You can view this report online at : https://www.hackerrank.com/x/tests/1726224/candidates/57321987/report

Full Name:	Instructor
Email:	aisha.batool@sse.habib.edu.pk
Test Name:	**CS101 - LW9 - Fall23**
Taken On:	22 Oct 2023 18:51:29 PKT
Time Taken:	6 min 57 sec/ 165 min
Work Experience:	> 5 years
Invited by:	Aisha
Skills Score:	
Tags Score:	CS101 10/10 Strings 10/10

100%

80/80

scored in **CS101 - LW9 - Fall23** in 6 min 57 sec on 22 Oct 2023 18:51:29 PKT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Hurricane Harvey - Variables and Inputs > Coding	2 min 3 sec	10/ 10	✓
Q2	Problem Solving - Pattern 4 > Coding	1 min 15 sec	10/ 10	✓
Q3	Breakdown > Coding	44 sec	10/ 10	✓
Q4	Stretch a string > Coding	49 sec	10/ 10	✓
Q5	Third or fifth > Coding	23 sec	10/ 10	✓
Q6	GPA > Coding	32 sec	10/ 10	✓
Q7	Odd Sum > Coding	31 sec	10/ 10	✓
Q8	Fibonacci Series > Coding	28 sec	10/ 10	✓

QUESTION 1

Correct Answer

Score 10

Hurricane Harvey - Variables and Inputs > Coding

QUESTION DESCRIPTION

Problem

Your relatives in Houston are reliving their Pakistan days by experiencing power cuts caused by Hurricane Harvey. They tell you that they are experiencing wind speeds up to 130 miles per hour. Your only measure of speed is through your car's speedometer which shows speed in km per hour.

You are required to take `mph` as input and prints its equivalent in km/h.

Interaction

The input comprises a single line containing a float denoting the value of `mph`.
The output must show a line stating `mph` and its equivalent in km/h as shown in sample.

Calculation

You may assume that **1 mile = 1.6 km**

Sample

```
Input: 130

Output: 130.0 mi/h are equivalent to 208.0 km/h.
```

Implementation notes

Strictly observe the output format.

Hint

To read floating-point number from command line in Python, we can use `input()` built-in function. Since, input() function reads a string from standard input, we can use `float()` function to convert the string to float.

The following is a simple code snippet to read a float into variable `x`.

```
x = float(input())
```

INTERVIEWER GUIDELINES

Solution

```
mph = float(input())
print(mph, 'mi/h are equivalent to', mph*1.6, 'km/h.')
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1 mph = float(input())
2 print(mph, 'mi/h are equivalent to', mph*1.6, 'km/h.')
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	5	0.059 sec	9.53 KB
Testcase 1	Easy	Hidden case	Success	5	0.0733 sec	9.61 KB

No Comments

QUESTION 2

Correct Answer

Score 10

Problem Solving - Pattern 4 › Coding

QUESTION DESCRIPTION

Problem

Write an *iterative* function named `pattern` to generate the following pattern for a given parameter, `n`.

Sample

```
>>> pattern(3)
1
2 1
4 2 1

>>> pattern(1)
1

>>> pattern(2)
1
2 1

>>> pattern(6)
1
2 1
4 2 1
8 4 2 1
16 8 4 2 1
32 16 8 4 2 1

>>> pattern(8)
1
2 1
4 2 1
8 4 2 1
16 8 4 2 1
32 16 8 4 2 1
64 32 16 8 4 2 1
128 64 32 16 8 4 2 1
```

Input

Input `n` from the console without any prompt.

Constraints

- `isinstance(n, int)` is `True`
- n >= 1

INTERVIEWER GUIDELINES

```
n=int(input())
def iterative_pattern(n):

    count=0

    line=""

    while count<n:

        line=str(2**count)+" "+line

        count=count+1

        print(line)


iterative_pattern(n)
```

CANDIDATE ANSWER

Language used: **Python 3**

```
n=int(input())
def pattern(n):
    count=0
    line=""
    while count<n:
        line=str(2**count)+" "+line
        count=count+1
        print(line)

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	Success	2	0.0512 sec	9.6 KB
TestCase 1	Easy	Hidden case	Success	2	0.0575 sec	9.47 KB
TestCase 2	Easy	Hidden case	Success	2	0.0622 sec	9.59 KB
TestCase 3	Easy	Sample case	Success	2	0.0601 sec	9.54 KB
Testcase 4	Easy	Sample case	Success	2	0.115 sec	9.3 KB

