LAST LESSON

FUNCTION

- Named block of code
- Accepts arguments
- Can return result

FUNCTION

```
def add_numbers(number_1, number_2):
    result = number_1 + number_2
    return result
```

```
>>> add_numbers(4, 6)
10
>>> add_numbers(1, 2)
3
```

FUNCTION WITHOUT RETURN

```
def add_year(data_dict, year):
    data_dict['year'] = year
```

```
>>> data = {'movie': 'Joker'}
>>> add_year(data, 2019)
>>> data
{'movie': 'Joker', 'year': 2019}
```

FUNCTION WITHOUT ARGUMENTS AND WITHOUT RETURN

```
def print_separator():
    print('-' * 80)
```

MORE COMPLEX FUNCTION

```
def sum_positive(numbers):
    total = 0
    for number in numbers:
        if number > 0:
            total += number
    return total
```

```
>>> sum_positive([1, 0, -5, 4, -10, 3])
8
```

ERROR IN FUNCTION

```
def sum_positive(numbers):
    total = 0
    for number in numbers:
        if number > 0:
            total += number
    return total

number = 42
sum_positive(number)
```

```
$ python3 error_in_function.py
Traceback (most recent call last):
  File "error_in_function.py", line 9, in <module>
    sum_positive(number)
  File "error_in_function.py", line 3, in sum_positive
    for number in numbers:
TypeError: 'int' object is not iterable
```

COLLECTION ASSIGNMENT

```
>>> a, b, c = (1, 2, 3)
>>> a
1
>>> b
2
>>> c
3
```

```
>>> a, *b = (1, 2, 3)
>>> a
1
>>> b
[2, 3]
```

```
>>> a, *b, c = (1, 2, 3, 4, 5)
>>> a
1
>>> b
[2, 3, 4]
>>> c
5
```

IMPORT

```
>>> import random
>>> random.random()
0.6914667260222296
>>> random.randint(10, 20)
17
>>> import string
>>> random.choice(string.ascii_letters)
'Q'
```

https://docs.python.org/3/library/random.html

https://docs.python.org/3/library/string.html

SPECIAL CHARACTERS

- '\n' - newline
- '\t' - tabulator
- '\r' - carriage return

```
>>> print('This is the first line\nAnd this is the second line')
This is the first line
And this is the second line
>>> print('This is just symbol backslash \\')
This is just symbol backslash \
```

LIST COMPREHENSION

```
powers = []
for x in range(10):
        powers.append(x ** 2)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
powers = [x ** 2 for x in range(10)]
```

DICT COMPREHENSION

```
powers = {}
for x in range(10):
        powers[x] = x ** 2
```

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81
```

```
powers = {x: x ** 2 for x in range(10)}
```

TODAY'S LESSON

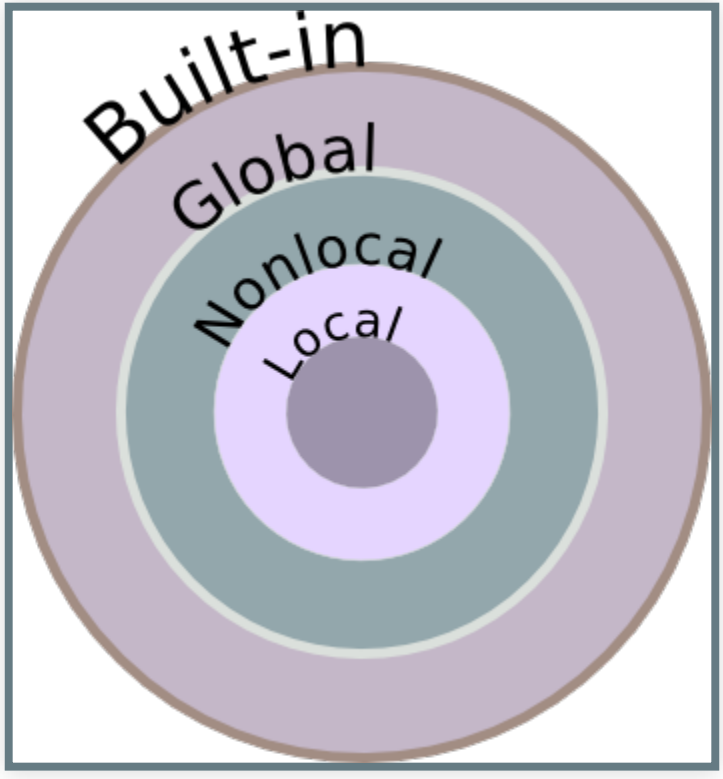
SCOPES

```
>>> name = 'John'
>>> def func():
...     print(name)
>>> func()
John
```

SCOPES

```
>>> def func():
...     name = 'John'
...     print(name)
>>> print(name, 'Smith')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'name' is not defined
```

SCOPES



BUILT-IN SCOPE

- Built-in types
- Built-in functions

```
>>> import pprint
>>> pprint.pprint(__builtins__.__dict__)
{'ArithmeticError': <class 'ArithmeticError'>,
 'AssertionError': <class 'AssertionError'>,
 'AttributeError': <class 'AttributeError'>,
...
 'abs': <built-in function abs>,
 'all': <built-in function all>,
 'any': <built-in function any>,
...
 'tuple': <class 'tuple'>,
 'type': <class 'type'>,
 'vars': <built-in function vars>,
 'zip': <class 'zip'>}
```

GLOBAL

