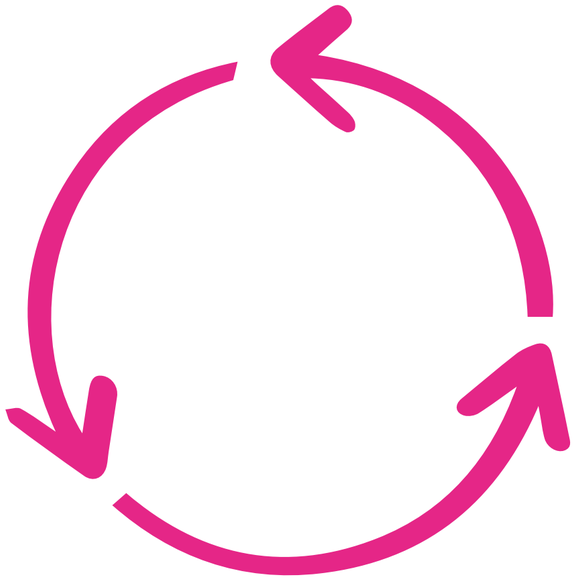
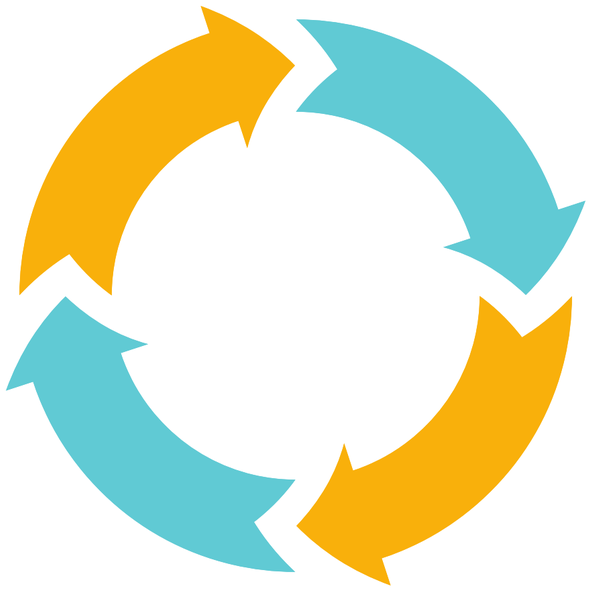
Recursion



Recursion

A recursive function is a function that calls **itself** during its execution



Code Example

```
def factorial(n):
    # Base case: factorial of 0 or 1 is 1
    if n == 0 or n == 1:
        return 1
    else:
        # Recursive case: multiply n with
        # factorial of (n-1)
        return n * factorial(n - 1)
```

factorial(4):

return 4 * factorial(3)

24

6

factorial(3):

return 3 * factorial(2)

2

factorial(2):

return 2 * factorial(1)

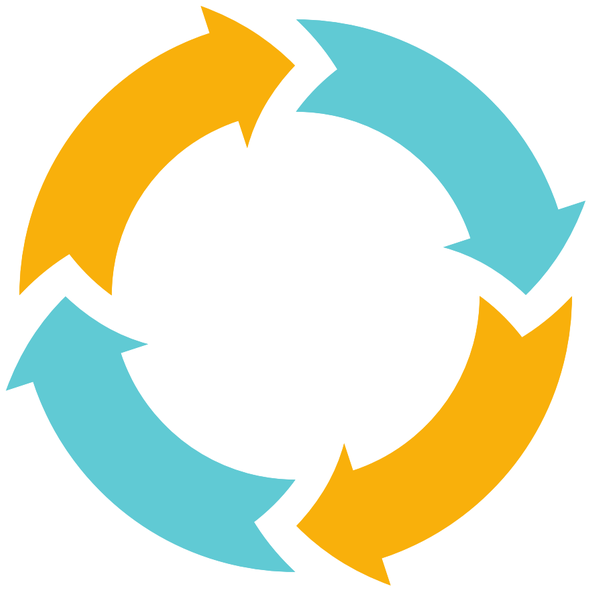
1

return 1

Breaking Example

*and ** Operator

* and ** operators are widely used in Python function definitions and calls for packing and unpacking arguments