No Comments

QUESTION 3

Correct Answer

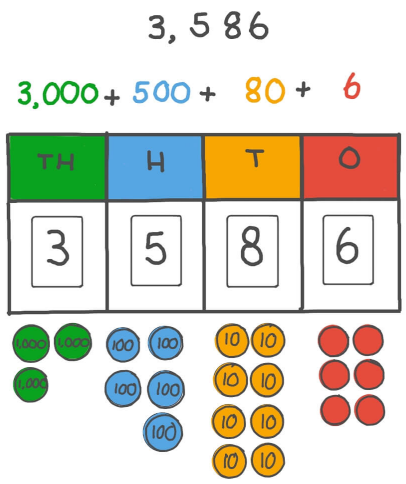
Score 10

Breakdown › Coding

QUESTION DESCRIPTION

Problem

You are given a 4-digit number, `num4`, which has exactly one digit in each of the *thousands*, *hundreds*, *tens*, and *units* position. For example, 3586 has 3 in the *thousands* position, 5 in the *hundreds* position, 8 in the *tens* position, and 6 in the *units* position.



Write a Python function, `breakdown()`, that takes `num4` as an argument and prints the digits in each of the positions following the format shown in the sample output.

Sample

```
>> breakdown(5327)

Thousands: 5
Hundreds: 3
Tens: 2
Units: 7
```

Constraints

- `num4` is a non-negative 4 digit integer.
- The function must not convert to a `str`.

INTERVIEWER GUIDELINES

Solution

```
def breakdown(num4):
    num4 = int(num4)
    n = num4//1000
    print("Thousands:", n)
    num4 = num4 - n*1000
    n = num4//100
    print("Hundreds:", n)
    num4 = num4 - n*100
    n = num4//10
    print("Tens:", n)
    num4 = num4 - n*10
    print("Units:", num4)
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1  # Enter your code here.
2  def breakdown(num4):
3      num4 = int(num4)
4      n = num4//1000
5      print("Thousands:", n)
6      num4 = num4 - n*1000
7      n = num4//100
8      print("Hundreds:", n)
9      num4 = num4 - n*100
10     n = num4//10
11     print("Tens:", n)
12     num4 = num4 - n*10
13     print("Units:", num4)
14
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Medium	Sample case	Success	5	0.101 sec	11.8 KB
Testcase 1	Medium	Hidden case	Success	5	0.0901 sec	11.6 KB

No Comments

QUESTION 4

Stretch a string › Coding CS101 Strings

Correct Answer

Score 10

QUESTION DESCRIPTION

Challenge

Write a function `stretch(s)` that takes a string argument `s` and returns a new string such that the first character appears once, the second character is repeated twice, the third character is repeated thrice, and so on.

Sample

```
>>> print(stretch('Gum'))
Guummm
>>> print(stretch('Pizza!'))
Piizzzzzzaaaaa!!!!!!
```

INTERVIEWER GUIDELINES

Solution

```
def stretch(s):
    newStr = ""
    i = 1
    for st in s:
        newStr = newStr + (st*i)
        i = i + 1
    return newStr
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1 def stretch(s):
2     newStr = ""
3     i = 1
4     for st in s:
5         newStr = newStr + (st*i)
6         i = i + 1
7     return newStr
8
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	2.5	0.061 sec	9.24 KB
Testcase 1	Easy	Sample case	Success	2.5	0.0657 sec	9.42 KB
Testcase 2	Easy	Hidden case	Success	2.5	0.0583 sec	9.3 KB
Testcase 3	Easy	Hidden case	Success	2.5	0.0438 sec	9.58 KB

No Comments

QUESTION 5

Correct Answer

Score 10

Third or fifth › Coding

QUESTION DESCRIPTION

Challenge

Write a function `third_or_fifth(s)` that returns every third and every fifth letter in the string `s`.

Sample

```
>>> print(third_or_fifth('12345678901234567890123456789O'))
35690258014570
>>> print(third_or_fifth('pomegranate'))
mgrat
```

INTERVIEWER GUIDELINES

```
def third_or_fifth(s):
    st = ""
    for i in range(len(s)):
        if ((i+1)%3==0) or ((i+1)%5==0):
          st+=s[i]
    return st
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1  def third_or_fifth(s):
2      st = ""
3      for i in range(len(s)):
4          if ((i+1)%3==0) or ((i+1)%5==0):
5            st+=s[i]
6      return st
7
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	2.5	0.071 sec	9.54 KB
Testcase 1	Easy	Sample case	Success	2.5	0.0942 sec	9.42 KB
Testcase 2	Easy	Sample case	Success	2.5	0.0678 sec	9.18 KB
Testcase 3	Easy	Sample case	Success	2.5	0.0695 sec	9.56 KB

No Comments

QUESTION 6

Correct Answer

Score 10

GPA › Coding

QUESTION DESCRIPTION

Problem

Your university follows the grading scheme below.

Marks	Grade	Grade Point
[95, 100]	A+	4.00
[90, 95)	A	4.00
[85, 90)	A-	3.67
[80, 85)	B+	3.33

[75, 80)	B	3.00
[70, 75)	B-	2.67
[67, 70)	C+	2.33
[63, 67)	C	2.00
[60, 63)	C-	1.67
[0, 60)	F	0.00

Write a function named `gpa_calculator` that takes `marks_1, marks_2, marks_3, marks_4` as parameters and outputs the corresponding grades and GPA. The GPA is the arithmetic mean of the grade points. You may assume that all 4 courses are 3 credits hours each.

Hint: Make helper functions, `grade` that returns the grade corresponding to a mark, and `points` that returns the grade points corresponding to a grade.

Sample

