Python

Python and R for Data Science

Data Science and Management

Exercise 1: sum of two integers

1. Define two variables `a` and `b` with intial values `10` and `12`, respectively.
2. Define the variable `c` as the sum of `a` and `b`
3. Print `c`

In [3]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [4]:
```
try: assert c == (a + b) and c == 22 and not print("Test passed")
except: print('Test failed')
```

```
Test failed
```

Exercise 2: area of a triangle

1. Compute the area of a triangle with:
    - `base`: `5.0`
    - `height`: `7.5`
2. Store the result in the variable `area`
3. Print the area

In [6]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [7]:
```
try: assert area == 17.5 and not print("Test passed")
except: print('Test failed')
```

```
Test failed
```

Exercise 3: volume of a cube

1. Compute the volume of a cube with `side` equal to `8.0`
2. Store the result in the variable `volume`
3. Print the area

In [9]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [10]:
```
try: assert volume == 512 and not print("Test passed")
except: print('Test failed')
```

```
Test failed
```

Exercise 4: compute equation

1. Define:
    - `x` equal to `10`
    - `y` equal to `20`
2. Compute the result of:

$$(x - 4)^3 + 5$$

$$4 \cdot (y \mod 3)$$

3. Store the result in the variable `result`
4. Print the result

In [12]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [13]:
```
try: assert result == 27.625 and not print("Test passed")
except: print('Test failed')
```

```
Test failed
```

Exercise 5: count characters

1. Define the string `s` equal to `Bazinga!`
2. Count the number of characters in `s` (without using a loop!) and store the result in the variable `length`
3. Print `length`

In [15]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [16]:
```
try: assert length == 8 and not print("Test passed")
except: print('Test failed')
```

```
Test failed
```

Exercise 6: concatenate strings

1. Define one string `s1` with the content `Francesco`
2. Define one string `s2` with the content ` `
3. Define one string `s3` with the content `Totti`
4. Concatenate `s1`, `s2`, and `s3` into `s4`
5. Print the concatenated string `s4`

In [18]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [19]:
```
try: assert s4 == "Francesco Totti" and not print("Test passed")
except: print('Test failed')
```

```
Test failed
```

Exercise 7: get the last three chars from a string

1. Define the string `s1` with the content `Totti`
2. Extract the last three characters of `s1` into `s2`
3. Print the string `s2`

In [21]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [22]:
```
try: assert s2 == "ti" and not print("Test passed")
except: print('Test failed')
```

```
Test failed
```

Exercise 8: convert types

1. Define the **integer** `n1` equal to `10`
2. Define the **string** `s1` equal to `"20.5"`
3. Convert `n1` and `s1` to `float` and then sum the two values into `f1`
4. Print `f1`

In [24]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [25]:
```
try: type(s1) == str and type(n1) == int and f1 == 30.5 and not print('
except: print('Test failed')
```

```
Test failed
```

Exercise 9: largest average of two sequences

1. Define the numbers `a1`, `a2`, and `a3` with values equal to `10`, `12`, `8`, respectively
2. Define the numbers `b1`, `b2`, and `b3` with values equal to `7`, `10`, `18`, respectively
3. Compute the average of:
    - `a1`, `a2`, and `a3` into `avg_a`
    - `b1`, `b2`, and `b3` into `avg_b`
4. Conditionally define the string `s` to be equal to:
    - if `avg_a` is larger than `avg_b`: `Sequence A is on average larger than sequence B`
    - if `avg_b` is smaller than `avg_b`: `Sequence A is on average smaller than sequence B`
    - otherwise: `avg_b`: `The two sequences have the same average`
5. Print `avg_a`, `avg_b`, `s`

In [27]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [28]:
```
try: assert round(avg_a, 1) == 24.7 and avg_b == 23 and s == "Sequence
except: print('Test failed')
```

```
Test failed
```

Exercise 10: perfect square

1. Define the numbers `n1` and `n3` with values equal to `9` and `12``, respectively

2. Check whether these numbers are *perfect squares*:

   > *In mathematics, a square number or perfect square is an integer that is the square of an integer;[1] in other words, it is the product of some integer with itself.*

