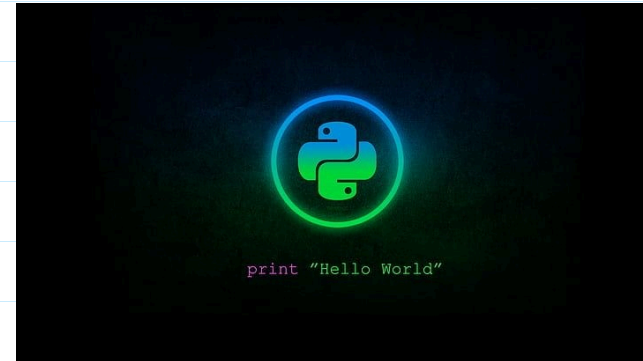


Week 1 Lecture : 3 (Basic Python Syntax and First Program in Python)

Thursday, 27 June 2024 12:53 PM

First Program (Hello Word)

Lets Jumps into the code



()

Print() Function

The print() function in Python is used to output data to the console.

Code :

Print ("Hello World")

A screenshot of a code editor window titled 'HelloWorld.py'. The editor shows a single line of code: `print("Hello World")`, which is highlighted with a red box. Below the code editor, there is a panel with tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'OUTPUT' tab is selected and circled in red. It shows the command `/usr/bin/python3 /Users/suyashchaudhary/HelloWorld.py` and the output `Hello World`, both of which are highlighted with red boxes. Red arrows point from the code in the editor to the 'OUTPUT' tab and from the output text to the terminal output.

Python Syntax Identifiers

When we talk about variables, functions, classes, or modules, we use “identifiers” to identify them.

Now just like for indentation, we have some naming rules here.

- The identifier will contain a combination of lowercase (a-z) or uppercase(A-Z) alphabets or numbers (0-9) or underscore (_). ✓ ✓ ✓
- ✗ • The Identifier cannot start with a digit
- ✗ • Any reserved words or keywords cannot be used as identifier names (you'll read more about keywords in the article soon)
- ✗ • Symbols or Special characters cannot be used in the identifier. \$ #
- Remember that Python is a case-sensitive language, so var and VAR are two different identifiers

Examples of correct naming: var, Robot, python_007, etc

Examples of **incorrect** naming: 97learning, hello etc

Some identifier examples for variables:

var = 310 ✓

My_var = 20 ✓

My_var344 = 10 ✓

STRING = "PYTHON" ✓

fLoat = 3.142 ✓

Sahil
sahil
SAMI L

labC while
@abc if
for

```
Identifiers.py x
Users > suyashchaudhary > Identifiers.py > ...
1 var = 310
2 My_var = 20
3 My_var344 = 10
4 STRING = "PYTHON"
5 fLoat = 3.142
6
```

```
Identifiers.py 6 x
Users > suyashchaudhary > Identifiers.py > ...
1 labC = 23
2 @abc = "Hello"
3 for = "variable"
4
5 print(labC, @abc, for)
```

```
print(var, My_var, My_var344, STRING, float)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + v

```
/usr/bin/python3 /Users/suyashchaudhary/Identifiers.py  
suyashchaudhary@Suyashs-MacBook-Air ~ % /usr/bin/python3 /Users/suyashchaudhary/Identifiers.py  
310 20 10 PYTHON 3.142
```

Valid Identifiers

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + v

```
/usr/bin/python3 /Users/suyashchaudhary/Identifiers.py  
suyashchaudhary@Suyashs-MacBook-Air ~ % /usr/bin/python3 /Users/suyashchaudhary/Identifiers.py  
File "/Users/suyashchaudhary/Identifiers.py", line 1  
1abc = 23  
^  
SyntaxError: invalid syntax
```

Invalid Identifiers

Python Keywords

They are some unique purpose words (Reserved Words) that can be used only for specific cases and not as identifiers.

Every language contains words and a set of rules that would make a sentence meaningful.

Similarly, in Python programming language, there are a set of predefined words, called Keywords which along with Identifiers

will form meaningful sentences when used together.

Python keywords cannot be used as the names of variables, functions, and classes.

False

__peg_parser_
assert
break
def
else
for
if
is
not
raise
while

None

and
async
class
del
except
from
import
lambda
or
return
with

True

as
await
continue
elif
finally
global
in
nonlocal
pass
try
yield

Getting the List all Python Keywords

Code :