```
>>> gpa_calculator(100, 77, 64, 30)
Your grades are A+ B C F
Your GPA is 2.25
>>> gpa_calculator(75, 91, 66, 50)
Your grades are B A C F
Your GPA is 2.25
```

Constraints

- `marks_1, marks_2, marks_3, marks_4` are *integers*
- 0 <= `marks_1, marks_2, marks_3, marks_4` <= 100

INTERVIEWER GUIDELINES

Solution

```
def grade(marks):
    if 0 <= marks < 60:
        return 'F'
    elif 60 <= marks < 63:
        return 'C-'
    elif 63 <= marks < 67:
        return 'C'
    elif 67 <= marks < 70:
        return 'C+'
    elif 70 <= marks < 75:
        return 'B-'
    elif 75 <= marks < 80:
        return 'B'
    elif 80 <= marks < 85:
        return 'B+'
    elif 85 <= marks < 90:
        return 'A-'
    elif 90 <= marks < 95:
        return 'A'
    elif 95 <= marks <= 100:
        return 'A+'

def points(grade):
    if grade == 'A+':
        return 4
    elif grade == 'A':
        return 4
    elif grade == 'A-':
        return 3.67
```

```
    elif grade == 'B+':
        return 3.33
    elif grade == 'B':
        return 3
    elif grade == 'B-':
        return 2.67
    elif grade == 'C+':
        return 2.33
    elif grade == 'C':
        return 2
    elif grade == 'C-':
        return 1.67
    elif grade == 'F':
        return 0

def gpa_calculator(marks_1, marks_2, marks_3, marks_4):
    grade_1 = grade(marks_1)
    points_1 = points(grade_1)
    grade_2 = grade(marks_2)
    points_2 = points(grade_2)
    grade_3 = grade(marks_3)
    points_3 = points(grade_3)
    grade_4 = grade(marks_4)
    points_4 = points(grade_4)
    gpa = (3*points_1 + 3*points_2 + 3*points_3 + 3*points_4)/12
    print('Your grades are', grade_1, grade_2, grade_3, grade_4)
    print('Your GPA is', gpa)
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1
2  def grade(marks):
3      if 0 <= marks < 60:
4          return 'F'
5      elif 60 <= marks < 63:
6          return 'C-'
7      elif 63 <= marks < 67:
8          return 'C'
9      elif 67 <= marks < 70:
10         return 'C+'
11     elif 70 <= marks < 75:
12         return 'B-'
13     elif 75 <= marks < 80:
14         return 'B'
15     elif 80 <= marks < 85:
16         return 'B+'
17     elif 85 <= marks < 90:
18         return 'A-'
19     elif 90 <= marks < 95:
20         return 'A'
21     elif 95 <= marks <= 100:
22         return 'A+'
23
24 def points(grade):
25     if grade == 'A+':
26         return 4
27     elif grade == 'A':
28         return 4
29     elif grade == 'A-':
30         return 3.67
```

```
    elif grade == 'B+':
        return 3.33
    elif grade == 'B':
        return 3
    elif grade == 'B-':
        return 2.67
    elif grade == 'C+':
        return 2.33
    elif grade == 'C':
        return 2
    elif grade == 'C-':
        return 1.67
    elif grade == 'F':
        return 0

def gpa_calculator(marks_1, marks_2, marks_3, marks_4):
    grade_1 = grade(marks_1)
    points_1 = points(grade_1)
    grade_2 = grade(marks_2)
    points_2 = points(grade_2)
    grade_3 = grade(marks_3)
    points_3 = points(grade_3)
    grade_4 = grade(marks_4)
    points_4 = points(grade_4)
    gpa = (3*points_1 + 3*points_2 + 3*points_3 + 3*points_4)/12
    print('Your grades are', grade_1, grade_2, grade_3, grade_4)
    print('Your GPA is', gpa)

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Medium	Sample case	Success	2.5	0.0541 sec	9.61 KB
Testcase 1	Medium	Sample case	Success	2.5	0.0972 sec	9.82 KB
Testcase 2	Medium	Sample case	Success	2.5	0.0657 sec	9.9 KB
Testcase 3	Medium	Sample case	Success	2.5	0.079 sec	9.86 KB

No Comments

QUESTION 7

Correct Answer

Score 10

Odd Sum › Coding

QUESTION DESCRIPTION

Problem

Write an *iterative* function named `sum_odd` that takes two parameters called `a` and `b` and returns the sum of all odd numbers between `a` and `b` inclusive.

