Python Strings

Strings

- Strings in python are surrounded by either single quotation marks, or double quotation marks.

- 'hello' is the same as "hello".

```
print("Hello")
print('Hello')
```

Strings

- Strings in python are surrounded by either single quotation marks, or double quotation marks.

- 'hello' is the same as "hello".

```
print("Hello")
print('Hello')
```

Assignment not Supported

```
name="Hello"
name[0]="B"
```

```
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    name[0]="B"
TypeError: 'str' object does not support item assignment
```

Traversing a String

Traversing refers to iterating through the elements of a string. one character at a time. To traverse through a string you can write a loop like

```
name="Hello"
for ch in name:
    print(ch , " ~ ",end=" ")
```

```
H ~ e ~ l ~ l ~ o ~
```

String Operators in Python

- Assignment operator: “=.”
- Concatenate operator: “+.”
- String repetition operator: “*.”
- String slicing operator: “[]”
- String comparison operator: “==” & “!=”
- Membership operator: “in” & “not in”

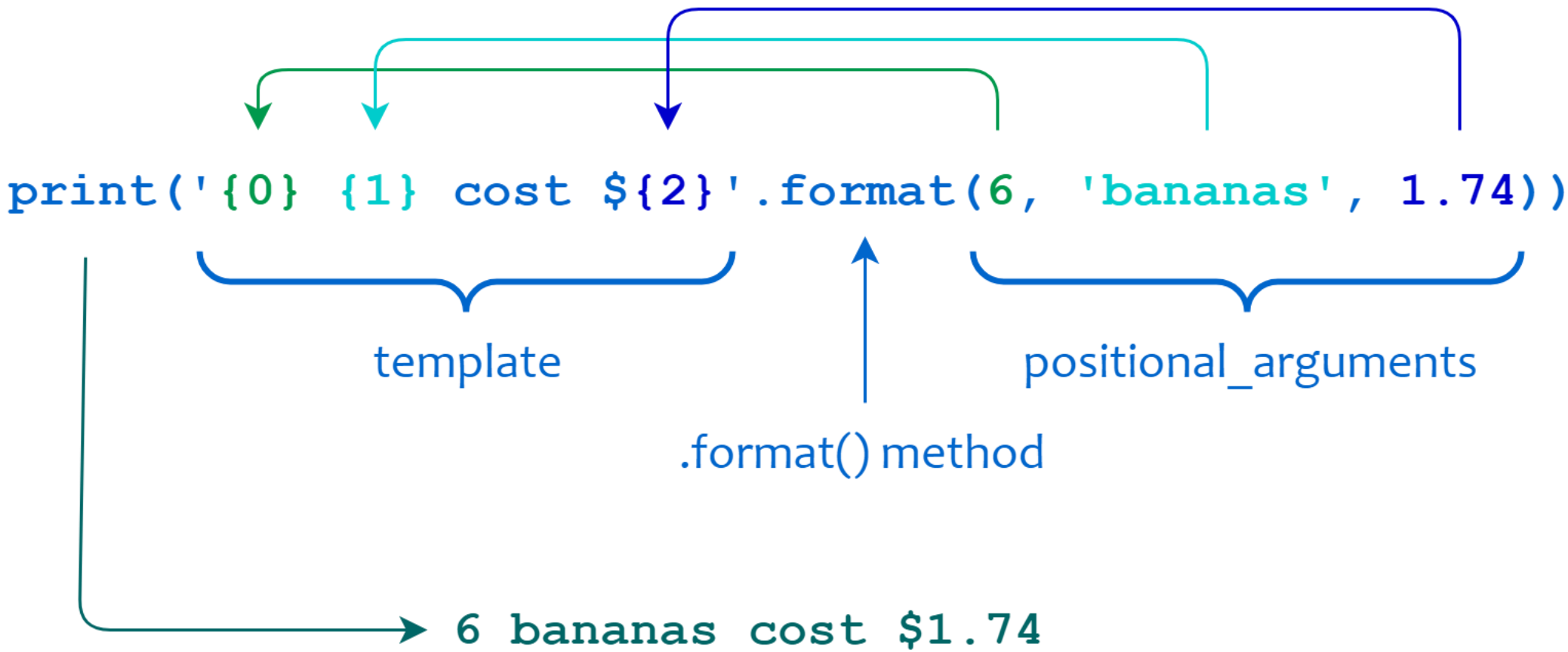
```
a=’Hello’
print(a)
>>> 'hello'+'anand'
'helloanand'
>>> 'anand'*3
'anandanandanand'
>>> 'anand'=='Anand'
False
>>> 'A' in 'anand'
False
```

