Day 10

Python for Data Science
Built-in Functions

Python has several functions that are readily available for use. These functions are called built-in functions https://www.w3schools.com/python/python_ref_functions.asp

A	B	C	D	E
abs()	bin()	callable()	delattr()	enumerate()
all()	bool()	chr()	dict()	eval()
any()	breakpoint()	classmethod()	dir()	exec()
ascii()	bytearray()	compile()	divmod()	
	bytes()	complex()		

F	G	H	I	L
filter()	getattr()	hasattr()	id()	len()
float()	globals()	hash()	input()	list()
format()		help()	int()	locals()
frozenset()		hex()	isinstance()	
			issubclass()	
			iter()	

M	N	O	P	R
map()	next()	object()	pow()	range()
max()		oct()	print()	repr()
memoryview()		open()	property()	reversed()
min()		ord()		round()

S	T	V	Z	-
set()	tuple()	vars()	zip()	_import_()
setattr()	type()			
slice()				
sorted()				
staticmethod()				
str()				
sum()				
super()				

abs()

Returns the absolute value of a number

```
num1 = int(input('Enter a number: '))
print('The entered number is', num1)
num2 = float(input('Enter a number: '))
print('The entered number is', num2)
print('The absolute value of the first number is', abs(num1))
print('The absolute number of the second number is', abs(num2))
print('The difference between the two numbers is', abs(num1-num2))
```

```
Enter a number: -2
The entered number is -2
Enter a number: -8
The entered number is -8.0
The absolute value of the first number is 2
The absolute number of the second number is 8.0
The difference between the two numbers is 6.0
```

all()

Returns True if all elements in passes iterable are true. When the iterable object is empty, it returns True. Here,0 and False return False in this function

```
nlis1 = [0.577, 1.618, 2.718, 3.14, 6, 28, 37, 1729]
print(all(nlis1))
nlis1.append(0) # Add '0' to the end of the list
print(nlis1)
print(all(nlis1))
nlis1.append(False) # Adds 'False' to the end of the list
print(nlis1)
print(all(nlis1))
nlis1.clear() # It returns an emtpy list
print(nlis1)
print(all(nlis1))
```

```
True
[0.577, 1.618, 2.718, 3.14, 6, 28, 37, 1729, 0]
False
[0.577, 1.618, 2.718, 3.14, 6, 28, 37, 1729, 0, False]
False
[]
True
```

bin()

Returns the binary representation of a specificied integer

```
num = int(input('Enter a number: '))
print(f'The entered number is {num}.')
print(f'The binary representation of {num} is {bin(num)}.')
```

```
Enter a number: 2
The entered number is 2.
The binary representation of 2 is 0b10.
```

bool()

Converts a value to boolean, namely True and False

```
lis, dict, tuple = [], {}, ()
print(bool(lis), bool(dict), bool(tuple))
lis, dict, tuple = [0], {'a': 1}, (1,)
print(bool(lis), bool(dict), bool(tuple))
lis, dict, tuple = [0.0], {'a': 1.0}, (1.0,)
print(bool(lis), bool(dict), bool(tuple))
a, b, c = 0, 3.14, 'Hello, Python!'
print(bool(a), bool(b), bool(c))
statement = None
print(bool(None))
true = True
print(bool(true))
```

```
False False False
True True True
True True True
False True True
False
True
```

bytes()

Returns a btyes object

```
msg = 'Hello, Python!'
new_msg = bytes(msg, 'utf-8')
print(new_msg)
```

```
b'Hello, Python!'
```

callable()

Checks and returns True if the object passed appears to be callable

```
var = 3.14
print(callable(var)) # since the object does not appear callable,
it returns False
def function(): # since the object appears callable, it returns
True
  print('Hi, Python!')
msg = function
print(callable(msg))
```

```
False
True
```

chr()

It returns a character from the specified Unicode code.

```
print(chr(66))
print(chr(89))
print(chr(132))
print(chr(1500))
print(chr(3))
print(chr(-500)) # The argument must be inside of the range.
```

```
B
Y
?
Ɔ
?

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [8], in <cell line: 6>()
      4 print(chr(1500))
      5 print(chr(3))
----> 6 print(chr(-500))

ValueError: chr() arg not in range(0x110000)
```

```
print(chr('Python')) # The argument maut be integer.
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [9], in <cell line: 1>()
----> 1 print(chr('Python'))

TypeError: an integer is required (got type str)
```

enumerate()

It takes a collection (e.g. a tuple) and returns it as an enumerate object.

```
str_list = ['Hello Python!','Hello, World!']
for i, str_list in enumerate(str_list):
  print(i, str_list)
```

0 Hello Python!
1 Hello, World!

```
str_list = ['Hello Python!','Hello, World!']
enumerate_list = enumerate(str_list)
print(list(enumerate_list))
```

[(0, 'Hello Python!'), (1, 'Hello, World!')]

filter()

It excludes items in an iterable object.

```
def filtering(data):
  if data > 30:
    return data
data = [0.577, 1.618, 2.718, 3.14, 6, 28, 37, 1729]
result = filter(filtering, data)
print(list(result))
```

0 Hello Python!
1 Hello, World!

eval()

This function evaluates and executes an expression.

```
num = int(input('Enter a number: '))
print(f'The entered number is {num}.')
print(eval('num*num'))
```

Enter a number: 2
The entered number is 2.
4

map()

It returns the specified iterator with the specified function applied to each item.

```
special_nums = [0.577, 1.618, 2.718, 3.14, 28, 37, 1729]
def division(number):
  return number/number
division_number_iterator = map(division, special_nums)
divided_nums = list(division_number_iterator)
print(divided_nums)
```

<map object at 0x0000028AFA2641F0>
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]

open()

It opens a file and returns a file object.

```
path = "authors.txt"
file = open(path, mode = 'r', encoding='utf-8')
print(file.name)
print(file.read())
```

authors.txt
English,Charles Severance
English,Sue Blumenberg
English,Elloitt Hauser
Spanish,Fernando TardÃo MuÃ±iz

```
# Open file using with
path = "authors.txt"
with open(path, "r") as file:
  FileContent = file.read()
  print(FileContent)
```

English,Charles Severance
English,Sue Blumenberg
English,Elloitt Hauser
Spanish,Fernando TardÃƒo MuÃƒÂ±iz

complex()

It returns a complex number.

```
print(complex(1))
print(complex(2, 2))
print(complex(3.14, 1.618))
```

```
(1+0j)
(2+2j)
(3.14+1.618j)
```