Solutions,14feb,2022

1.Define a function that can accept two strings as input and print the string with maximum length in console. If two strings have the same length, then the function should print all strings line by line.

Solution:

```
>>>def printVal(s1,s2):
>>> len1 = len(s1)
>>>     len2 = len(s2)
>>>     if len1 > len2:
>>>         print(s1)
>>>     elif len1 < len2:
>>>         print(s2)
>>>     else:
>>>         print(s1)
>>>         print(s2)
>>>
>>> s1,s2=input().split()
>>> printVal(s1,s2)
```

2.Define a function which can generate a list where the values are square of numbers between 1 and 20 (both included). Then the function needs to print the last 5 elements in the list.

Solution:

```
>>> def printList():
>>>     lst = [i ** 2 for i in range(1, 21)]
>>>     for i in range(19,14,-1):
>>>         print(lst[i])
>>>
>>> printList()
```

3.With a given tuple (1,2,3,4,5,6,7,8,9,10), write a program to print the first half values in one line and the last half values in one line.

Solution:

```
>>> tpl = (1,2,3,4,5,6,7,8,9,10)
>>>
>>> for i in range(0,5):
>>>     print(tpl[i],end = ' ')
>>> print()
>>> for i in range(5,10):
>>>     print(tpl[i],end = ' ')
```

4.Define a class named Shape and its subclass Square. The Square class has an init function which takes a length as argument. Both classes have a area function which can print the area of the shape where Shape's area is 0 by default.

Solution:

```
>>> class Shape():
>>>     def __init__(self):
>>>         pass
>>>
>>>     def area(self):
>>>         return 0
>>>
>>> class Square(Shape):
>>>     def __init__(self,length = 0):
>>>         Shape.__init__(self)
>>>         self.length = length
>>>
>>>     def area(self):
>>>         return self.length*self.length
>>>
>>> Asqr = Square(5)
>>> print(Asqr.area())      # prints 25 as given argument
>>>
>>> print(Square().area())  # prints zero as default area
```

5.Write a program to compute:

f(n)=f(n-1)+100 when n>0
and f(0)=0
with a given n input by console (n>0).

Solution:

```
>>> def f(n):
>>>     if n == 0:
>>>         return 0
>>>     return f(n-1) + 100
>>>
>>> n = int(input())
>>> print(f(n))
```

6.Please write a binary search function which searches an item in a sorted list. The function should return the index of element to be searched in the list.

Solution:

```
>>> def binary_search_Ascending(array, >>> target):
>>>     lower = 0
>>>    upper = len(array)
>>>     print('Array Length:',upper)
>>>     while lower < upper:
>>>         x = (lower + upper) // 2
>>>        print('Middle Value:',x)
>>>         value = array[x]
>>>         if target == value:
>>>             return x
>>>         elif target > value:
>>>             lower = x
>>>         elif target < value:
>>>             upper = x
>>>
>>> Array=[1,5,8,10,12,13,55,66,73,78,82,85,88,99]
>>> print('The Value Found at Index:',binary_search_Ascending(Array, 82))
```

7.Please write a program to generate all sentences where subject is in ["I", "You"] and verb is in ["Play", "Love"] and the object is in ["Hockey","Football"].

Solution:

```
>>> subjects=["I", "You"]
>>> verbs=["Play", "Love"]
>>> objects=["Hockey","Football"]
>>>
>>> for sub in subjects:
>>>     for verb in verbs:
>>>         for obj in objects:
>>>            print("{} {} {}".format(sub,verb,obj))
```

8.Define a class Person and its two child classes: Male and Female. All classes have a method "getGender" which can print "Male" for Male class and "Female" for Female class.

Solution :

```
>>> class Person(object):
>>>     def __init__(self):
>>>      self.gender = "unknown"
>>>
>>>    def getGender(self):
>>>      print(self.gender)
>>>
>>> class Male(Person):
>>>     def __init__(self):
>>>      self.gender = "Male"
>>>
>>> class Female(Person):
>>>     def __init__(self):
>>>      self.gender = "Female"
>>>
>>> sharan = Female()
>>>>dong = Male()
>>> sharan.getGender()
>>> dong.getGender()
```

9.Please write a program which prints all permutations of [1,2,3]

Solution:

```
>>> from itertools import permutations
>>>
>>> def permuation_generator(iterable):
>>>     p = permutations(iterable)
>>>     for i in p:
>>>         print(i)
>>>
>>>
>>> x = [1,2,3]
>>> permuation_generator(x)
```

10.Write a program to solve a classic ancient Chinese puzzle: We count 35 heads and 94 legs among the chickens and rabbits in a farm. How many rabbits and how many chickens do we have?

Solution:

```
>>> import itertools
>>>
>>> def animal_counter(lst):
>>>     chickens = 0
>>>     rabbits = 0
>>>     for i in lst:
>>>         if i == 2:
>>>             chickens += 1
>>>         elif i == 4:
>>>             rabbits += 1
>>>     print("Number of chickens is {chickens}\nNumber of rabbits is {rabbits}")
>>>
>>>
>>> def animal_calculator(total_legs, total_heads, legs_of_each_species):
>>>     combinations = itertools.combinations_with_replacement(legs_of_each_species,
total_heads)
>>>     correct_combos = []
>>>     for i in list(combinations):
>>>         if sum(i) == total_legs:
>>>             correct_combos.append(i)
>>>     print(correct_combos)
>>>     for i in correct_combos:
```

```
>>>         animal_counter(i)
>>>
>>> animal_calculator(94, 35,legs_of_each_species=[2,4])
```