- Escape sequence operator: “\.”

```
txt = "We are the so-called \"Vikings\" from the north."
print(txt)
```

- String formatting operator: “%” & “{}”

```
print('{0} {1} cost ${2}'.format(6, 'bananas', 1.74))
6 bananas cost $1.74
```


print('{0} {1} cost ${2}'.format(6, 'bananas', 1.74))
template
positional_arguments
.format() method
6 bananas cost $1.74

Slicing String

```
b = "Hello, World!"
print(b[2:5])
#llo
```

```
b = "Hello, World!"
print(b[:5])
#Hello
```

```
b = "Hello, World!"
print(b[2:])
#llo, World!
```

```
b = "Hello, World!"
print(b[-6:-2])
#Worl
```

```
b = "Hello, World!"
print(b[:5])
#Hello
```

```
b = "Hello, World!"
print(b[2:])
#llo, World!
```

Python String Function

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found

isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isascii()	Returns True if all characters in the string are ascii characters
isdecimal()	Returns True if all characters in the string are decimals
isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isprintable()	Returns True if all characters in the string are printable
isspace()	Returns True if all characters in the string are whitespaces

istitle()	Returns True if the string follows the rules of a title
isupper()	Returns True if all characters in the string are upper case
join()	Converts the elements of an iterable into a string
ljust()	Returns a left justified version of the string
lower()	Converts a string into lower case
lstrip()	Returns a left trim version of the string
maketrans()	Returns a translation table to be used in translations
partition()	Returns a tuple where the string is parted into three parts
replace()	Returns a string where a specified value is replaced with a specified value
rfind()	Searches the string for a specified value and returns the last position of where it was found
rindex()	Searches the string for a specified value and returns the last position of where it was found
rjust()	Returns a right justified version of the string

rjust()	Returns a right justified version of the string
rpartition()	Returns a tuple where the string is parted into three parts
rsplit()	Splits the string at the specified separator, and returns a list
rstrip()	Returns a right trim version of the string
split()	Splits the string at the specified separator, and returns a list
splitlines()	Splits the string at line breaks and returns a list
startswith()	Returns true if the string starts with the specified value
strip()	Returns a trimmed version of the string
swapcase()	Swaps cases, lower case becomes upper case and vice versa
title()	Converts the first character of each word to upper case
translate()	Returns a translated string
upper()	Converts a string into upper case
zfill()	Fills the string with a specified number of 0 values at the beginning

List

Lists are used to store multiple items in a single variable.

```
thislist = ["apple", "banana", "cherry"]
print(thislist)

#['apple', 'banana', 'cherry']
```

Traversing a String

Traversing refers to iterating through the elements of a string. one character at a time. To traverse through a string you can write a loop like

```
thislist = ["apple", "banana", "cherry"]
for ch in name:
    print(ch)
```

```
apple
banana
cherry
```

Joining List and RepeatingList

```
l=[1,2,3]
l2=[4,5,6]
print(l+l2)
#[1, 2, 3, 4, 5, 6]

print(l*3)
#[1,2,3,1,2,3,1,2,3]
```

Slicing List

```
b =[1,2,3,4,5,6,7]
print(b[2:5])
print(b[:5])
print(b[2:])
print(b[-6:-2])
print(b[:5])
print(b[2:])
```

```
[3, 4, 5]
[1, 2, 3, 4, 5]
[3, 4, 5, 6, 7]
[2, 3, 4, 5]
[1, 2, 3, 4, 5]
[3, 4, 5, 6, 7]
```