👆

Problem Set 1

---

a. is_profitable

Prompt

Write a function called `is_profitable` that takes in two integers as parameters: revenue and expenses. Return `True` if your profit, defined as the difference between revenue and expenses, is greater than 0. Otherwise, return `False`.

Starter code

```
def is_profitable(revenue, expenses):
```

Example Input + Output

input	**output**	notes
is_profitable(2400, 1300)	True	
is_profitable(1350, 1350)	False	0 case, breaks even.
is_profitable(905, 1500)	False	

Example Solution

▼ Click here for the solution

```
def is_profitable(revenue, expenses):
  return revenue > expenses
```

Notes

▼ Click here for more information

- What's being assessed: booleans
- What's challenging: translating story elements into the solution
- Source/Inspired by: idk
- Written by: Paul Pham

Task Complexity

Branch	YES
Loop	NO
Nested Branch	NO

b. count_reverse

Prompt

Create a function called `count_reverse` that takes an integer n and returns a string comprised of every number between n and 0 in descending order. For example, `count_reverse(3)` should return the string ‘321’.

Positive and negative numbers should yield the same output. For example, `count_reverse(-3)` should return the string ‘321’.

Starter code

```
def count_reverse(n):
```

Example Input + Output

input	output	notes
count_reverse(7)	‘7654321’	
count_reverse(-3)	‘321’	

Example Solution

▼ Click here for the solution

```
def count_reverse(n):
  n = abs(n)

  result = ''
  while n > 0:
    digit = n % 10
    result += str(digit)
    digit //= 10

  return result
```

Notes

▼ Click here for more information

- What's being assessed: for loops with start, stop, and step (negative)
- What's challenging: creating the range of i, casting
- Source/Inspired by: previous problem in CSE 110
- Written by: Jared Lim + Paul Pham
- You could also accomplish this with a while loop

Task Complexity

Branch	NO
Loop	YES
Nested Branch	NO

c. pet_years

Prompt

It turns out that dogs and cats age differently from humans and from each other! There's a rule that many dog and cat owners use to convert the age of their pet to their equivalent age in dog/cat years. Write a function called `pet_years` that takes in two parameters: `animal` and `age`.

The `animal` parameter is a string that represents an animal. You may assume this string will always be lowercase. The `age` parameter is an integer that represents the age of that animal in human years.

This function should implement the following rules:

1. `animal` can only be "dog" or "cat" as a string. If it isn't, return -1.
2. If the `age` is less than 0, return -1.
3. For dogs, the general rules are:
   a. 1 human year = 15 pet years
   b. 2 human years = 24 pet years
   c. 3 or more human years = 24 + (5 pet years PER human year AFTER the second year)
4. For cats, the general rules are:
   a. 1 human year = 15 pet years
   b. 2 human years = 24 pet years
   c. 3 or more human years = 24 + (4 pet years PER human year AFTER the second year)

For example, the call `pet_years('cat', 4)` would return `32` since 24 + (4 * (4 - 2)) = 32.

Starter code

```
def pet_years(animal, age):
```

Example Input + Output

input	output	notes
pet_years(dog, 1)	15	fixed output
pet_years(dog, 3)	29	The rule over 2 years old applies here
pet_years(cat, 2)	24	fixed output
pet_years(cat, 4)	32	The rule over 2 years old applies here
pet_years(dog, -1)	-1	special case, negative
pet_years(turtle, 2)	-1	special case, not dog or cat

Example Solution

▼ Click here for the solution

```
def pet_years(animal, age):
  # special cases
  if age < 0:
    return -1
  elif animal != 'dog' and animal != 'cat':
    return -1

  pet_age = 0
  if age == 1:
    pet_age += 15
  elif age == 2:
    pet_age += 24
  else:
    if animal == 'dog':
      pet_age += 24 + (5 * (age - 2))
    else: # animal == 'cat'
      pet_age += 24 + (4 * (age - 2))

  return pet_age
```

Notes

▼ Click here for more information

- What's being assessed: efficiently structuring conditions (if, elif, else), operators (<, ==, >), data types
- What's challenging: dealing with multiple data types, casting
- Source/Inspired by: Seattle Central College, CSE 110 problem sets
- Written by: Jared

Task Complexity

Branch	YES
Loop	NO
Nested Branch	YES

---

d. mail_on_time

Prompt

The person who delivers mail is very punctual, but sometimes the odds can be stacked against them. They can deliver all of their mail on time when their speed is at least 70 km (i.e. 70 or greater). If it’s snowing, they must lower their speed by 10 in order to get around the world safely.

