

General Notes

All in Python3.

0 Day 0: Hello, World.

```
# Read a full line of input from stdin and save it to our
    dynamically typed variable, input_string.
inputString = input()
print (inputString)
```

1 Day 1: Data Types

```
i = 4
d = 4.0
s = 'HackerRank '

# Declare second integer, double, and String variables.
i2 = int(input())    # read int
d2 = float(input())  # read double
s2 = input()         # read string

# print summed and concatenated values
print(i + i2)
print(d + d2)
print(s + s2)
```

2 Day 2: Operators

```
mealCost = float(input())
tipPercent = int(input())
taxPercent = int(input())

tip = mealCost * (tipPercent/100.)
tax = mealCost * (taxPercent/100.)

totalCost = mealCost + tip + tax
total = round(totalCost)

print ('The total meal cost is', int(total), 'dollars.')
```

3 Day 3: Intro to Conditional Statements

```
import sys

N = int(input().strip())

condition = 'Not Weird'

if N % 2 != 0:
    condition = 'Weird'
elif N % 2 == 0 and (N >= 6 and N <= 20):
    condition = 'Weird'
else:
    condition = 'Not Weird'

print(condition)
```

4 Day 4: Class vs. Instance

'''

Objective:

In this challenge, we are going to learn about the difference between a class and an instance; because this is an Object Oriented concept, it is only enabled in certain languages. Check out the Tutorial tab for learning materials and an instructional video!

Task:

Write a Person class with an instance variable, age, and a constructor that takes an integer, initialAge, as a parameter. The constructor must assign initialAge to age after confirming the argument passed as initialAge is not negative; if a negative argument is passed as initialAge, the constructor should set age to 0 and print Age is not valid, setting age to 0. In addition, you must write the following instance methods:

yearPasses() should increase the age instance variable by 1.

amIOld() should perform the following conditional actions:

If age < 13, print You are young.

If >= 13 and age < 18, print You are a teenager.

Otherwise, print You are old.

To help you learn by example and complete this challenge, much of the code is provided for you, but you'll be writing everything in the future. The code that creates each instance of your Person class is in the main method. Dont worry if you dont understand it all quite yet!

'''

class Person:

def __init__(self, initialAge):

Add some more code to run some checks on initialAge

self.age = 0

if initialAge < 0:

print ("Age is not valid, setting age to 0.")

else:

self.age = initialAge

def amIOld(self):

Do some computations in here and print out the correct statement to the console

if age < 13:

print("You are young.")

elif 13 <= age < 18:

```

        print("You are a teenager.")
    elif age >= 18:
        print("You are old.")

    def yearPasses(self):
        # Increment the age of the person in here
        global age #NPR: don't quite understand what global does
        here...
        age += 1

t = int(input())
for i in range(0, t):
    age = int(input())
    p = Person(age)
    p.amIOld()
    for j in range(0, 3):
        p.yearPasses()
    p.amIOld()
    print("")

```

5 Day 5: Loops

```
'''
```

Objective:

In this challenge, we are going to use loops to help us do some simple math. Check out the Tutorial tab to learn more.

Task

Given an integer, `n`, print its first multiples. Each multiple (where `i` is from 1 to 10) should be printed on a new line in the form: `N x i = result`.

```
'''
```

```
import sys
```

```
N = int(input().strip())
```

```
for ii in range(1, 11):  
    print (N, 'x', ii, '=', N*ii)
```

6 Day 6: Let's Review

7 Day 7: Arrays

8 Day 8: Dictionaries and Maps

9 Day 9: Recursion

10 Day 10: Binary Numbers

11 Day 11: 2D Arrays

12 Day 12: Inheritance

13 Day 13: Abstract Classes

14 Day 14: Scope

15 Day 15: Linked List

16 Day 16: Exceptions - String to Integer

17 Day 17: More Exceptions

18 Day 18: Queues and Stacks

19 Day 19: Interfaces

20 Day 20: Sorting

21 Day 21: Generics

22 Day 22: Binary Search Trees

23 Day 23: BST Level-Order Traversal

24 Day 24: More Linked Lists

25 Day 25: Running Time and Complexity

26 Day 26: Nested Logic

27 Day 27: Testing

28 Day 28: RegEx, Patterns, and Intro to Databases

29 Day 29: Bitwise AND
