assignment2

April 11, 2023

0.1 QUESTION 1

0.2 ANS- def is the keyword that is used to create function

[12]:
```
## Create a function to return a list of odd numbers in the range of 1 to 25.
def odd_nums():
    odd=[]
    for i in range(1,26):
        if i%2!=0:
            odd.append(i)
    return odd
```

[13]:
```
odd_nums()
```

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

0.3 QUESTION 2

0.4 ANS- *args and **kvargs are the special syntax in python that allows the function to accept arbitory number of arguments

0.5 *args is used to pass a variable number of non key arguments to the function and it collects all of them in tuples

0.6 **kvargs is used to pass a variable number of key arguments to the function and collects them in form of dictionary

[14]:
```
## example of *args and **kvargs
def test(*args,**kvargs):
    return args,kvargs
```

[16]:
```
test(1,2,3,4,5,a=6,b=7,c=8,d=["aryan","naveen"])
```

[16]:
```
((1, 2, 3, 4, 5), {'a': 6, 'b': 7, 'c': 8, 'd': ['aryan', 'naveen']})
```

0.7 QUESTION 3

0.8 ANS- An iterator in Python is an object that allows you to iterate (i.e., loop) through a collection of data, such as a list, tuple, or dictionary. An iterator can be used to retrieve each element of the collection one at a time, which can be useful for processing large amounts of data without loading everything into memory at once.

0.9 you can create an iterator object by calling the iter() function on a collection of data. This creates an iterator that points to the first element of the collection. You can then iterate through the collection by calling the next() function on the iterator object. This moves the iterator to the next element of the collection and returns its value

[18]:
```
## iter() and next() example
list=[2, 4, 6, 8, 10, 12, 14, 16,18, 20]
my_iterator=iter(list)
for i in range(5):
    print(next(my_iterator))
```

```
2
4
6
8
10
```

0.10 QUESTION 4

0.11 ANS- A generator function is a special type of function in Python that allows you to generate a sequence of values on-the-fly. Instead of returning a value and exiting like a normal function, a generator function uses the yield keyword to produce a series of values that can be iterated over using a for loop or other iteration techniques.

0.12 The yield keyword is used in generator functions to pause the execution of the function and produce a value to be returned. When the yield statement is executed, the current state of the function is saved, along with the current value of the yield expression. The function then returns the value to the caller, but unlike a normal function, the function's state is saved, so that when the function is called again, it resumes execution from where it left off, with the saved state restored.

[21]:
```
## EXAMPLE
def even_num(n):
    for i in range(n):
        if i%2==0:
            yield(i)
```

[22]:
```
for i in even_num(10):
    print(i)
```

```
0
2
4
6
8
```

0.13 QUESTION 5

[23]:
```
def primes():
    n = 2
    while n < 1000:
        if all(n % i != 0 for i in range(2, int(n ** 0.5) + 1)):
            yield n
        n += 1
```

[24]:
```
prime_generator = primes()

for i in range(20):
    print(next(prime_generator))
```

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: