Built-in Functions

```
# all(iterable)
# - returns True if all items in an iterabl is true
# iterable can be: list, tuple, dictionary
x = [1, 2, 3]
y = [1, 2, 3, False]
print(all(x))       # True
print(all(y))       # False


# bin()
# convert the numbers to binary
print(bin(120))     # 0b1111000


# sum(iterable, start)
# - used to sum elements in the iterable
# start: this will be added to the iterable
numbers = [1, 2, 3]
print(sum(numbers, 10))     # 16
print(sum(numbers, 14))     # 20


# round(num, num_of_digits)
# - used to round number to decimal, get the nearest integer
# - returnin floating point number that is rounded version of the specific number
# - the default number of decimal is 0
print(round(10.2255, 2))        # 10.23
print(round(10.53, 1))          # 10.5


# range(start, end, step)
# - returns a sequence of numbers, starting from zero by default, and increment by 1
by default
# - Note That: end is not included on the range
# - start: optional, an integer numer specifying at which position to start
# - end: required, an integer number specifying at which position to stop
# - optional, an integer number specifying the incrementation, default is 1
x = range(10, 20)
for num in x:
    print(num)

print(list(range(10)))  # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]


# print(obj, sep, end, file, flush)
# - print function prints the specified message to the screen, or other standard
output device
# - the message can be a string, or any other object
# - the object will be converted into string before written to the screen
# - object: any object, that will be converted to string before printed
# - sep: optional, specify how to separate the objects, if there is more one,
default is ""
# - end: optional, specify what to print at the end, default is line feed "\n"
# - file: optional, an object wiht a write method, default is sys.stdout
```

```python
# - flush: optional, a boolean , specifying if the output is flushed (true) or
buffered (false), default is false

print("Hello", "how are you?")      # Hello how are you?
x = ("apple", "banana", "cherry")
print(x)        # ('apple', 'banana', 'cherry')
print(type(x))  # <class 'tuple'>
print("Hello", "how are you?", sep="---")    #Hello---how are you?


# abs()
# - this get the absolute value of teh number
# - the distance between your number and zero
# - this convert number to positive
print(abs(-10))     # 10
print(abs(-10.5))   # 10.5


# pow(base, exp, mod)
# - get the power of given number
# - if the mod is written, the function will return the mode of the powered number
print(pow(10, 2))   # 100
print(pow(5, 2))    # 25
print(pow(2,2,2))   # 0


# min()
# - used to get the minimum value
print(min(10, 9))       # 9


# max()
# - used to get the maximum value
print(max(10, 9))       # 10


# slice(start, end, step)
# - return a slice object
# - start: refers to position to start the slicing, default is 0
# - end: refers to the end of slicing
# - step: refers to the step of slicing, defaul is 1
a = ("a", "b", "c", "d", "e", "f", "g", "h")
x = slice(3, 5)
print(a[x])     # ('d', 'e')

x = slice(0, 8, 3)      # ('a', 'd', 'g')
print(a[x])


# map()
# - this take function and iterable
# - map called map because it map the function on every element
# - the function can be pre-defined or lambda function

def formatText(text):
    return f"- {text.title()} "

myText = ['OSama', "AHMED", "sAYed"]
```

```python
myFormatedText = map(formatText, myText)

for name in myFormatedText:
    print(name)

# another way
for name in map(formatText, myText):
    print(name)

# use map with pre-defined function
myFriends = ['OSama', "AHMED", "sAYed"]
for name in map(lambda text: f"-> {text.title()}", myFriends):
    print(name)



# filter()
# - take a function and iterable
# - filter run a function on every element on the iterable
# - the function can be pre-defined function or lambda function
# - filter out all elements for switch the function return True
# - the function needs to return boolean value
# - usually used to test the iterable if them is accepted if true of not

ages = [5, 15, 12, 30, 24, 32]

def func(x):
    if x < 18:
        return False
    else:
        return True

adults = filter(func, ages)
for x in adults:
    print(x, end=", ")  # 30, 24, 32,


numbers = [10, 20, 5, 2, 6]
def less_10(num):
    if num < 10:
        return True
    else:
        return False
x = filter(less_10, numbers)
for value in x:
    print(value, end=" - ")     # 5 - 2 - 6 -



# reduce()
# - take function and iterable
# - reduce run a function on first and second element and give the result
# then run function on result and third number and get result
# then run function on result and fourth element and get the result
# - till one element is left and this is the is result of the reduce
# - the function can be lambda function or pre-defined function
# - to use reduce funcrion, you should import the module [ from functools import
reduce ]
```

```python
from functools import reduce
def sumALl(num1, num2):
    return num1 + num2

numbers = [1,23,4,90,6,7,8]
result = reduce(sumALl, numbers)
print(result)       # 139

# enumerate(iterable, start)
# - this add counter to the iterable while making loop
mySkills = ["HTML", "CSS", "JS"]
mySkillsWithCounter = enumerate(mySkills, 1)

for c, s in mySkillsWithCounter:
    print(f" {c}- {s}")

# ============
# == Output ==
# ============
# 1- HTML
# 2- CSS
# 3- JS

# help()
# - used to help you to get the manual of the function
print(help(print))

# ============
# == Output ==
# ============
# Help on built-in function print in module builtins:

# print(...)
#    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

#     Prints the values to a stream, or to sys.stdout by default.
#     Optional keyword arguments:
#     file:  a file-like object (stream); defaults to the current sys.stdout.
#     sep:   string inserted between values, default a space.
#     end:   string appended after the last value, default a newline.
#     flush: whether to forcibly flush the stream.



# reversed()
# - function returns a reversed iterator object.
string = 'Osama'
str = reversed(string)
for letter in str:
    print(letter, end = " " )   # a m a s O
```