functions assignment1

August 2, 2023

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.

[1]:
```
# def keyword used for create funtion in python.
def odd():
    list1 = []
    for num in range(1,25):
        only_odd = [num for num in list1 if num % 2 == 1]
        list1.append(num)
    print(only_odd)
odd()
```

```
[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.

[1]:
```
# *args:-
'''
Python has *args which allow us to pass the variable number of non keyword␣
 ↪arguments to function.
In the function, we should use an asterisk * before the parameter name to pass␣
 ↪variable length arguments.
The arguments arepassed as a tuple and these passed arguments make tuple inside␣
 ↪the function with same name
as the parameter excluding asterisk *.
'''
#Ex. of *args:-

def adder(*num):
    sum = 0

    for n in num:
        sum = sum + n

    print("Sum:",sum)

adder(3,5)
adder(4,5,6,7)
```

```
adder(1,2,3,5,6)
# **kwargs:-

'''
Python passes variable length non keyword argument to function using *args but we cannot use this
to pass keyword argument. For this problem Python has got a solution called **kwargs, it allows us to pass
the variable length of keyword arguments to the function.
In the function, we use the double asterisk ** before the parameter name to denote this type of argument.
The arguments are passed as a dictionary and these arguments make a dictionary inside function with name
same as the parameter excluding double asterisk **.
'''
#Ex. of **kwargs:-

def intro(**data):
    print("\nData type of argument:",type(data))

    for key, value in data.items():
        print("{} is {}".format(key,value))

intro(Firstname="Sita", Lastname="Sharma", Age=22, Phone=1234567890)
intro(Firstname="John", Lastname="Wood", Email="johnwood@nomail.com", Country="Wakanda", Age=25, Phone=9876543210)
```

```
Sum: 8
Sum: 22
Sum: 17

Data type of argument: <class 'dict'>
Firstname is Sita
Lastname is Sharma
Age is 22
Phone is 1234567890

Data type of argument: <class 'dict'>
Firstname is John
Lastname is Wood
Email is johnwood@nomail.com
Country is Wakanda
Age is 25
Phone is 9876543210
```

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]

[2]:
```
# iterator in python:
'''
An iterator is an object that contains a countable number of values.
An iterator is an object that can be iterated upon, meaning that you can␣
 ↪traverse through all the values.
'''
#Name the method used to initialise the iterator object and the method used for␣
 ↪iteration:
'''
The iterator object is initialized using the iter() method. It uses the next()␣
 ↪method for iteration.
1]__iter__(): The iter() method is called for the initialization of an iterator.
 ↪ This returns an iterator object.

2]__next__(): The next method returns the next value for the iterable. When we␣
 ↪use a for loop to traverse any iterable
object, internally it uses the iter() method to get an iterator object, which␣
 ↪further uses the next() method to iterate over.
This method raises a StopIteration to signal the end of the iteration.
'''
def printValues():
        l = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
        for i in range(1,5):
                l.append(i)
        print(l[:5]) #To print first 5 elements.
        #print(l[-5:]) # To print last 5 elements.

printValues()
```

```
[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.

[3]:
```
# Generator:
'''
A generator is a function that returns an iterator that produces a sequence of␣
 ↪values when iterated over.
Generators are useful when we want to produce a large sequence of values,but we␣
 ↪don't want to store all of
them in memory at once.
'''
# Why yield keyword is used:
'''
The advantages of using yield keywords instead of return are that the values␣
 ↪returned by yield statement are
```

```
stored as local variables states, which allows control over memory overhead␣
 ↪allocation. Also, each time, the
execution does not start from the beginning, since the previous state is␣
 ↪retained.
'''
# Ex of Generator:
def test_fib(n):
    a,b=0,1
    for i in range(n):
         yield a
         a,b = b ,a+b
test_fib(10)
for i in test_fib(10):
    print(i)
```

```
0
1
1
2
3
5
8
13
21
34
```

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

[2]:
```
def is_prime(n):
    if n<2:
        return False
    for i in range(2,n):
        if n%i==0:
          return False
    return True

def prime_generator():
    num=2
    while num<1000:
       if is_prime(num):
          yield num
       num+=1
primeCheck=prime_generator()
for i in range(20):
    print(next(primeCheck))
```

```
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.

[3]:
```
n1=0
n2=1
i=0
while i<10:
    num=n1+n2
    n1=n2
    n2=num
    i+=1
    print(num)
```

```
1
2
3
5
8
13
21
34
55
89
```

Q7. Write a List Comprehension to iterate through the given string: 'pwskills'. Expected output: ['p', 'w', 's', 'k', 'i', 'l', 'l', 's']

[4]:
```
output=[i for i in 'pwskills']
print(output)
```

```
['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.
```

[5]:
```
def is_palindrome(num):
  new_num = 0
  original_num = num

  while num > 0:
    new_num = new_num * 10 + num % 10
    num = num // 10

  if new_num == original_num:
        print(f"Given {original_num} is a palindrome number")
  else:
        print(f"Given {original_num} is not a palindrome number")

is_palindrome(1212)
is_palindrome(12321)
```

```
Given 1212 is not a palindrome number
Given 12321 is a palindrome number
```

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

[6]:
```
odd_num=[x for x in range(100) if x%2!=0]
print(odd_num)
```

```
[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]
```

[ ]:
```

```