```
name = 'John'

def print_name():
        print(name)

print_name()
```

```
John
```

GLOBAL

```
name = 'John'

def print_name():
        print(name)

print_name()
name = 'Bob'
print_name()
```

```
John
Bob
```

GLOBAL

```
name = 'John'

def print_name():
        name = 'Bob'
        print(name)

print_name()
print(name)
```

```
Bob
John
```

GLOBAL

```
name = 'John'

def print_name():
        name = 'Bob'
        print('Locals:', locals())
        print('Globals:', globals())
        print(name)

print_name()
print(name)
```

```
Locals: {'name': 'Bob'}
Globals {
        '__name__': '__main__',
        '__builtins__': <module 'builtins' (built-in)>,
        '__file__': '/home/psebek/projects/engeto/07/global.py',
        'name': 'John',
        'print_name': <function print_name at 0x7f8751945ef0>
        ...
}
Bob
John
```

GLOBAL

```
name = 'John'

def print_name():
        global name # <---- global keyword
        name = 'Bob'
        print(name)

print_name()
print(name)
```

```
Bob
Bob
```

NESTED FUNCTIONS

```
import random

def wrapper():
    start = random.randint(1, 5)
    end = random.randint(5, 10)
    print('start = ', start, ', end = ', end)
    def inner():
        return random.randint(start, end)
    return inner

random_limits = wrapper()
print('#' * 20)
print(random_limits())
print('-' * 20)
print(random_limits())
```

```
start =  3, end = 8
####################
6
--------------------
3
```

NESTED FUNCTIONS

```
import random

def wrapper():
    start = random.randint(1, 5)
    end = random.randint(5, 10)
    print('start = ', start)
    print('end = ', end)

    def inner():
        nonlocal start
        nonlocal end
        start = end = 12
        return random.randint(start, end)

    return inner
```

```
start =  2
end =  5
####################
12
--------------------
12
```

FUNCTION ARGUMENTS

```
>>> def function(argument_1, argument_2):
...     print(argument_1, argument_2)
>>> function(1, 2)
1 2
```

KEYWORD ARGUMENTS

```
>>> def function(argument, default_argument = 2):
...     print(argument, default_argument)
```

```
>>> function(1)
1 2
```

```
>>> function(1, 3)
1 3
```

```
>>> function(1, default_argument = 4)
1 4
```

```
>>> function(argument = 2, default_argument = 5)
2 5
```

```
>>> function(default_argument = 5, argument = 3)
3 5
```

VARIABLE NUMBER OF ARGUMENTS *

```
>>> def function(*args):
...     print(args)
```

```
>>> function(1, 2, 3)
(1, 2, 3)
```

```
>>> function()
()
```

```
>>> l = [1, 2, 3]
>>> function(*l)
(1, 2, 3)
```

VARIABLE NUMBER OF KEYWORD ARGUMENTS **

```
>>> def function(**kwargs):
...     print(kwargs)
```

```
>>> function(a = 1, b = 2)
{'a': 1, 'b': 2}
```

```
>>> function()
{}
```

```
>>> arguments = {'c': 3, 'd': 4}
>>> function(**arguments)
{'c': 3, 'd': 4}
```

ULTIMATE FORM OF ARGUMENTS

```
>>> def function(a, b, *args, default = None, **kwargs):
...     print(a, b, args, default, kwargs)
```

```
>>> function(1, 2, 3, 4, 5, key = 'KEY')
1 2 (3, 4, 5) None {'key': 'KEY'}
```

NEVER USE MUTABLES AS DEFAULTS

```
>>> def append(item, l = []):
...     l.append(item)
...     return l

>>> i = append(1)
>>> print(i)
[1]
>>> j = append(2)
>>> print(j)
[1, 2]
>>> print(i)
[1, 2]
```

NEVER USE MUTABLES AS DEFAULTS

```
>>> def append(item, l = None):
...     if l is None:
...         l = []
...     l.append(item)
...     return l

>>> i = append(1)
>>> print(i)
[1]
>>> j = append(2)
>>> print(j)
[2]
>>> print(i)
[1]
```

EXERCISES

MAXIMUM

- Variable number of inputs
- Returns maximum item, if empty sequence return None

```
>>> maximum(1, 2, 3, 4)
4
>>> maximum()
None
```

MAXIMUM WITH DEFAULT

- Variable number of inputs, default keyword argument
- Returns maximum item, if empty sequence return default

```
>>> maximum(1, 2, 3, 4, default = -1)
4
>>> maximum()
None
>>> maximum(default = -1)
-1
```

MAXIMUM WITH DEFAULT AND KEY

- Variable number of inputs, keyword argument key
- Returns maximum item according to key function if provided

```
>>> maximum('Aš', 'Brno', 'Pardubice', 'Praha')
'Praha'
>>> maximum('Aš', 'Brno', 'Pardubice', 'Praha', key = len)
'Pardubice'
```

END