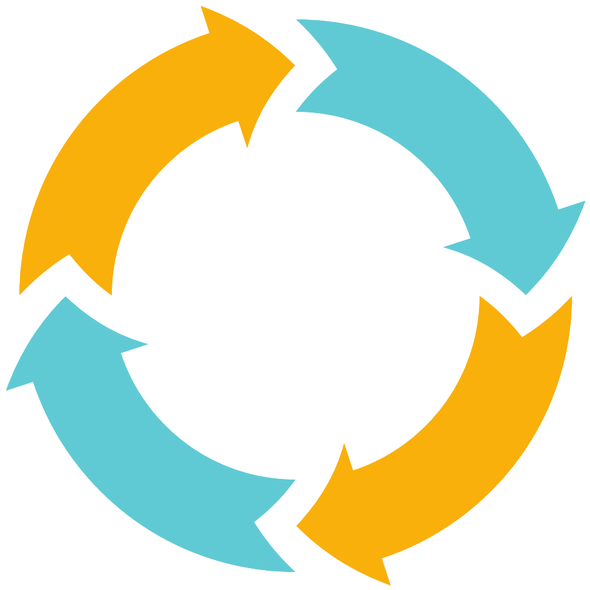


Code Example

```
# * in function definitions
def func(*args):
    for arg in args:
        print(arg)

func(1, 2, 3, 4)
 # prints 1, 2, 3, 4 each on a new line
```

*args allows the function to accept any number of positional arguments
args within the function is a tuple of the provided positional arguments

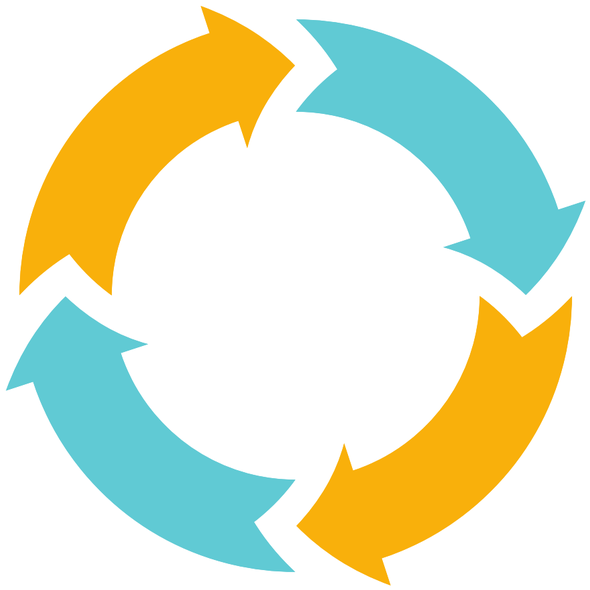


Code Example

```
# ** in function definitions
def func(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")

func(a=1, b=2)  # prints "a: 1" and "b: 2"
each on a new line
```

**kwargs allows the function to accept any number of keyword arguments
kwargs within the function is a dictionary

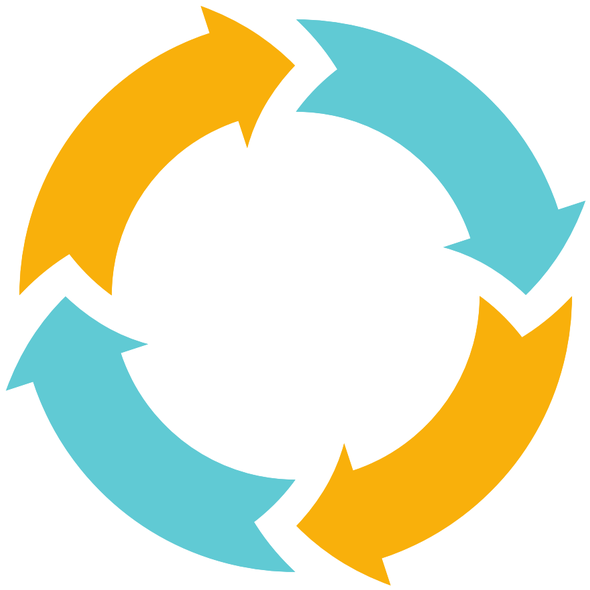


Code Example

```
# * in function calls
def func(a, b, c):
    print(a, b, c)

args = [1, 2, 3]
func(*args)  # prints "1 2 3"
```

When calling a function, * can be used to unpack an iterable into positional arguments



Code Example

```
# ** in function calls
def func(a, b, c):
    print(a, b, c)

kwargs = {'a': 1, 'b': 2, 'c': 3}
func(**kwargs)  # prints "1 2 3"
```

When calling a function, ** can be used to unpack a dictionary into keyword arguments

String Syntax

String is immutable

str_identifier = ' ' or " "

str_identifier = str(var)

any valid identifier in Python

Supports indexing

#immutable

Any operation that seems to modify a string actually creates a new one

```
greeting1 = 'Hello, World! '
greeting2 = "Hello, World!"
greeting3 = """
Hello, World
!
"""
```

#Indexing

```
first_char =
my_string[0]
```

Multiline strings can be initialized with the help of triple quotes

#Concatenation

```
s1 = "Hello"
s2 = "World"
s3 = s1 + s2
```

```
s1 = 12
s2 = "digits"
s3 = str(s1) + s2
```

```
s1 = "Hello"
s2 = 90
s3 = s1 + s2
```

