W9.2 Applied

Catch the exception if you can!

In this activity, please attempt to predict the outcome of the following code *without using the Python interpreter*.

Question 1

```
'1.0' / 0.0
```

- `ValueError`
- `ZeroDivisionError`
- `TypeError`
- `NameError`

Question 2

```
1.0 / 0.0
```

- `ValueError`
- `ZeroDivisionError`
- `TypeError`

- `NameError`

Question 3

```
int(a)
```

- `ValueError`
- `NameError`
- `TypeError`
- `KeyError`

Question 4

```
int('a')
```

- `ValueError`
- `NameError`
- `TypeError`
- `KeyError`

Question 5

```
int(a))
```

- `ValueError`
- `NameError`
- `TypeError`
- `SyntaxError`

Question 6

```
b = dict()
b[1]
```

- `ValueError`
- `IndexError`
- `TypeError`
- `KeyError`

Question 7

```
b = list()
b[1]
```

- `ValueError`
- `IndexError`
- `TypeError`
- `KeyError`

Question 8

```
a = []
b = dict()
b[a]
```

- `ValueError`
- `IndexError`
- `TypeError`
- `KeyError`

Question 9

```
assert False
```

- `AssertionError`

- `TypeError`
- `ValueError`
- No error

Question 10

```
assert True
```

- AssertionError
- TypeError
- ValueError
- No error

Make Assertions

In this activity, you will modify the functions you have implemented for 'Re-implement Built-in Functions' in W5.2 Applied and use `assert` statements to enforce the provided type annotations and any additional assumptions specified for both the input and the output of the functions. Namely, the functions you will be modifying are provided below together with their original instructions:

- `my_any`: Write a function `my_any(lst)` that models a limited version of the behaviour of the built-in function `any()`. You may use the built-in function `bool()`.
  Input: a list of objects `lst`.
  Output: a Boolean: `True` if at least one object in `lst` has a truth value of `True`; otherwise `False`.
  Think about the behaviour of the function for the empty list as input.
- `my_abs`: Write a function `my_abs(num)` that models a limited version of the behaviour of the built-in function (it is not necessary to deal with complex numbers).
  Input: a float `num`.
  Output: a float that is the absolute value of `num`.
- `my_enumerate`: Write a function `my_enumerate(lst)` that models a simplified version of the behaviour of the built-in function. In Python, `enumerate()` returns an enumerate object – enumerate is a specific type that is different to the list type.
  Input: a list `lst`.
  Output: a list of tuples, where each tuple is of the form `(i, element)`; `i` is the index of the corresponding element within `lst`.
- `my_sum`: Write a function `my_sum(lst)` that models a limited version of the behaviour of the built-in function.
  Input: a list of numbers `lst` that all have to be of type `float`.
  Output: the sum (of type `float`) of all numbers in `lst`.
- `my_is_in`: Write a function `my_is_in(char, string)` that models a limited version of the behaviour of the `in` keyword (for more details, refer to the Python docs). You **cannot** use `char in string` in your function, but you **can use a for loop** which is a different use of the `in` keyword.
  Input: a string `char` of a single character that **has to be alphanumeric** for the purposes of this exercise and a string `string` comprising arbitrary characters.
  Output: a Boolean, `True` if `char` is found in `string`; otherwise `False`.

  > Note that Python's `in` keyword is more powerful than the function `my_is_in()` in that it is able to test whether a string appears as a substring of another string. You are asked here only to implement the simpler function that tests if a single character appears in a string.

- `my_is_sorted`: Write a function `my_is_sorted(lst)` that checks whether a given list of comparable elements is sorted from smallest to largest.
  Input: a list `lst` of comparable elements.

Output: a Boolean, `True` if for all items in the list `lst`, the item at index `i` is less than or equal to the item at index `i+1`; otherwise `False`.

What does it mean for a pair of elements (e.g., lst[i] and lst[i+1]) to be comparable?

Read a table of integers

Write a function `read_file(file_name)` in the Python file named `read_file_1.py` that takes in a file name `file_name`, attempts to read the contents of the file located at `file_name` and outputs a list of lists of integers representing the contents of `file`. If the file is not located at `file_name`, the function returns the string:

`[Errno 2] No such file or directory:` *file*

where *file* is the argument of the function. Moreover, if any element of the content is not an integer, that element is replaced by the string `'n/a'`. You can assume the elements are separated by a comma (i.e., `,`).

For example, for the file `file_int.txt` that is provided, the function call `read_file('file_int.txt')` should return the following list of lists:

```
[[3, 7, 2, 5, 3, 1, 4, 8, 1],
 [1, 3, 'n/a', 2, 5, 0, 1, 'n/a'],
 [6, 3, 'n/a', 3, 1, 'n/a', 5, 3, 0],
 ['n/a', 'n/a', 'n/a', 'n/a', 'n/a', 'n/a'],
 [5, 3, 1, 4, 8, 1]]
```

Similarly, the function call `read_file('file_int_na.txt')` should return the following string:

```
[Errno 2] No such file or directory: 'file_int_na.txt'
```

Read student information

Write a function in the Python file named `read_file_2.py` that takes in a `file_name`, reads the contents of the file located at `file_name` and outputs a list of tuples representing its contents. The function `read_file` should check the validity of each line as follows:

- If there are more or less than three columns, then ignore that row and print the following message:

  `Missing or have extra column(s).`
  row
  `is ignored!`

  Where *row* is the input row.
- The first column denotes the name of the student. If the name is in the list `forbidden_names`, then ignore that row and print the following message:

  `Name is in the forbidden list of names.`
  row
  `is ignored!`

- The second column denotes the age of the student. If the age is not an integer number, then ignore that row and print the following message:

  `Age has to be an integer number.`
  row
  `is ignored!`
- The third column denotes the campus of the student. If the campus is not in the list `campuses`, then ignore that row and print the following message:

  `Incorrect campus.`
  row
  `is ignored!`

For example, for the file `'students.csv'` that is provided, the function call `read_file('students.csv')` should return the following list of tuples:

```
[('Billy Real', 19, 'Australia'), ('Eduardo Milin', 35, 'Malaysia'), ('Oskar Vilimar'
, 26, 'Australia'), ('Will Pompei', 27, 'Malaysia')]
```

and display the following error messages:

```
Age has to be an integer number.
Lisa Mei,twenty two,Australia
is ignored!
Incorrect campus.
Ellie Troy,22,Austria
is ignored!
Name is in the forbidden list of names.
hacker,33,Australia
is ignored!
Missing or have extra column(s)
Allie Millis,21,Australia,Hello!
is ignored!
Missing or have extra column(s)
54,Australia
is ignored!
```

Feedback

Question 1

Feedback

What worked best in this lesson?

No response

Question 2

Feedback

What needs improvement most?

No response