   Store the result of the checks in `is_n1_perfect_square` and `is_n2_perfect_square`

3. Print `is_n1_perfect_square` and `is_n2_perfect_square`

In [30]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [31]:
```
try: assert is_n1_perfect_square and not is_n2_perfect_square and not p
except: print('Test failed')
```

```
Test failed
```

Exercise 11: count digits

1. Define the integer `n1` equal to `123450`
2. Using a loop, count the number of digits in `n1` and store the result in `ndigits`
3. print `ndigits`

In [33]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [34]:
```
try: assert ndigits == 6 and not print("Test passed")
except: print('Test failed')
```

```
Test failed
```

Exercise 12: quotient and remainder by hand

1. Define the integer `n1` equal to `123450`
2. Define the integer `n2` equal to `57`
3. Using a loop, without using the `/` and/or `//` and/or `%` operators, compute the quotient `q` and the remainder `r` of the integer division of `n1` by `n2`
4. print `r` and `q`

In [36]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [37]:
```
try: assert q == int(n1/n2) and r == (n1 % n2) and not print("Test pass
except: print('Test failed')
```

```
Test failed
```

Exercise 13: sum of the first n numbers

1. Define the integer `n` equal to `100`
2. Using a loop compute the sum of the first `n` numbers (starting from `1`), storing the result into `s`
3. print `s`

In [39]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [40]:
```
try: assert s == 5050 and not print("Test passed")
except: print('Test failed')
```

```
Test failed
```

Exercise 14: sum of the prime numbers

1. Define the integer `n` equal to `100`
2. Using a loop compute the sum of **prime** numbers up to `n`, storing the result into `s`
3. print `s`

In [42]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [43]:
```
try: assert s == 1060 and not print("Test passed")
except: print('Test failed')
```

```
Test failed
```

Exercise 15: prefixes of a string

1. Define the string `s` equal to `The Big Bang Theory`
2. Create the empty list `p`
3. Using a loop, add all prefixes of `s` to `p` (note: `The Big Bang Theory` is a prefix of `The Big Bang Theory`)
4. print `p`

In [45]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [46]:
```
try: assert sorted(p) == ['T', 'Th', 'The', 'The ', 'The B', 'The Bi',
except: print('Test failed')
```

```
Test failed
```

Exercise 16: check postfixes of a string

1. Define the string `s` equal to `The Big Bang Theory`
2. Define the list `p` equal to `["y", "ry", "ery", ""]`
3. Remove from `p` any string that is not a postfix of `s` (note: `""` is a postfix of `s`)
4. print `p`

In [48]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [49]:
```
try: assert p == ['y', 'ry', ''] and not print("Test passed")
except: print('Test failed')
```

```
Test failed
```

Exercise 17: max of a list

Define a function `max_from_list` that:

- takes as arguments a list of integers
- returns:
    - if the list is not empty: the maximum value in the list
    - otherwise: `None`

Do not use the built-in function `max` in this exercise.

In [51]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [52]:
```
try: max_from_list([]) == None and max_from_list([1, 2, 3]) == 3 and nc
except: print('Test failed')
```

```
Test failed
```

Exercise 19: prime numbers

Define a function `is_prime` that:

- takes as arguments a list `L` of positive integers
- returns:
    - if the list is not empty: a new list where the i-th element is a boolean asserting whether the i-th element from `L` is a prime number
    - otherwise: `[]`

In [54]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [55]:
```
try: assert is_prime([]) == [] and is_prime([3, 4, 9, 11]) == [True, Fa
except: print('Test failed')
```

```
Test failed
```

Exercise 20: word frequency

Define a function `count_freq` that:

- takes as arguments:
    - a string `s`
    - a list `L` of words
- returns:
    - if the list is not empty: a new list where the i-th element is the number of occurences in `s` of the i-th word from `L`
    - otherwise: `[]`

In [ ]:
```
# Solution goes here
```

Test your code

Run this code to test your solution:

In [1]:
```
try: assert count_freq("test", []) == [] and count_freq("Aejeje", ["e",
except: print('Test failed')
```

```
Test failed
```