Common Operations

```
s = "Hello " * 3
# Output "Hello Hello Hello "
```

#replication

```
s = "Hello, World!"
print(s[7:12])
 # Output "World"
```

#slicing

Common Operations

```
s = "Hello, World!"
print(len(s))
# Output 13
```

#len() function

```
s = "Hello, World!"
print("World" in s)
 # Output True
```

#membership test

Capitalize()

```
# Converts the first character of a string to uppercase and the rest to lowercase

string = "hello world"
capitalized_string = string.capitalize()
print(capitalized_string)
# Output: "Hello world"
```

casefold()

```
# Returns a lowercase version of the string, suitable for case-insensitive comparisons

string = "Hello World"
casefolded_string = string.casefold()
print(casefolded_string)
# Output: "hello world"
```

center(width[, fillchar])

```
# Returns a centered string by padding it with a specified character on both sides

string = "hello"
centered_string = string.center(10, "-")
print(centered_string)
# Output: "--hello---"
```

count(sub[, start[, end]])

```
# count(sub[, start[, end]])
# Returns the number of non-overlapping occurrences of a substring in a string

string = "hello world"
count = string.count("o")
print(count)  # Output: 2
```

endswith(suffix[, start[, end]])

```
# Checks if a string ends with a specified suffix and returns a boolean

string = "hello world"
ends_with_world = string.endswith("world")
print(ends_with_world)
# Output: True
```

find(sub[, start[, end]])

```
# Returns the lowest index of a substring in a string, or -1 if not found

string = "hello world"
index = string.find("world")
print(index)
# Output: 6
```

format(*args, **kwargs)

```
# # Formats a string by replacing placeholders with values from arguments or keyword arguments

name = "Alice"
age = 25
formatted_string = "My name is {} and I'm {} years old.".format(name, age)
print(formatted_string)  # Output: "My name is Alice and I'm 25 years old."
```

format_map(mapping)

```
# # Formats a string by replacing placeholders with values from a mapping object

person = {"name": "Bob", "age": 30}
formatted_string = "My name is {name} and I'm {age} years old.".format_map(person)
print(formatted_string)

# Output: "My name is Bob and I'm 30 years old."
```

index(sub)

```
# Returns the lowest index of a substring in a string, or raises a ValueError if not found

string = "hello world"
index = string.index("world")
print(index)

# Output: 6
```

isalnum()

```
# Checks if a string consists only of alphanumeric characters and returns a boolean

string = "hello123"
is_alnum = string.isalnum()
print(is_alnum)

# Output: True
```

isalpha()

```
# Checks if a string consists only of alphabetic characters and returns a boolean

string = "hello"
is_alpha = string.isalpha()
print(is_alpha)

# Output: True
```

isascii()

```
# Checks if a string consists only of ASCII characters and returns a boolean

string = "hello"
is_ascii = string.isascii()
print(is_ascii)


# Output: True
```

isdecimal()

```
# Checks if a string consists only of
decimal characters and returns a boolean

string = "123"
is_decimal = string.isdecimal()
print(is_decimal)

# Output: True
```

islower()

```
# Checks if all characters in a string are lowercase and returns a boolean

string = "hello"
is_lower = string.islower()
print(is_lower)


# Output: True
```

isspace()

```
# Checks if a string consists only of whitespace characters and returns a boolean

string = "   "
is_space = string.isspace()
print(is_space)

# Output: True
```

join(iterable)

```
# Concatenates elements of an iterable (e.g., list) into a single string, using the string as a separator

words = ["Hello", "world"]
joined_string = " ".join(words)
print(joined_string)

# Output: "Hello world"
```

lower()

```
# Converts all characters in a string to lowercase

string = "Hello"
lowercase_string = string.lower()
print(lowercase_string)

# Output: "hello"
```

replace(old, new)

```
# Replaces all occurrences of a substring
with another substring

string = "hello world"
replaced_string = string.replace("world",
"everyone")
print(replaced_string)

# Output: "hello everyone"
```

split(sep=None)

```
# Splits a string into a list of substrings using a specified separator

string = "hello world"
split_list = string.split(" ")
print(split_list)


# Output: ["hello", "world"]
```

startswith(prefix)

```
# Checks if a string starts with a specified prefix and returns a boolean

string = "hello world"
starts_with_hello = string.startswith("hello")
print(starts_with_hello)


# Output: True
```

strip()

```
# Removes leading and trailing spaces rs,
optional) from a string

string = "   hello   "
stripped_string = string.strip()
print(stripped_string)


# Output: "hello"
```

swapcase()

```
# Swaps the case of each character in a string (e.g., lowercase to uppercase and vice versa)

string = "Hello World"
swapped_case_string = string.swapcase()
print(swapped_case_string)

# Output: "hELLO wORLD"
```