Python Identifiers

An identifier is a name given to entities like class, functions, variables, etc.
It helps to differentiate one entity from another.

Multi-line statement

In Python, end of a statement is marked by a newline character.
But we can make a statement extend over multiple lines with the
line continuation character (\). For example:

```
a = 1 + 2 + 3 + \
4 + 5 + 6 + \
7 + 8 + 9
```

Python Comments

Comments are very important while writing a program. It describes what's going
on inside a program so that a person looking at the source code does not have a
hard time figuring it out.

```
#This is a comment
#print out Hello
print('Hello')
```

Multi-line comments

If we have comments that extend multiple lines, one way of doing it is to use hash (#)
in the beginning of each line. For example:

```
#This is a long comment
#and it extends
#to multiple lines
```

Another way of doing this is to use triple quotes, either ''' or """.

2. Variables And Simple Data Types

Python Variables

A variable is a named location used to store data in the memory.
It is helpful to think of variables as a container that holds data which can be
changed later throughout programming. For example,

```
number = 10

a, b, c = 5, 3.2, "Hello"

print (a)
print (b)
print (c)
```

Create a constant.py

```
PI = 3.14
GRAVITY = 9.8
```

Create a main.py

```
import constant

print(constant.PI)
print(constant.GRAVITY)
```

Literals

Literal is a raw data given in a variable or constant.
In Python, there are various types of literals they are as follows:

Numeric Literals
Numeric Literals are immutable (unchangeable). Numeric literals can belong to 3 different numerical types Integer, Float and Complex.

String literals
A string literal is a sequence of characters surrounded by quotes.
We can use both single, double or triple quotes for a string. And, a character literal is a single character surrounded by single or double quotes.

```
strings = "This is Python"
char = "C"
multiline_str = """This is a multiline string with more than one line code."""
unicode = u"\u00dcnic\u00f6de"
raw_str = r"raw \n string"

print(strings)
print(char)
print(multiline_str)
print(unicode)
print(raw_str)
```

Literal Collections

There are four different literal collections List literals, Tuple literals, Dict literals, and Set literals.

```
fruits = ["apple", "mango", "orange"] #list
numbers = (1, 2, 3) #tuple
alphabets = {'a':'apple', 'b':'ball', 'c':'cat'} #dictionary
vowels = {'a', 'e', 'i' , 'o', 'u'} #set
```

```
        print(fruits)
        print(numbers)
        print(alphabets)
        print(vowels)

print(1,2,3,4,sep='*')
# Output: 1*2*3*4

print(1,2,3,4,sep='#',end='&')
# Output: 1#2#3#4&

print('I love {0} and {1}'.format('bread','butter'))
# Output: I love bread and butter

print('I love {1} and {0}'.format('bread','butter'))
# Output: I love butter and bread
```

strings

Python String

What is String in Python?

A string is a sequence of characters.
A character is simply a symbol. For example, the English language has 26 characters.

```
# all of the following are equivalent
my_string = 'Hello'
print(my_string)

my_string = "Hello"
print(my_string)

my_string = '''Hello'''
print(my_string)

# triple quotes string can extend multiple lines
my_string = """Hello, welcome to
                    the world of Python"""
print(my_string)

# using triple quotes
print('''He said, "What's there?"''')

# escaping single quotes
print('He said, "What\'s there?"')
```

```
# escaping double quotes
print("He said, \"What's there?\"")

Escape Sequence          Description
\newline          Backslash and newline ignored
\\          Backslash
\'          Single quote
\"          Double quote
\a

ASCII Bell
\b          ASCII Backspace
\f          ASCII Formfeed
\n          ASCII Linefeed
\r          ASCII Carriage Return
\t          ASCII Horizontal Tab
\v          ASCII Vertical Tab
\ooo     Character with octal value ooo
\xHH     Character with hexadecimal value HH

# default(implicit) order
default_order = "{}, {} and {}".format('John','Bill','Sean')
print('\n--- Default Order ---')
print(default_order)

# order using positional argument
positional_order = "{1}, {0} and {2}".format('John','Bill','Sean')
print('\n--- Positional Order ---')
print(positional_order)

# order using keyword argument
keyword_order = "{s}, {b} and {j}".format(j='John',b='Bill',s='Sean')
print('\n--- Keyword Order ---')
print(keyword_order)

name = "nazmul haque"
print(name.title())
print(name.upper())
print(name.lower())

first_name = "nazmul"
last_name = "haque"
u full_name = first_name + " " + last_name
print(full_name)

print("Hello, " + full_name.title() + "!")
#or
message = "Hello, " + full_name.title() + "!"
```

```
print(message)

favorite_language = 'python '
>>> favorite_language
'python '
>>> favorite_language.rstrip()
'python'
>>> favorite_language
'python '


favorite_language = ' python '
>>> favorite_language.rstrip()
' python'
>>> favorite_language.lstrip()
'python '
>>> favorite_language.strip()
'python'
```

numbers

Integers

You can add (+), subtract (-), multiply (*), and divide (/) integers in Python.

Floats

Python calls any number with a decimal point a float.

```
age = 23
message = "Happy " + str(age) + "rd Birthday!"
```

Comments

Comments are an extremely useful feature in most programming languages.

```
# Say hello to everyone.
print("Hello Python people!")
```