a generator function for prime numbers less than 1000. Use the␣
 ↪next() method to print the
first 20 prime numbers.
```

[ ]:
```
ANS -
```

[13]:
```
def get_primes(n):
    primes = []
    for num in range(2, n):
        if all(num % i != 0 for i in range(2, num)):
            primes.append(num)
            if len(primes) == 20:
                break
    return primes

primes = get_primes(1000)
for prime in primes:
    print(prime)
```

```
2
3
5
7
11
13
17
```

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

[ ]: