Problem solving from scratch [SOLUTIONS]

NB: there are likely to be many ways of solving these problems. Below are some suggested solutions, but may not be the only - or even best - ways to get the right functionality.

---

1. Variables, basic control flow, simple types, textual I/O, extending to related functions

1. State Change

```
temp = int( input( "Enter the current temperature: " ) )

state = "liquid"
if temp <= 0:
    state = "solid"
elif temp >= 100:
    state = "gas"

print( "At " + str( temp ) + " degrees centigrade, " +
     "water will be a " + state + "." )
```

2. Average Sleep Calculator

```
total = 0
days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
         "Saturday", "Sunday"]
for i in range(0, 7):
    hours = input("Hours of sleep on night "+days[i]+": ")
    total += int(hours)
print(total/7)
```

3. Age Checker (Integer in Range Checker)

```
def getInRange(type, min, max):
    user_in = int(input("Input "+type+": "))
    while user_in < min or user_in > max:
        user_in = int(input("Invalid entry. Input "+type+": "))
    return user_in
```

4. Concussion check calculator

```
def getInRange(type, min, max):
    user_in = int(input("Input "+type+": "))
    while user_in < min or user_in > max:
        user_in = int(input("Invalid entry. Input "+type+": "))
    return user_in

print("Rate symptoms from 0 not at all to 10 being the most severe")

vom = getInRange("how stringly they feel they are going to be sick", 0,
10)
bal = getInRange("how bad their sense of balance is", 0, 10)
drows = getInRange("how drowsy they feel", 0, 10)
mem = getInRange("how bad their memory of recent events is", 0, 10)

v_weight = 3
b_weight = 2
d_weight = 1
m_weight = 4
concussed = v_weight * vom + b_weight * bal + d_weight * drows + m_weight
* mem

print()
print("They have a " + str(concussed) + "% chance of being concussed")
```

5. Ordinal Numbers

```
endings = [ "th", "st", "nd", "rd" ]

for i in range( 1,32 ):
    if (i > 10 and i < 14) or i % 10 > 3:
        ind = 0
    else:
        ind = i % 10

    print( str( i ) + endings[ ind ] )
```

Now, write a function that takes a number between 1 and 31 and returns the abbreviated ordinal number for that number, e.g. 1 gives 1st, 2 gives 2nd, 3 gives 3rd, and so on.

```
endings = [ "th", "st", "nd", "rd" ]

def getAbbreviatedOrdinal( i ):
    if (i > 10 and i < 14) or i % 10 > 3:
        ind = 0
    else:
        ind = i % 10
        return str( i ) + endings[ ind ]
```

6. Find Max without max()

```
    def max(a,b,c):
        if (a > b) and (a > c):
            return a
        elif (b > a) and (b > c):
            return b
        elif (c > a) and (c > b)
            return c

    #or...
    def max(a,b,c):
        if a > b:
            if a > c:
                return a
            else:
                return c
        elif b > c:
            return b
        else:
            return c
```

7. Guessing Game

```
    import random

    number = random.randint(1,100)
    guess = 0

    while guess != number:
         guess = input( "Please enter your guess." )
         guess = int(guess)

         if guess < number:
             print("Too low!")
         elif guess > number:
             print("Too high!")
         else:
             print("You got it!")
```

8. Sum of List of Integers

```
    def addem (numbers):
         total = 0
```

```
        for n in numbers:
            total += n
        return total
```

2. Adding in strings

9. Anagram Maker/Scrambler

```
import random
word = input("Enter word for anagram:")

anagram = ""
for i in range( len( word ) ):
    newLetterPos = random.randint( 0, len( word ) - 1 )
    anagram = anagram + word[ newLetterPos ]
    word = word[ :newLetterPos ] + word[ newLetterPos + 1: ]

print(anagram)
```

10. String Reversal

```
inputString = input("Please enter string:")

newString = ''
index = len(inputString)

while index > 0:
    index -= 1
    newString += inputString[index]

print(newString)
```

11. Tabletop Dice Roller

```
from random import randint
input_string = input("Enter your dice: ")

while input_string != "":
    inputs = input_string.split(" ")
    dice = inputs[0]
    modifier = inputs[1]
    dice_split = dice.split("d")
    dice_count = int(dice_split[0])
    dice_sides = int(dice_split[1])
    total = 0
    for i in range(0, dice_count):
        roll = randint(1, dice_sides)
        total += roll

    if modifier[0] == "+":
        total += int(modifier[1:])
    else:
        total -= int(modifier[1:])

    print(total)
    input_string = input("Enter your dice (leave blank to close): ")
```

3. Adding in lists (array lists)

12. Increment every Integer in a List

```
theList = [ 3, 6, 4, 8, 19, 2, 34, 17 ]

for i in range(len(theList)):
    theList[ i ] = theList[ i ] + 1
```

13. Playlist Shuffle

```
from random import shuffle
playlist = ["Ashes by Celine Dion",
            "Welcome to the Party by Diplo",
            "Nobody Speak by DJ Shadow",
            "In Your Eyes by Peter Gabriel",
            "Take on Me by a-ha",
            "If I Could Turn Back Time by Cher",
            "9 to 5 by Dolly Parton",
            "All Out Of Love by Air Supply",
            "Bangarang by Skrillex"]

shuffled = shuffle(playlist)
for line in playlist:
    print(line)
```

4. Adding in dictionaries as a look-up table

14. Word Counter

```
wordCounts = {}

line = input()
```

```
while line != ".":
    words = line.split( " " )

    for word in words:
        if word in wordCounts:
            wordCounts[ word ] = wordCounts[ word ] + 1
        else:
            wordCounts[ word ] = 1

    line = input()

for word in wordCounts:
    print( word + ": " + str( wordCounts[ word ] ) )
```

15. Square Root Lookup

```
import math
sqrts = {}

for i in range( 1, 100 ):
    sqrts[ i ] = math.sqrt( i )

while True:
    square = int( input( "Type in the number: " ) )

    if square > 0 and square <= 100:
        print( "The square root is " + str( sqrts[ square ] ) )
    else:
        print( "The number you have entered is out of range." )
```

16. Stone, Paper, Scissors Game

```
import random
weapon = raw_input("Please enter stone, paper or scissors:")

winningWeapon = {
    "paper": "stone",
    "stone": "scissors",
    "scissors": "paper"
}

computerWeapon = random.choice(winningWeapon.keys())
print("Computer picks: " + computerWeapon)
if computerWeapon == weapon:
    print("Draw")
elif winningWeapon[computerWeapon] == weapon:
    print("Computer wins")
elif winningWeapon[weapon] == computerWeapon:
    print("You win")
else:
    print("Error")
```

5. Adding in dictionaries used as a “record” data type

17. Older than Average

```
people = []
totalAge = 0

name = input( "Name (full-stop to finish): " )
while name != ".":
    age = int( input( "Age: " ) )
    totalAge = totalAge + age
    people = people + [ { "name" : name, "age" : age } ]

    name = input( "Name (full-stop to finish): " )

averageAge = totalAge / len( people )

for person in people:
    if person[ "age" ] > averageAge:
        print ( person[ "name" ] )
```

6. Adding in file I/O

18. People over 21

```
f = open( "namesAges.txt" )

lines = f.readlines()

# Assumes the name comes first, then the age
for line in lines:
    pieces = line.split( " " )
    if int( pieces[ 1 ] ) >= 21:
        print( pieces[ 0 ] )
```

19. Longest Name/Reversed Names

```
f = open( "names.txt" )
lines = f.readlines()
```

```
lenLongestName = len( lines[ 0 ] )

for name in lines[ 1: ]:
    if len( name ) > lenLongestName:
        lenLongestName = len( name )

for name in lines:
    if len( name ) == lenLongestName:
        print( name )

for nameIndex in range( len( lines ) - 1, -1, -1 ):
    print ( lines[ nameIndex ] )
```

20. Age Lookup

```
f = open( "namesAges.txt" )

lines = f.readlines()

people = []

# Assumes the name comes first, then the age
for line in lines:
    pieces = line[:-1].split( " " )
    people = people + [ { "name" : pieces[ 0 ], "age" : int(
pieces[ 1 ] ) } ]

age = int( input() )
while age != 0:
    noNames = True

    for person in people:
        if person[ "age" ] == age:
            print ( person[ "name" ] )
            noNames = False

    if noNames:
        print ( "No names of that age." )

    age = int( input() )
```

