Warmup-1

March 22, 2021

Q1) Monkey_trouble

We have two monkeys, a and b, and the parameters a_smile and b_smile indicate if each is smiling. We are in trouble if they are both smiling or if neither of them is smiling. Return True if we are in trouble.

monkey_trouble(True, True) True monkey_trouble(False, False) True monkey_trouble(True, False) False

[1]:
```
def monkey_trouble(a_smile, b_smile):
  if (a_smile and b_smile) or (not a_smile and not b_smile):
    return True
  else:
    return False
```

Q2) sum_double

Given two int values, return their sum. Unless the two values are the same, then return double their sum.

sum_double(1, 2) 3 sum_double(3, 2) 5 sum_double(2, 2) 8

[2]:
```
def sum_double(a, b):
  if a == b :
    return ( a + b ) * 2
  else:
    return a + b
```

Q3) diff21 Given an int n, return the absolute difference between n and 21, except return double the absolute difference if n is over 21.

diff21(19) 2 diff21(10) 11 diff21(21) 0

hint:

The absolute difference of two real numbers x, y is given by |x y|, https://en.wikipedia.org/wiki/Absolute_difference

[5]:
```
def diff21(n):
  diff = abs( n - 21 )
  if n > 21:
    return diff * 2
  else:
    return diff
```

Q4) parrot_trouble

We have a loud talking parrot. The “hour” parameter is the current hour time in the range 0..23. We are in trouble if the parrot is talking and the hour is before 7 or after 20. Return True if we are in trouble.

parrot_trouble(True, 6) True parrot_trouble(True, 7) False parrot_trouble(False, 6) False

[6]:
```
def parrot_trouble(talking , hour):
  if talking and ( hour < 7 or hour > 20):
    return True
  else:
    return False
```

Q5) makes10 Given 2 ints, a and b, return True if one if them is 10 or if their sum is 10. makes10(9, 10) True makes10(9, 9) False makes10(1, 9) True

[7]:
```
def makes10(a, b):
  return (( a == 10 or b == 10 ) or ( a + b == 10 ))
```

Q6) near_hundred

Given an int n, return True if it is within 10 of 100 or 200. Note: abs(num) computes the absolute value of a number.

near_hundred(93) True near_hundred(90) True near_hundred(89) False

[8]:
```
def near_hundred(n):
  return ( abs(n - 100) <= 10 or abs(n-200) <=10 )
```

Q7) pos_neg

Given 2 int values, return True if one is negative and one is positive. Except if the parameter “negative” is True, then return True only if both are negative.

pos_neg(1, -1, False) True pos_neg(-1, 1, False) True pos_neg(-4, -5, True) True

[9]:
```
def pos_neg(a, b, negative):
  if negative:
    return ( a < 0 and b < 0)
  else:
    return ( a < 0 and b > 0 or a > 0 and b < 0 )
```

Q8) not_string

Given a string, return a new string where “not” has been added to the front. However, if the string already begins with “not”, return the string unchanged.

not_string(‘candy’) ‘not candy’ not_string(‘x’) ‘not x’ not_string(‘not bad’) ‘not bad’

hint : startwith method https://www.w3schools.com/python/ref_string_startswith.asp

[11]:
```
def not_string(str):
  if str.startswith("not"):
    return str
  else:
    return 'not ' + str
```

Q9) missing_char

Given a non-empty string and an int n, return a new string where the char at index n has been removed. The value of n will be a valid index of a char in the original string (i.e. n will be in the range 0..len(str)-1 inclusive).

missing_char('kitten', 1) 'ktten' missing_char('kitten', 0) 'itten' missing_char('kitten', 4) 'kittn'

[12]:
```
def missing_char(str, n):
  for i in range(0 , len(str)):
    return str.replace(str[n] , '')
```

Q10) front_back

Given a string, return a new string where the first and last chars have been exchanged.

front_back('code') 'eodc' front_back('a') 'a' front_back('ab') 'ba'

[46]:
```
def front_back(str):
  if len(str) <= 1 :
    return str
  else:
    empty_list = []
    for char in str :
      empty_list.append(char)
    empty_list[-1] , empty_list[0] = empty_list[0] , empty_list[-1]
    return ''.join(empty_list)
```

[48]:
```
#Solution:
def front_back(str):
  if len(str) <= 1:
    return str

  mid = str[1:len(str)-1]  # can be written as str[1:-1]

  # last + mid + first
  return str[len(str)-1] + mid + str[0]
```

Q11) front3 Given a string, we'll say that the front is the first 3 chars of the string. If the string length is less than 3, the front is whatever is there. Return a new string which is 3 copies of the front.

front3('Java') 'JavJavJav' front3('Chocolate') 'ChoChoCho' front3('abc') 'abcabcabc'

[49]:
```
def front3(str):
  front = str[0:3]
  if len(str) < 3 :
    return str * 3
  else:
    return front * 3
```

0.1 Warmup-1 (DONE)

[ ]:
```
```

[ ]:
```
```

[ ]:
```
```