Built-In Functions

What are Built-In Functions...?

- A **function** that is **built** into an application and can be accessed by end-users.

- For example, most spreadsheet applications support a **built**-in SUM **function** that adds up all cells in a row or column.

abs()

The `abs()` function returns the absolute value of the specified number.

```
>>> abs(-7)
7
>>>
>>> abs(7)
7
>>>
>>> abs(0)
0
>>> x = abs(3+5j)
>>> print(x)
5.830951894845301
>>>
```

all()

- The all() function returns True if all items in an iterable are true, otherwise it returns False.
- If the iterable object is empty, the all() function also returns True.

```
>>>
>>>
>>>
>>> all({'*','',''})
False
>>> all([' ',' ',' '])
True
>>> mylist = [0, 1, 1]
>>> x = all(mylist)
>>> print(x)
False
>>>
```

any()

- The any() function returns True if any item in an iterable are true, otherwise it returns False.
- If the iterable object is empty, the any() function will return False.

```
>>>
>>>
>>>
>>>
>>> any((1,0,0))
True
>>> any((0,0,0))
False
>>> mydict = {0 : "Apple", 1 : "Orange"}
>>> x = any(mydict)
>>> print(x)
True
>>> # Returns True because the second key is True.
...
>>> # For dictionaries the any() function checks the keys, not the values.
```

ascii()

GKTCS INNOVATIONS

- The ascii() function returns a readable version of any object (Strings, Tuples, Lists, etc).
- The ascii() function will replace any non-ascii characters with escape characters: å will be replaced with \xe5.

```
>>> print(x)
'My name is St\xe5le'
>>>
>>>
>>>
>>> x = ascii("My name is Ståle")
>>> print(x)
'My name is St\xe5le'
>>>
```

bin()

- bin() converts an integer to a binary string.

```
>>>
>>>
>>>
>>> bin(7)
'0b111'
>>>
```

bool()

- The bool() function returns the Boolean value of a specified object.
- The object will always return True, unless:
- The object is empty, like [], (), {}
- The object is False
- The object is 0
- The object is None

```
>>>
>>>
>>>
>>> bin(7)
'0b111'
>>>
>>>
>>> x = bool(1)
>>> print(x)
True
>>>
```

GKTCS INNOVATIONS

bytearray()

- bytearray() returns a python array of a given byte size.

```
x = bytearray(4)
print(x)
array(b'\x00\x00\x00\x00')
```

bytes()

- bytes() returns an immutable bytes object.

```
>>>
>>>
>>>
>>> x = bytearray(4)
>>> print(x)
bytearray(b'\x00\x00\x00\x00')
>>>
>>>
>>>
>>> x = bytes(4)
>>> print(x)
b'\x00\x00\x00\x00'
>>>
```

callable()

- callable() tells us if an object can be called.

```
>>>
>>>
>>>
>>> def x():
...     a = 5
...
>>> print(callable(x))
True
>>>
```

chr()

- chr() function returns the character that represents the specified Unicode.

```
True
>>>
>>>
>>>
>>> x = chr(97)
>>>
>>> print(x)
a
>>>
```

compile()

- compile() returns a Python code object. We use Python in built function to convert a string code into object code.

```
>>>
>>>
>>>
>>> x = chr(97)
>>>
>>> print(x)
a
>>>
>>>
>>> x = compile('print(55)', 'test', 'eval')
>>> exec(x)
55
>>>
```

delattr()

- delattr() function will delete the specified attribute from the specified object.

```
>>>
>>>
>>> class Person:
...   name = "John"
...   age = 36
...   country = "Norway"
...
>>> delattr(Person, 'age')
>>>
>>> # The Person object will no longer contain an "age" property
...
>>>
>>>
>>>
```

dict()

- The dict() function creates a dictionary.

- A dictionary is a collection which is unordered, changeable and

```
>>>
>>>
>>>
>>> x = dict(name = "John", age = 36, country = "Norway")
>>>
>>> print(x)
{'name': 'John', 'age': 36, 'country': 'Norway'}
>>>
>>>
>>>
>>>
>>>
>>>
```

dir()

- dir() returns an object’s attributes.

```
>>>
>>> class Person:
...   name = "John"
...   age = 36
...   country = "Norway"
...
>>> print(dir(Person))
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getatt
ribute__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne_
_', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subcl
asshook__', '__weakref__', 'age', 'country', 'name']
>>>
>>>
>>>
>>>
>>>
>>>
>>>
```

divmod()

- divmod() in Python built-in functions, takes two parameters, and returns a tuple of their quotient and remainder.

- In other words, it returns the floor division and the modulus of the two numbers.

```
>>>
>>>
>>>
>>> x = divmod(5, 2)
>>>
>>> print(x)
(2, 1)
>>>
>>>
```

enumerate()

- This Python Built In function returns an enumerate object.
- In other words, it adds a counter to the inerrable.

```
>>>
>>> x = divmod(5, 2)
>>>
>>> print(x)
(2, 1)
>>>
>>>
>>>
>>> x = ('apple', 'banana', 'cherry')
>>> y = enumerate(x)
>>>
>>> print(list(y))
[(0, 'apple'), (1, 'banana'), (2, 'cherry')]
>>>
>>>
```

eval()

- This Function takes a string as an argument, which is parsed as an expression.

```
>>>
>>>
>>> x = 'print(55)'
>>> eval(x)
55
>>>
```

filter()

- filter() filters out the items for which the condition is True.

```
>>>
>>>
>>> ages = [5, 12, 17, 18, 24, 32]
>>>
>>> def myFunc(x):
...   if x < 18:
...     return False
...   else:
...     return True
...
>>> adults = filter(myFunc, ages)
>>>
>>> for x in adults:
...   print(x)
...
18
24
32
```

exec()

GKTCS INNOVATIONS

- exec() runs Python code dynamically.

```
32
>>>
>>>
>>> x = 'name = "John"\nprint(name)'
>>> exec(x)
John
>>>
>>>
>>>
```

float()

- This Python Built In function converts an int or a compatible value into a float.

```
>>>
>>> x = float(3)
>>>
>>> print(x)
3.0
>>>
>>>
>>>
```

input()

- The input() function allows user input.

```
3.0
>>>
>>>
>>> print("Enter your name:")
Enter your name:
>>> x = input()
print("Hello, " + x)
>>>
>>>
```

len()

- The len() function returns the number of items in an object.

- When the object is a string, the len() function returns the number of characters in the string.

```
>>>
>>>
>>>
>>>
>>> mylist = ["apple", "orange", "cherry"]
>>>
>>> x = len(mylist)
>>>
>>> print(x)
3
>>>
>>>
```

max()

- The max() function returns the item with the highest value, or the item with the highest value in an iterable.

- If the values are strings, an alphabetically comparison is done.

```
>>>
>>>
>>>
>>> x = max(5, 10)
>>>
>>> print(x)
10
>>>
```