Module 3

Basic Fundamentals of Python

Print Hello world

When we want to show something as output, we use print function

```
print ("Hello world")
```

```
Hello world
```

Variables

Variables are place holders for storing data

```
x = 10
```

- `10` ← value to be assigned
- `=` ← Assignment operator
- `x` ← Variable name

Rules for naming Python variables :

- A variable name must start with a letter (a-z, A-Z) or the underscore character (_)
- A variable name cannot start with a number (0-9)
- A variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9 and _)
- Variable names are case sensitive (Age, age and AGE are three different variables)
- Inbuilt Keywords should not be used as identifiers

Variable names

```
a = 5
print(a)
```

```
5
```

```
3a = 34
```

```
SyntaxError : invalid syntax
```

```
uva_89 = 45
print (uva_89)
```

```
45
```

help("keywords")

Here is a list of Python keywords. Enter any keyword to get more help.

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

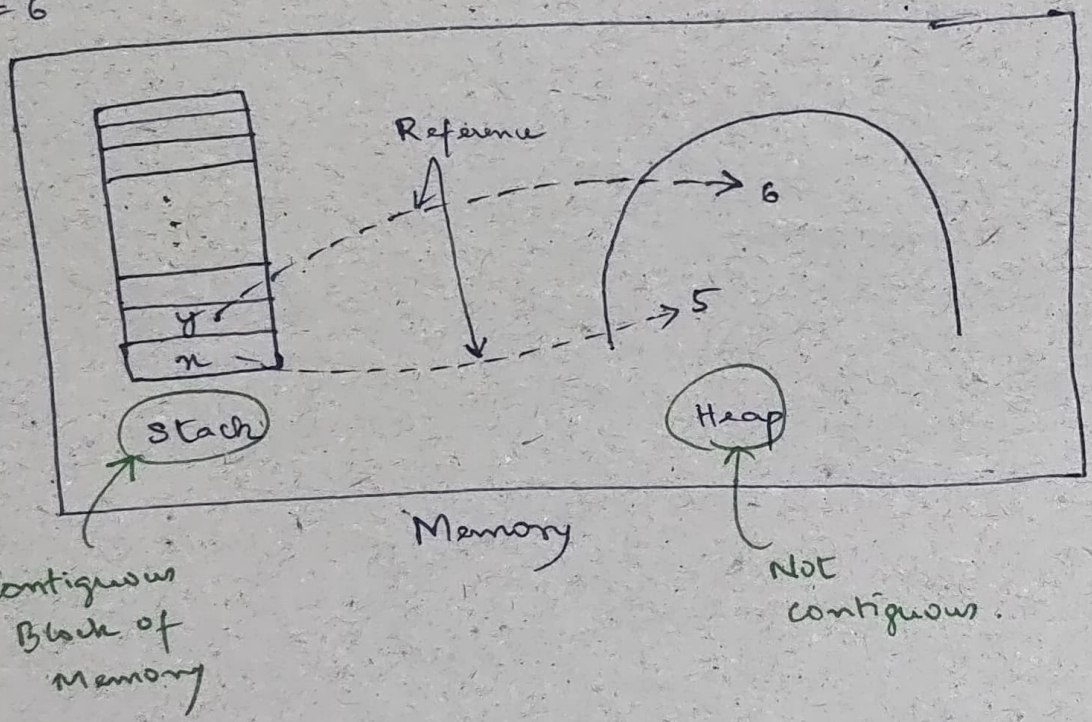
for = 89

SyntaxError: invalid syntax

Stack and Heap memory (Not Data Structures)