```
import keyword  
print(keyword.kwlist)
```

OUTPUT:

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global',  
'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

What are variables?

Python Variable is containers that store values.

There is no specific command in Python to declare a variable.

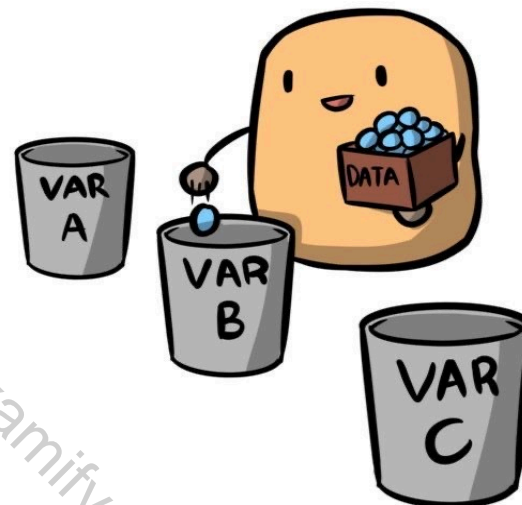
A variable is created the moment a value is assigned to it.

Code :

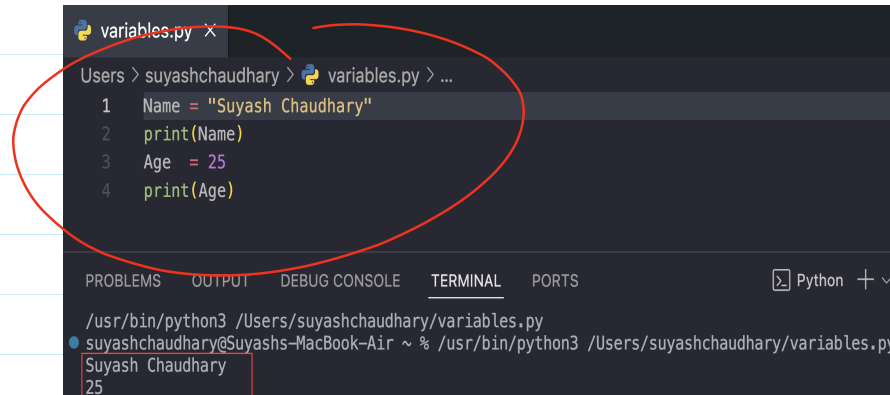
```
Name = "Suyash Chaudhary"
```

```
print(Name)
```

```
Age = 25
```



print(age)



```
variables.py x
Users > suyashchaudhary > variables.py > ...
1 Name = "Suyash Chaudhary"
2 print(Name)
3 Age = 25
4 print(Age)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + v
/usr/bin/python3 /Users/suyashchaudhary/variables.py
suyashchaudhary@Suyashs-MacBook-Air ~ % /usr/bin/python3 /Users/suyashchaudhary/variables.py
Suyash Chaudhary
25
```

Variables Assignment in Python Or variable declarations

We have assigned a number, a floating point number, and a string to a variable such as age, salary, and name.

```
age = 25 # An integer assignment
salary = 92137.02 # A floating point
name = "Suyash" # A string

print(age)
print(salary)
print(name)
```

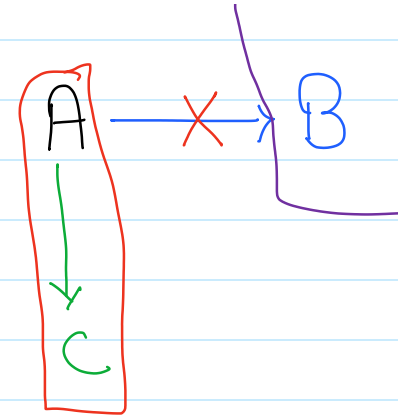
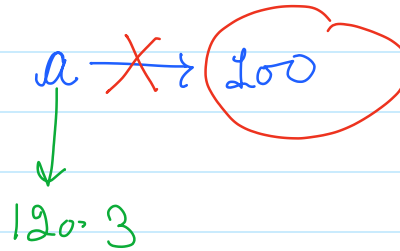
EC

Re-declaring variables in Python

We can re-declare the Python variable once we have declared the variable and define variable in python already.

