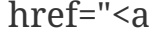
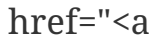
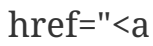
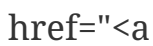
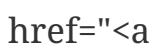
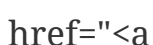


PythonTools

<https://github.com/KasparJohannesSchneider/PythonTools/actions/workflows/test.yml>  <https://github.com/KasparJohannesSchneider/PythonTools/actions/workflows/doc.yml>  <https://codecov.io/gh/KasparJohannesSchneider/PythonTools>  <https://lgtm.com/projects/g/KasparJohannesSchneider/PythonTools/alerts>  PythonTools <https://lgtm.com/projects/g/KasparJohannesSchneider/PythonTools/alerts>  PythonTools <https://github.com/KasparJohannesSchneider/PythonTools/blob/main/LICENCE>  License MIT  <https://www.python.org/>  Language Python 3.7

A collection of tools for Python that I created for my personal use.

Table of Contents

1. Import	1
2. Debug Tools	1
2.1. Debug Wrapper <code>@debug</code>	2
2.2. Timer Wrapper <code>@timer</code>	2
2.3. Run Function and get STDOUT <code>run_fct_get_stdout(fct: callable, *args) → str</code>	3
3. Math Tools	3
3.1. Sum from 1 to n <code>sum_1_n(n: int) → int</code>	3
3.2. Lower Triangular Number <code>ltm(n: int) → int</code>	3
3.3. Is Triangular? <code>is_triangular(n: int) → bool</code>	4

1. Import

```
import python_tools as pt
```

2. Debug Tools

2.1. Debug Wrapper @debug

A debug wrapper that prints some useful information about a function call.

2.1.1. How to use

This wrapper can be used by placing the corresponding decorator above the declaration of the function.

```
import python_tools as pt

@pt.debug
def a_function(x, y, z):
    pass
```

2.1.2. Expected output

```
--debug--debug--debug--debug--debug--debug--debug--debug--debug--
-- Function: a_function(x, y, z)
-- Arguments: (1, 2, 3)
-- Returned: None
-- Time elapsed [s]: 0.0
--debug--debug--debug--debug--debug--debug--debug--debug--debug--
```

2.2. Timer Wrapper @timer

2.2.1. How to use

This wrapper can be used by placing the corresponding decorator above the declaration of the function.

```
import python_tools as pt

@pt.timer
def a_function(x, y, z):
    pass
```

2.2.2. Expected output

```
--timer--timer--timer--timer--timer--timer--timer--timer--timer--timer--
-- Function: a_function(x, y, z)
-- Time elapsed [s]: 0.0
--timer--timer--timer--timer--timer--timer--timer--timer--timer--timer--
```

2.3. Run Function and get STDOUT

run_fct_get_stdout(fct: callable, *args) → str:

Runs a function and collects stdout during the execution of said function and returns the collected stdout as a string.

2.3.1. How to use

```
>>> import python_tools as pt
>>> pt.run_fct_get_stdout(print, 'Hello world!')
'Hello world!\n'
```

3. Math Tools

3.1. Sum from 1 to n **sum_1_n(n: int) → int**

This function calculates the sum of all numbers from 1 to **n**.

[Wikipedia: 1 + 2 + 3 + 4 + ...](#)

3.1.1. Example

```
>>> import python_tools as pt
>>> pt.sum_1_n(10)
55
```

3.2. Lower Triangular Number **ltm(n: int) → int**

This function returns **n** if it is a triangular number, or the next lower triangular number.

[Wikipedia: Triangular number](#)

3.2.1. Example

```
>>> import python_tools as pt
>>> pt.ltm(16)
15
```

3.3. Is Triangular? `is_triangular(n: int) → bool`

This function checks if a number is triangular.

[Wikipedia: Triangular number](#)

3.3.1. Example

```
>>> import python_tools as pt
>>> pt.is_triangular(15)
True
>>> pt.is_triangular(16)
False
>>> pt.is_triangular(21)
True
```