x = 5

y = 6



id function

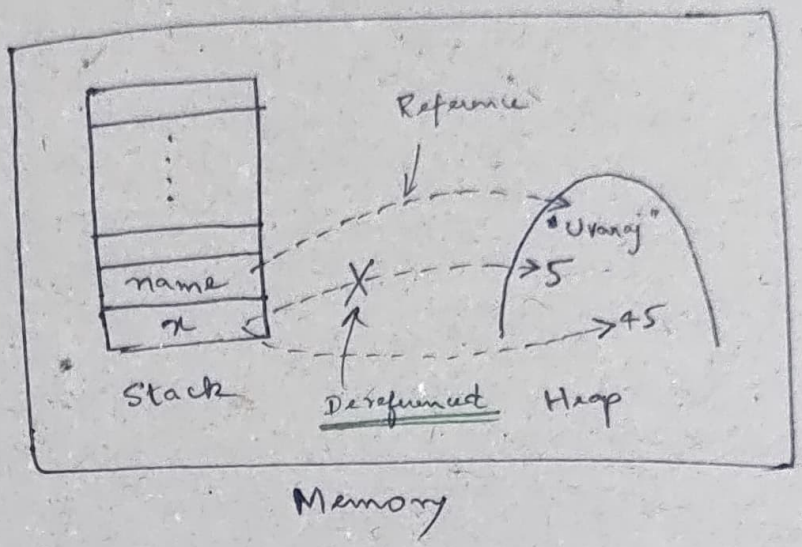
id() function returns the address, where the value of the variable is stored.

```
x = 5
id(x)
140166123678128

x = 45
print(id(x))
140166123679408

print(x)
45

name = "Uvaraj"
print(id(name))
140166125608624
```



Python Comments

Python uses the '#' symbol to comment out a particular line of code.

```
# This is a variable
a = 5
# x = 23
# y = 45
z = 32
print(z)
32

Print(y)
NameError: name 'y' is not defined
```

Why comments ?

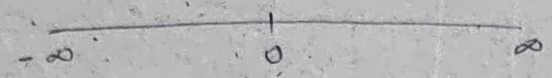
- To describe the program
- To temporarily remove part of code

Data types in Python

In Python, there are 5 different data types.

- Integers (int)
- Floats (float)
- Booleans (bool)
- Strings (str)
- NOne (NoneType)

Integers are all the whole numbers, starting from $-\infty$ to $+\infty$



All the whole numbers on this line are Integers. Eg. 5, 10, -15,

Floats are also in the range from $-\infty$ to $+\infty$, but there will be a decimal point (i) a dot that separates the whole number part from the fractional part of the number. Eg. 1.5, 1.0, 2.6,

```
# Integer
number = 24
print (number)
24

# type() function
a = 23
print (type(a))
print (a)
<class 'int'>
23

print (a)
23
print ("a")
a

# Float
f = 1.0
print (f)
print (type(f))
1.0
<class 'float'>
```

Built-in function used to return the type of data stored in the objects or variables.

Prints the value in the variable

Prints the string

Booleans represent one of the two values : True or False.
It is used to evaluate if an expression is True or False.

```
# Boolean
b = True
print (type (b))
print b
<class 'bool'>
True

b1 = False
print (type (b1))
<class 'bool'>

b2 = true
NameError : name 'true' is not defined
```

Strings in Python are surrounded by either single quotation marks (or) double quotation marks. Eg. 'hello' is same as "hello".

```
# Strings
s = " Uvaraj 123 @ + "
Print (type (s))
print (s)
<class 'str'>
Uvaraj 123 @ +

s1 = ' Uvaraj 123 @ + '
print (type (s1))
print (s1)
<class 'str'>
Uvaraj 123 @ +
```

Both are SAME

```
# Multiline string
m = " " "
  this is a multiline string !
" " "
print (m)
print (type (m))
this is a multiline string !
<class 'str'>
```

None is used to indicate that a variable or object doesn't have value assigned to it. It is similar to NULL in other languages.

```
# None
n = None
print (n)
Print (type (n))
```

'None' is the only value of the NoneType datatype.

```
None
<class 'NoneType'>
```

---

input () function in Python

Python input () is used to take user input. By default, it returns the user input in the form of String.

```
name = input ()
  Uvaraj
print (name)
Uvaraj
print (type (name))
<class 'str'>
```

String as input

```
x = input ()
  1.5
print (x)
1.5
print (type (x))
<class 'str'>
```

Number as input

```
y = float (x)
print (y)
print (type (y))
1.5
<class 'float'>
```

Type Casting (str → float)

"1.5" → 1.5

name1 = int(name) ← Type casting (str → int)

Value Error : invalid literal "Uvaraj" → ?
for int() with base 10: 'Uvaraj'
Not possible..

y1 = int(x)

value Error : invalid literal
for int() with base 10: '1.5'

Typecasting (str → int)
"1.5" → 1
Not possible directly..

y1 = float(x)
z = int(y1)
print(z)
print(type(z))

Typecasting (str → float → int)
"1.5" → 1.5 → 1

1
<class 'int'>