21. Lottery Prize Calculator (Number of 3-Number Winners)

```
file = open('lottery_tickets.txt','r')
```

```
winningNumbers = input("Please enter the winning numbers
1,2,3,4,5,6:")
winningList = winningNumbers.split(",")
winningIDs = ""

for line in file.readlines():
     id, numbers = line.split(" ")
     numbersList = numbers.split(",")
     count = 0
     for num in numbersList:
          if num in winningList:
               count += 1
     if count == 3:
          winningIDs += id + " "
print("The number winning three numbers is:" + str(winningIDs))
```

22. Pretty Print Tables

```
def whitespace(string, length):
    while len(string) < length:
        string += " "
    return string

f = open( "games.txt" )
raw_rows = f.readlines()
rows = []
for row in raw_rows:
    row = row[:-1].split(",")
    rows+=[row]
num_columns = len(rows[0])
column_widths = [0]*num_columns

for i in range(num_columns):
    max_width = 0
    for row in rows:
        if len(row[i]) > max_width:
            max_width = len(row[i])
    column_widths[i] = max_width

h_rule= "+"
for i in range(num_columns):
    h_rule+= "-"*column_widths[i] + "--+"
print(h_rule)
out_row = "| "
```

```
    for i in range(num_columns):
        out_row += whitespace(rows[0][i], column_widths[i]) + " | "
    print(out_row)
    print(h_rule)
    for row in rows[1:]:
        out_row = "| "
        for i in range(num_columns):
            out_row += whitespace(row[i], column_widths[i]) + " | "
        print(out_row)
    print(h_rule)
```

23. Word Blanker

```
    f = open("input.txt")
    text = f.readline()

    output_string = ""

    words = text.split(" ")
    for word in words:
        if len(word) <= 2:
            output_string += word
        else:
            output_string += word[0] + "*"*(len(word)-2) + word[-1]
        output_string += " "
    print(output_string)
```

24. Mail Merge

```
    # Read in the data file to a list of dictionaries (records)
    # - turn first line into a list of field names
    # - then use this list to create a new dictionary for each
    # - remaining line

    f = open( "mergeData.txt" )

    fieldLine = f.readline()
    fields = fieldLine[ :-1 ].split( "," )

    data = []
    linesRemaining = f.readlines()

    for line in linesRemaining:
        items = line[ :-1 ].split( "," )
        newRecord = {}
        for i in range( len( fields ) ):
```

```
        newRecord[ fields[ i ] ] = items[ i ]
    data = data + [ newRecord ]

f.close()

# Read in the template file as a single string and then loop over
# the merge data, creating a new string each time, replaced fields
# with real data

f = open( "mergeTemplate.txt" )

template = f.read()

for thisData in data:
    newMsg = ""
    i = 0
    while i < len( template ):
        if template[ i ] != "<":
            newMsg = newMsg + template[ i ]
        else:
            i += 1
            fieldName = ""
            while template[ i ] != ">":
                fieldName += template[ i ]
                i += 1
            newMsg = newMsg + thisData[ fieldName ]
        i += 1
    print( newMsg )
```

25. Bus Timetable

```
# Read in the bus timetable file
# Keep a dictionary of the bus stop names, mapping to route position, and
# a list of lists for the timings of each service

f = open( "busTimetable.txt" )

lines = f.readlines()

busStopNames = lines[ 0 ][ :-1].split( "," )

busStopPositionLookup = {}
position = 0
for name in busStopNames:
    busStopPositionLookup[ name ] = position
    position += 1

services = []
```

```
for serviceLine in lines[ 1: ]:
    nextService = serviceLine[ :-1 ].split( "," )
    services += [ nextService ]

# Respond to user requests

while True:
    time = input( "Type in the time: " )
    stop = input( "Type in the bus-stop name: " )
    if stop in busStopPositionLookup:
        stopPosition = busStopPositionLookup[ stop ]

        nextService = 0
        nextTimeAtStop = services[ nextService ][ stopPosition ]

        while time > nextTimeAtStop and nextService < len( services ) -
1:
            nextService += 1
            nextTimeAtStop = services[ nextService ][ stopPosition ]

        if time <= nextTimeAtStop:
            print( "Your next service will arrive at " + nextTimeAtStop )
        else:
            print( "There are no more services at your stop today" )

    else:
        print( "Bus stop not found in route, please try again." )
```