Module 3: Deep Dive - Functions, Sorting, Errors and Exception Handling, Regular Expressions and Packages

Assignment Solution

1. Build an interactive application which should simulate a Quiz contest. The following questions might be asked as input from user:

Choose level (easy, intermediate, and hard): → 3 modes of difficulty and user should input one of these choices.

Please give us the number of question you want to attempt: → No of questions thrown should be the number entered through this prompt.

Specify the question type (multiplication:M, addition:A, subtraction:S, division:D): → One of these operations to be performed.

If the answer is right or wrong, appropriate messages should be printed and move to next question if attempt count is not exceeded.

Hint: Random utility can be used to change complexity of questions.

The program should ask if the user wants to continue even after attempting the number of questions specified and accordingly should loop or terminate.

Sample:

Choose level (easy, intermediate, and hard): easy

Please give us the number of question you want to attempt: 3

Specify the question type (multiplication:M, addition:A, subtraction:S, division:D):D

What's 6 divided by 3?

2

That's right -- well done

Solution

```
from random import randint


solved = 0

total_num_q = 0


def play(num1, num2, type, solved):

    """ The main play function"""
```

```
def sp_type():
    type = raw_input("Specify the question type(multiplication:M, addition:A,
subtraction:S, division:D):")
    if type not in ['M','A','S','D']:
        print "Please input only char M,S,A and D"
    return type
type = ""
while type not in ['M','A','S','D']:
    type = sp_type()
if type == "M":
    ans = input("What's %d times %d? " % (num1, num2))
    result = num1 * num2
if type == "A":
    ans = input("What's %d plus %d? " % (num1, num2))
    result = num1 + num2
if type == "S":
    ans = input("What's %d minus %d? " % (num1, num2))
    result = num1 - num2
if type == "D":
    ans = input("What's %d divided by %d? " % (num1, num2))
    result = num1/num2

if ans == result:
    print "That's right -- well done.\n"
    solved = solved + 1
else:
    print "No, I'm afraid the answer is %d.\n" % result
return solved
```

```
def start_puzzle(solved, total_num_q):
  leval = ''
  play_leval = ['easy', 'intermediate', 'hard']
  while leval not in play_leval:
    leval = raw_input("Which leval you wanted to be in(easy, intermediate, hard): ")
    if leval not in play_leval:
      print "Please enter correct leval name.."
  num_q = input("Please give us the number of question you want to attempt: ")
  try:
    if int(num_q):
      pass
  except:
    print "Please enter only integet value"
  total_num_q = total_num_q + num_q
  for number_q in range(num_q):
    if leval == 'easy':
      num1 = randint(1, 10)
      num2 = randint(1, 10)
      solved = play(num1, num2, type, solved)
    elif leval == 'intermediate':
      num1 = randint(1, 20)
      num2 = randint(1, 20)
      solved = play(num1, num2, type, solved)
    elif leval == 'hard':
      num1 = randint(1, 30)
      num2 = randint(1, 30)
      solved = play(num1, num2, type, solved)
  return (solved, total_num_q)
```

```
inp = "yes"
while inp != 'no':
  solved,total_num_q = start_puzzle(solved,total_num_q)
  inp = raw_input("want to start puzzle(yes/no):")



print "\nOut of %s questions asked.  You got %d of them right." % (total_num_q,solved)
print "Well done!"
```

2. Write a recursive function to compute x raised to the power of n.

Solution

```
def fast_exp(x, n):
  if n == 0:
    return 1
  elif n % 2 == 0:
    return fast_exp(x*x, n/2))
  else:
    return x * fast_exp(x, n-1)
```

Alt solution :-

```
def exp(x, n):
  """
  Computes the result of x raised to the power of n.
    >>> exp(2, 3)
    8
    >>> exp(3, 2)
    9
  """
  if n == 0:
    return 1
  else:
    return x * exp(x, n-1)
```

3. Sort the list using lambda function mylist = [["john", 1, "a"], ["larry", 0, "b"]]. Sort the list by second item 1 and 0.

Solution

```
>>> mylist = [["john", 1, "a"], ["larry", 0, "b"]]
>>> mylist.sort(key=lambda x: x[1])
>>> print(mylist)
Output:
[['larry', 0, 'b'], ['john', 1, 'a']]
```

4. Sort the list using operator.itemgetter function mylist = [["john", 1, "a"], ["larry", 0, "b"]]. Sort the list by second item 1 and 0.

Solution

```
>>> from operator import itemgetter
>>> mylist = [["john", 1, "a"], ["larry", 0, "b"]]
>>> mylist.sort(key=itemgetter(1))
>>> print(mylist)
OutPut:
[['larry', 0, 'b'], ['john', 1, 'a']]
```