

Computing

# Lesson 3: Tracing Algorithms

**Algorithms**

Kashif Ahmed

*Materials from the Teach Computing Curriculum created by the National Centre for Computing Education*



# Task 1 - Tracing code - part 1

## Russian multiplication

The Python code in **Figure 1** is an implementation of the Russian multiplication algorithm. This method calculates the product of two numbers as a sum by using **integer division** and **modulo (MOD)**. Use the table below to help you investigate the algorithm in Python.



# Task 1 - Tracing code - part 1

	Explanation	Python example
<b>Modulo (MOD)</b>	Calculates the <b>remainder</b> of a division. For example, 7 MOD 3 will calculate as 1.	$7 \% 3$
<b>Integer division</b>	Calculates the <b>whole</b> number of times the divisor (3) will go into the dividend (7). For example, $7 \div 3$ will calculate as 2.	$7 // 3$



# Task 1 - Tracing code - part 1

```
1 print("Numbers:")
2 a = int(input())
3 b = int(input())
4 sum = 0
5 while b > 0:
6     if b % 2 == 1:
7         sum = sum + a
8         a = 2*a
9         b = b // 2
10 print(sum)
```

Figure 1



# Task 1 - Tracing code - part 1

**State** the result of the following calculation in Python: `14 % 4`

**State** the result of the following calculation in Python: `28 // 5`

Complete the trace table below using the algorithm in Figure 1. The values of a and b have been provided and the first iteration of the while loop has been filled in for you.



# Task 1 - Tracing code - part 1

Line	a	b	sum	Condition	Output
1					"Numbers:"
2	11				
3		7			
4			0		
5				True	
6				True	
7			11		



# Task 1 - Tracing code - part 1

Line	a	b	sum	Condition	Output
8	22				
9		3			
5				True	



# Task 1 - Tracing code - part 1

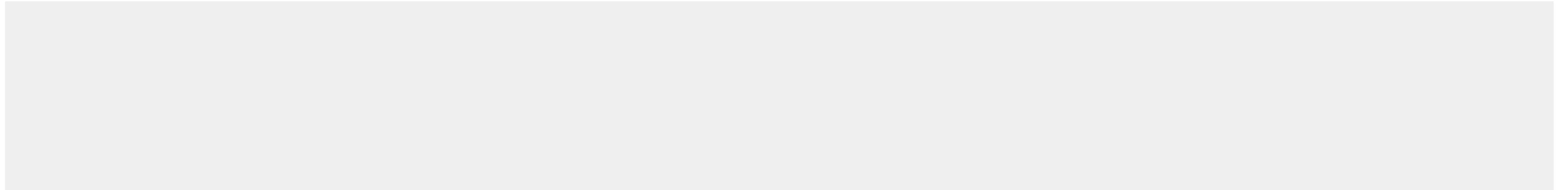
Line	a	b	sum	Condition	Output





# Task 1 - Tracing code - part 1

Does the algorithm in **Figure 1** loop infinitely, or not? Explain your answer.



# Task 1 - Tracing code - part 2

Lowest number in a list

In this task you are going to analyse a piece of code to check whether it is working correctly. The program is meant to find the lowest number from a list of integers called **items** and store the lowest value from this list in the variable **lowest**.



# Task 1 - Tracing code - part 2

```
1 lowest = items[0]
2 for current in range(1, len(items)):
3     if lowest < items[current]:
4         lowest = items[current]
```

Figure 2

**Complete** the trace table below using the algorithm in **Figure 2**. The list of items and the first two lines of code have been filled in for you.



# Task 1 - Tracing code - part 2

				items				
Line	lowest	current	Condition	[0]	[1]	[2]	[3]	[4]
				24	16	35	42	7
1	24							
2		1						



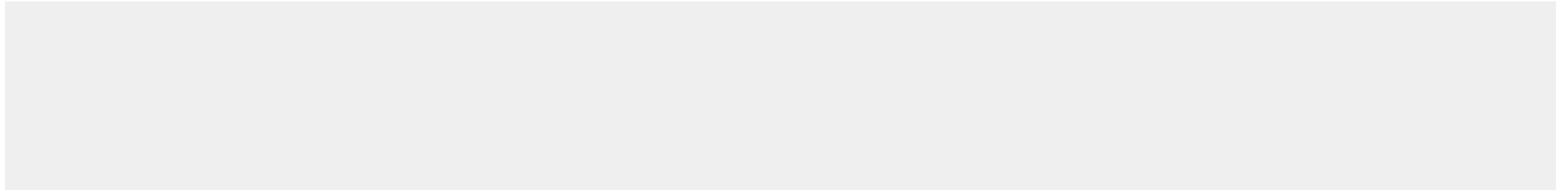
# Task 1 - Tracing code - part 2

				items				
Line	lowest	current	Condition	[0]	[1]	[2]	[3]	[4]



# Task 1 - Tracing code - part 2

**Explain** whether the algorithm in **Figure 2** works as intended.



# Task 1 - Tracing code - part 3

## Nested loops

The algorithm in **Figure 3** contains a nested loop: a loop within a loop. The outer **for** loop has a lower number of iterations than the inner loop. **Note** that the end number of the range is not included in the generated sequence because it is used as the stop point.



# Task 1 - Tracing code - part 3

```
1 total = 0
2 for i in range(1,3):
3     for j in range(2,5):
4         total = total + j
5     print(total)
```

Figure 3

**Complete** the trace table below using the algorithm in **Figure 3**.





# Task 1 - Tracing code - part 3

Line	total	i	j	Output



# Task 1 - Tracing code - part 3

Line	total	i	j	Output