```
Number = 100                # var declaration
print("Before declare: ", Number)

Number = 120.3              # re-declare the var
P:
print("After re-declare:", Number)
```



Python Assign values to Multiple variables

Python allows assigning a single value to several variables simultaneously with “=” operators.

```
a = b = c = 10
print(a)
print(b)
print(c)
```

a = b = c = 10

Assigning different values to multiple variables

Python allows adding different values in a single line with “,” operators.

```
a, b, c = 1, 20.2, "python"
```

a, b, c = 1, 20.2, "python"

print(a)

print(b)

print(c)

Can we Use the Same Name for Different types?

YES, If we use the same name, the variable starts referring to a new value and type.

a = 10

a = "Python"

print(a)

⇒ Int
⇒ String

a → 10
↓
Python

How does + operator work with variables?

The Python plus operator + provides a convenient way to add a value if it is a number and concatenate if it is a string.

a = 10

b = 20

print(a+b)

Output : 30

a = "Suyash"

10 + 20 = 30
"10" + "20" = "1020"

```
b = "Chaudhary"
```

```
print(a+b)
```

Output : "Suyash Chaudhary"

Can we use + for different Data Types also?

No use for different types would produce an error.

```
a = 10
```

```
b = "Geeks"
```

```
print(a+b)
```

TypeError: unsupported operand type(s) for +: 'int' and 'str'

Handwritten notes:
a = 10 ^{Int}
b = "Suyash" ^{str}
⊕
TypeError

Delete a variable

Python automatically removes variables and functions from memory when they are no longer in use, thereby liberating space.

Users have the option to manually eliminate variables and functions as well. You can delete a variable using the **del** command in Python.

```
number=101
```

```
del number
```

```
print(number)
```

NameError: name 'number' is not defined

Object References

Lets understanding how the Python interpreter works when we declare a variable.

We all know that Python is an object-oriented language. Hence, every object belongs to a specific class.

```
message="Welcome , Hello World".
```

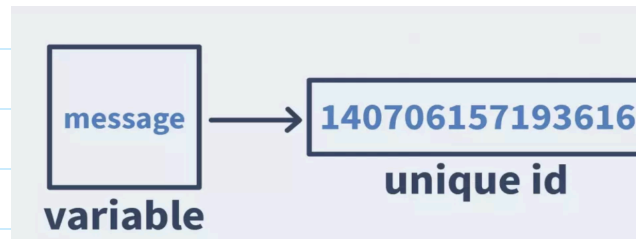
```
print(message)
```

```
print(type(message))
```

```
Welcome , Hello World
```

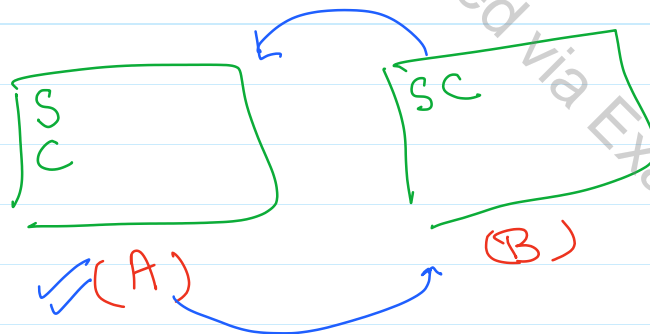
```
<class 'str'>
```


A string object is created of the class str!



ToDo

Print("Sugesh")
Print("Chauhanary")





Examify / Uploaded via Examify