Write a function called `mail_on_time` that takes two parameters: the mailperson’s initial speed (integer) and `is_snowing`, an integer that represents whether or not it is snowing (`1` if it is snowing and `0` if it isn’t. This function should return `1` if the mail will arrive on time and `0` otherwise.

Starter code

```
def mail_on_time(speed, is_snowing)
```

Example Input + Output

input	output	notes

mail_on_time(80, 1)	1	
mail_on_time(79, 1)	0	
mail_on_time(75, 1	0	
mail_on_time(65, 0)	0	

Example Solution

▼ Click here for the solution

```
def mail_on_time(speed, is_snowing):
    if is_snowing == 1:
        return speed - 10 >= 70
    else:
        return speed >= 70
```

Notes

▼ Click here for more information

- What's being assessed: boolean operators, conditions
- What's challenging: accounting for exclusive bound, separating conditions based on the `is_snowing` parameter
- Source/Inspired by: CodingBat `cigar_party`
- Written by: Paul Pham

Task Complexity

Branch	YES
Loop	NO
Nested Branch	NO

e. reverse_word

Prompt

Create a function called `reverse_word` that takes a string as a parameter and returns the string written backwards.

Starter code

```
def reverse_word(word):
```

Example Input + Output

input	output	notes
reverse_word('hello')	olleh	
reverse_word('HiHiHi')	iHiHiH	
reverse_word('a')	a	

Example Solution

▼ Click here for the solution

```
# common solution
def reverse_word(word):
  reverse = ''
  for i in range(len(word) - 1, -1, -1):
    reverse += word[i]
  return reverse

# 'pythonic' solution
def reverse_word(word):
  return word[::-1]
```

Notes

▼ Click here for more information

- What's being assessed: String concatenation, for loops with start, stop, step

- What's challenging: concatenating string
- Source/Inspired by: previous problems in CSE 110
- Written by: Jared Lim
- You could also accomplish with just one line: print(word[::-1]), although it may be less intuitive than looping over the word.

Task Complexity

Branch	NO
Loop	YES
Nested Branch	NO

---

f. digit_sum_match

Prompt

Write a function called `digit_sum_match` that takes two arguments, an integer n and an integer target, that returns True if the sum of the digits in n is the same as target, otherwise returning False.

You are NOT allowed to cast the number into a string to solve this problem. You can assume that the integer `n` will always be positive.

Starter code

```
def digit_sum_match(n, target):
```

Example Input + Output

input	**output**	**notes**
digit_sum_match(123, 6)	True	
digit_sum_match(123, 7)	False	

Example Solution

▼ Click here for the solution

```
def digit_sum_match(n, target):
  total = 0
  while n > 0:
    digit = n % 10
    total += digit
    n = n // 10
  return total == target
```

Notes

▼ Click here for more information

- What's being assessed: integer manipulation, while loops
- What's challenging: pulling off the last digit of an integer, using the while loop bound to handle that case
- Source/Inspired by: From CSE 163 Week 2 section handout
- Written by: Paul Pham

Task Complexity

Branch	NO
Loop	YES
Nested Branch	NO

---

g. expand

Prompt

Write a function called `expand` that takes in a string `s` that is formatted with alternating letters and numbers. This function should return a new string where each letter is repeated in the resulting string X times, where X is the number that proceeds the letter.

For example:

- `expand('a4')` would return the string `'aaaa'`
- `expand('h3e1l3l1o2')` would return the string `'hhhelllloo'`

You may assume that **the given string will always be in a valid format**, so it will always be an even number in length and it will always contain letters followed by numbers. Letters will come first and numbers will come second.

If the given string is empty, return the empty string.

Hint: you can turn a character of a number into an integer with casting! `int('1')` → `1`

Starter code

```
def expand(s):
```

Example Input + Output

input	output	notes
expand('h3e1l3l1o2')	'hhhelllloo'	
expand('a4')	'aaaa'	
expand('b1b2')	'bbb'	repeated letters
expand('o1o1o1o1')	'oooo'	repeated letters
expand('o2r2a2n2g2e2')	'oorraannggee'	single occurrences of every letter yeild the same input string without numbers
expand('')	''	special return: empty string

Example Solution

▼ Click here for the solution

```
def expand(s):
    if s == '':
        return s
    else:
        result = ''
        for i in range(1, len(s), 2):
            repeat = s[i]
            letter = s[i - 1]
            result += letter * int(repeat)
        return result
```

Notes

▼ Click here for more information

- What's being assessed: for loops with start, stop, step defined, string manipulation, casting
- What's challenging: avoiding out-of-bounds indexing, casting string to integer to multiply, parsing through characters in pairs
- Source/Inspired by: compress problem from CSE 163, Summer 2021
- Written by: Paul Pham

Task Complexity

Branch	YES
Loop	YES
Nested Branch	YES