0x02-python-import\_modules

**0x02-python-import\_modules**

**================================**

## README.md

# Python - import & modules

In this project, I learned about importing and using functions and creating

modules in Python. I further practiced using the builtin function

`dir()` and using command line arguments within Python programs.

## Tasks :page\_with\_curl:

\* \*\*0. Import a simple function from a simple file\*\*

  \* [0-add.py](./0-add.py): Python program that imports the function

  `def add(a, b):` from the file [add\_0.py](./add\_0.py) and prints the

  result of the addition `1 + 2 = 3`.

  \* Output: `<a value> + <b value> = <add(a, b) value>` followed by a new line.

\* \*\*1. My first toolbox!\*\*

  \* [1-calculation.py](./1-calculation.py): Python program that imports functions

  from the file [calculator\_1.py](./1-calculator.py) and prints the result

  of the addition, subtraction, multiplication and division of `10` and `5`.

  \* Output: `<a value> <operator> <b value> = <operation(a, b) value>` followed by a new line.

\* \*\*2. How to make a script dynamic!\*\*

  \* [2-args.py](./2-args.py): Python program that prints the number of

  and list of its arguments.

  \* Output: `[Number of arguments] argument` (if number is one) or `arguments` (otherwise), followed by:

    \* `:` (or `.` if no argumets were passed), followed by

    \* A new line, followed by

    \* One argument per line - the position of the argument (starting at `1`) followed by `:` followed by the argument value and another new line.

\* \*\*3. Infinite addition\*\*

  \* [3-infinite\_add.py](./3-infinite\_add.py): Python program that prints the result of the

  addition of all arguments.

  \* Output: Sum of the arguments followed by a new line.

\* \*\*4. Who are you?\*\*

  \* [4-hidden\_discovery.py](./4-hidden\_discovery.py): Python program that prints all the

  names defined by the compiled module `hidden\_4.pyc`.

  \* Output: One name per line in alphabetical order.

  \* Names starting with `\_\_` are not printed.

\* \*\*5. Everything can be imported\*\*

  \* [5-variable\_load.py](./5-variable\_load.py): Python program that imorts the

  variable `a` from the file [variable\_load\_5.py](./variable\_load\_5.py) and prints its value.

\* \*\*6. Build my own calculator!\*\*

  \* [100-my\_calculator.py](./100-my\_calculator.py): Python program that imports all functions

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## 0-add.py

#!/usr/bin/python3

if \_\_name\_\_ == "\_\_main\_\_":

    """Print the sum of 1 and 2."""

    from add\_0 import add

    a = 1

    b = 2

    print("{} + {} = {}".format(a, b, add(a, b)))

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 1-calculation.py

#!/usr/bin/python3

if \_\_name\_\_ == "\_\_main\_\_":

    """Print the sum, difference, multiple and quotient of 10 and 5."""

    from calculator\_1 import add, sub, mul, div

    a = 10

    b = 5

    print("{} + {} = {}".format(a, b, add(a, b)))

    print("{} - {} = {}".format(a, b, sub(a, b)))

    print("{} \* {} = {}".format(a, b, mul(a, b)))

    print("{} / {} = {}".format(a, b, div(a, b)))

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 100-my\_calculator.py

#!/usr/bin/python3

if \_\_name\_\_ == "\_\_main\_\_":

    """Handle basic arithmetic operations."""

    from calculator\_1 import add, sub, mul, div

    import sys

    if len(sys.argv) - 1 != 3:

        print("Usage: ./100-my\_calculator.py <a> <operator> <b>")

        sys.exit(1)

    ops = {"+": add, "-": sub, "\*": mul, "/": div}

    if sys.argv[2] not in list(ops.keys()):

        print("Unknown operator. Available operators: +, -, \* and /")

        sys.exit(1)

    a = int(sys.argv[1])

    b = int(sys.argv[3])

    print("{} {} {} = {}".format(a, sys.argv[2], b, ops[sys.argv[2]](a, b)))

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 101-easy\_print.py

#!/usr/bin/python3

\_\_import\_\_("os").write(1, "#pythoniscool\n".encode("UTF-8"))

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 102-magic\_calculation.py

#!/usr/bin/python3

def magic\_calculation(a, b):

    """Match bytecode provided by Holberton School."""

    from magic\_calculation\_102 import add, sub

    if a < b:

        c = add(a, b)

        for i in range(4, 6):

            c = add(c, i)

        return (c)

    else:

        return(sub(a, b))

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 103-fast\_alphabet.py

#!/usr/bin/python3

import string

print(string.ascii\_uppercase)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 2-args.py

#!/usr/bin/python3

if \_\_name\_\_ == "\_\_main\_\_":

    """Print the number of and list of arguments."""

    import sys

    count = len(sys.argv) - 1

    if count == 0:

        print("0 arguments.")

    elif count == 1:

        print("1 argument:")

    else:

        print("{} arguments:".format(count))

    for i in range(count):

        print("{}: {}".format(i + 1, sys.argv[i + 1]))

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 3-infinite\_add.py

#!/usr/bin/python3

if \_\_name\_\_ == "\_\_main\_\_":

    """Print the addition of all arguments."""

    import sys

    total = 0

    for i in range(len(sys.argv) - 1):

        total += int(sys.argv[i + 1])

    print("{}".format(total))

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**4-hidden\_discovery.py**

#!/usr/bin/python3

if \_\_name\_\_ == "\_\_main\_\_":

    """Print all names defined by hidden\_4 module."""

    import hidden\_4

    names = dir(hidden\_4)

    for name in names:

        if name[:2] != "\_\_":

            print(name)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 5-variable\_load.py

#!/usr/bin/python3

if \_\_name\_\_ == "\_\_main\_\_":

    """Print the value of variable a from variable\_load\_5."""

    from variable\_load\_5 import a

    print(a)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## calculator\_1.py

#!/usr/bin/python3

def add(a, b):

    """My addition function

    Args:

        a: first integer

        b: second integer

    Returns:

        The return value. a + b

    """

    return (a + b)

def sub(a, b):

    """My subtraction function

    Args:

        a: first integer

        b: second integer

    Returns:

        The return value. a - b

    """

    return (a - b)

def mul(a, b):

    """My multiplication function

    Args:

        a: first integer

        b: second integer

    Returns:

        The return value. a \* b

    """

    return (a \* b)

def div(a, b):

    """My division function

    Args:

        a: first integer

        b: second integer

    Returns:

         The return value. a / b

    """

    return int(a / b)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## variable\_load\_5.py

#!/usr/bin/python3

a = 98

"""Simple variable

"""