Sample

```
>>> sum_odd(1,1)
1
>>> sum_odd(2,2)
0
>>> sum_odd(1,2)
1
>>> sum_odd(2,3)
3
>>> sum_odd(13, 2)
48
```

```
>>> sum_odd(6.7, 10)
Error: bad argument. sum_odd is defined for integers only.
```

Constraints

None. Write appropriate *guardians* in your function.

Hint

Use the type function to check the type of a value/variable to confirm if it is the correct type or not. For Example:

```
>> type(3) == int              # This will return True as 3 is an integer
True

>> type('3') == int            # This will return False as '3' is of type
string
False

>> type(True) == bool          # This will return True as True is of type
boolean
True

>> type(3.14) != int           # This will return True as 3.14 is indeed
not of type int
False
```

INTERVIEWER GUIDELINES

Solution

```
def sum_odd(a, b):
    if not (isinstance(a, int) and isinstance(b,int)):
        print('Error: bad argument. sum_odd is defined for integers
only.')
        return
    if b < a:
        a, b = b, a
    total = 0
    if a % 2  == 0:
        a += 1
    while a <= b:
        total += a
        a += 2
    return total
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1  # Enter your code here.
2  def sum_odd(a, b):
3      if not (isinstance(a, int) and isinstance(b,int)):
4          print('Error: bad argument. sum_odd is defined for integers only.')
5          return
6      if b < a:
7          a, b = b, a
8      total = 0
9      if a % 2  == 0:
10         a += 1
11     while a <= b:
```

```
        total += a
        a += 2
    return total

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	Success	1.66	0.0598 sec	9.39 KB
TestCase 1	Easy	Sample case	Success	1.66	0.0614 sec	9.46 KB
TestCase 2	Easy	Sample case	Success	1.66	0.0544 sec	9.34 KB
TestCase 3	Easy	Sample case	Success	1.66	0.0551 sec	9.16 KB
TestCase 4	Easy	Sample case	Success	1.66	0.054 sec	9.45 KB
TestCase 5	Easy	Hidden case	Success	1.7	0.0572 sec	9.66 KB

No Comments

QUESTION 8

Correct Answer

Score 10

Fibonacci Series › Coding

QUESTION DESCRIPTION

Problem

The Fibonacci series begins with 0 and 1. Each subsequent term is computed as the sum of the last 2 terms, thus yielding

```
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...
```

Write an *iterative* function, `fibonacci_sequence()`, that takes a parameter named `n` and prints the series up to the `n`-th term.

Sample

```
>>> fibonacci_sequence(0)
0
>>> fibonacci_sequence(1)
0, 1
>>> fibonacci_sequence(5)
0, 1, 1, 2, 3, 5
>>> fibonacci_sequence(3.14)
Error: bad argument. fibonacci is defined for positive integers only.
>>> fibonacci_sequence(-6)
Error: bad argument. fibonacci is defined for positive integers only.
```

Constraints

None. Write appropriate *guardians* in your function.

Hint

Use the type function to check the type of a value/variable to confirm if it is the correct type or not. For Example:

```
>> type(3) == int             # This will return True as 3 is an integer
True

>> type('3') == int           # This will return False as '3' is of type
string
False

>> type(True) == bool         # This will return True as True is of type
```

```
boolean
True

>> type(3.14) != int          # This will return True as 3.14 is indeed
not of type int
False
```

INTERVIEWER GUIDELINES

Solution

```
def fibonacci_sequence(n):
    if not isinstance(n, int) or n < 0:
        print('Error: bad argument. fibonacci is defined for positive
integers only.')
        return
    a, b, c = 0, 1, 1
    while n > 0:
        print(a, end = ', ')
        a, b, c = b, c, b+c
        n -= 1
    print(a)
```

CANDIDATE ANSWER

Language used: **Python 3**

```
# Enter your code here.
def fibonacci_sequence(n):
    if not isinstance(n, int) or n < 0:
        print('Error: bad argument. fibonacci is defined for positive
integers only.')
        return
    a, b, c = 0, 1, 1
    while n > 0:
        print(a, end = ', ')
        a, b, c = b, c, b+c
        n -= 1
    print(a)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Medium	Sample case	Success	3.33	0.0326 sec	9.53 KB
TestCase 1	Medium	Sample case	Success	3.33	0.117 sec	9.26 KB
TestCase 2	Medium	Sample case	Success	3.34	0.039 sec	9.31 KB

No Comments