

# Python Tutorial for Beginners



# Agenda for Today's Session

**01** Get a brief understanding of what exactly is Python

## Python Tokens



**Introduction to Python 3.0**

**03** Learn and implement various data types in python

## Flow Control



**Data Types in Python**

**05** Understand how to define and call a function in python

## File Handling using Python



Understand the flow of execution of a program

**Function in Python**

**06** Learn how to open, read and write with file in python

Learn about Keywords, Identifiers, Literals and Operators

**02**

**04**

Do you know who created Python?

# Introduction To Python



Created in 1991 by **Guido van Rossum**

Ever wondered Why is it named Python?

# Introduction To Python



Named after **Monty's Python Flying Circus**

# Introduction To Python



## Why should YOU learn Python?



# Introduction To Python



Python can be used in lot of places...



# Introduction To Python



## Popularity of Python in Industry...



# Installing Python



# Python: Variable

Assigning value

Multiple assignment

## Assigning single value to a variable...

### Input

```
>>>a = 10
>>>name = 'Victor'
>>>salary = 2000.23
>>>print (a)
>>>print (name)
>>>print (salary)
```

### Output

```
>>>10
>>>Victor
>>>2000.23
```

# Python: Variable

Assigning value

Multiple assignment

## Assigning multiple value to a variable...

Input

```
>>>a=b=c=10
>>>x,y,z=10,20,30
>>>print (y)
>>>print (a)
```

Output

```
>>>20
>>>10
```

# Python: Tokens

Keywords

Identifiers

Literals

Operators

## What are Keywords?

- Python Keywords are special reserved words
- Convey a special meaning to the compiler/interpreter
- Each keyword have a special meaning and a specific operation
- **NEVER** use it as a variable

True	False	None	and	as
asset	def	class	continue	break
else	finally	elif	del	except
global	for	if	from	import
raise	try	or	return	pass
nonlocal	in	not	is	lambda

# Python: Tokens

Keywords

Identifiers

Literals

Operators

## What are Identifiers?

Identifiers are the name used to identify a variable, function, class or an object

**RULES** defined for naming an Identifiers:

- No special character except underscore ( \_ ) can be used as an identifier
- Keyword should not be used as an identifier name
- Python is case sensitive, i.e Var and var are two different identifier
- First character of an identifier can be character, underscore ( \_ ) but not digit

## Literals: Constant used in Python...

# Python: Tokens

Keywords

Identifiers

Literals

Operators

String Literals

Numeric Literals

Boolean Literals

Special Literals

# Literals in Python

String Literals

Numeric Literals

Boolean Literals

Special Literals

## What are String Literals?

- Formed by enclosing a text in the quotes
- Both single and double quotes can be used

Input

```
>>>name1="John"  
>>>name2='James'  
>>>print(name1)  
>>>print(name2)  
>>>text1='hello\  
World'  
>>>text1  
>>>multiline='''st1  
...st2  
...st3'''
```

Output

```
'helloworld'  
John  
James  
st1  
st2  
st3
```

## What are Numeric Literals?

# Literals in Python

String Literals

Numeric Literals

Boolean Literals

Special Literals

Int	Long	Float	Complex
+ve and -ve numbers(integers ) with no fractional part. Ex: 100,-234	Unlimited integer size followed by upper or lowercase L.	Real numbers with both integer and fractional part Ex: 233424243L	In the form of a+bj. 'a' forms the real part & b forms the imaginary part. Ex: 3.14j

- In Python, value of an integer is not restricted by the number of bits and can expand to the limit of the available memory
- No special arrangement is required for storing large numbers

# What are Boolean Literals?

Can have only two values:

- True
- False

# Literals in Python

String Literals

Numeric Literals

Boolean Literals

Special Literals

# What are Special Literals?

Python has one special literal: None

Used to specify to the field that is not created

## Example

```
>>> val1=10
>>> val2=None
>>> val1
10
>>> val2
>>> print(val2)
None
>>>
```

# Literals in Python

String Literals

Numeric Literals

Boolean Literals

Special Literals

# Types of Operators...

Arithmetic Operator

Assignment Operator

Comparison Operator

Logical Operator

Bitwise Operator

Identity Operator

Membership Operator

# Operators in Python

Keywords

Identifiers

Literals

Operators

# Operators in Python

Arithmetic Operator

Assignment Operator

Comparison Operator

Logical Operator

Bitwise Operator

Identity Operator

Membership Operator

Takes two operand to perform operations on them

Operators

+, -, \*, /, %

Example

```
>>>1+2  
3  
>>>1-2  
-1  
>>>2%1  
0
```

# Operators in Python

Arithmetic Operator

Assignment Operator

Comparison Operator

Logical Operator

Bitwise Operator

Identity Operator

Membership Operator

## Assign a value to a variable

Operators

=, +=, -=, \*=

Example

```
>>>a=10
>>>a*=10 #a=10*10
>>>print(a)
100
```

# Compare two values and returns true or false as output

## Operators

=, +=, -=, \*=

Arithmetic Operator

Assignment Operator

Comparison Operator

Logical Operator

Bitwise Operator

Identity Operator

Membership Operator

## Example

```
>>>a=10
>>>a*=10 #a=10*10
>>>print(a)
100
```

# Operators in Python

Arithmetic Operator

Assignment Operator

Comparison Operator

Logical Operator

Bitwise Operator

Identity Operator

Membership Operator

Used to compare two values and returns  
true or false as output

Operators

and, or , not

Example

```
>>> a=10<10 and 2>-1  
>>> print(a)  
False
```

# Operators in Python

Arithmetic Operator

Assignment Operator

Comparison Operator

Logical Operator

Bitwise Operator

Identity Operator

Membership Operator

Used to perform bitwise calculation

Operators

&, |, >>, <<, ~

Example

```
>>>7 | 5  
7  
>>>7&5  
5
```

How it is calculated?

$$\begin{array}{r} 111 \\ 101 \\ \hline 111 \end{array} \quad \begin{array}{r} 7 \\ 5 \\ \hline 7 \end{array}$$

# Operators in Python

Arithmetic Operator

Assignment Operator

Comparison Operator

Logical Operator

Bitwise Operator

Identity Operator

Membership Operator

Test if the two operands share an identity

Operators

is, is not

Example

```
>>>x = 10
>>>x is 10
True
>>>x = 10
>>>x is not 10
False
```

# Operators in Python

Arithmetic Operator

Assignment Operator

Comparison Operator

Logical Operator

Bitwise Operator

Identity Operator

Membership Operator

## Test whether a value is a member of a sequence

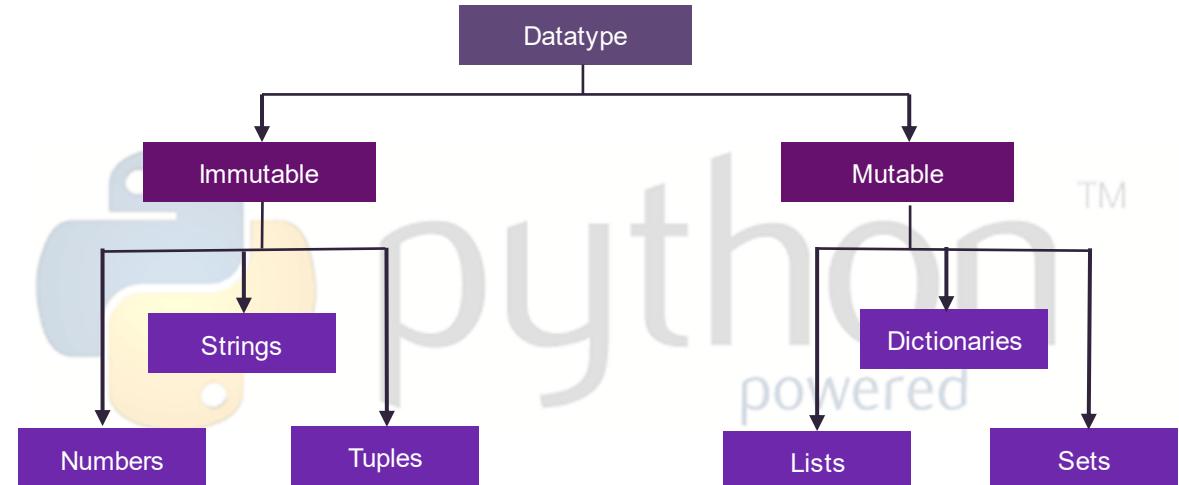
Operators

in, not in

Example

```
>>>pets=[ 'dog' , 'cat' , 'wolf' ]  
>>>'lion' in pets  
False  
>>>'wolf' in pets  
True  
>>>'me' in 'appointment'  
True
```

# Python: Datatype



# Datatypes in Python

Numbers

Strings

Tuples

Lists

Dictionaries

Sets

## Example

```
message = "Hello World"  
num = 1234  
rate = 13.6  
print(type(message)) #return a string  
print(type(n)) #return an integer  
print(type(pi)) #return a float
```

# Datatypes in Python

Numbers

Strings

Tuples

Lists

Dictionaries

Sets

## Example

```
var1 = 'Hello-World!'
var2 = 'PythonTutorial'

print(var1[0]) # prints the first
character in the string `H`
print(var2[1:5]) # prints the
substring `ytho`
```

# Datatypes in Python

Numbers

Strings

Tuples

Lists

Dictionaries

Sets

find()

replace()

split()

count()

# Datatypes in Python

Numbers

Strings

Tuples

Lists

Dictionaries

Sets

## Tuples are a group of values within ()

Example

```
>>>myGroup = ('a', 'b', 'c', 'd')
```

repetition

indexing

slicing

# Datatypes in Python

Numbers

Strings

Tuples

Lists

Dictionaries

Sets

## Tuples are a group of values within [ ]

Example

```
>>>myGroup = ('a', 10, 7.12, 'data')
```

concatenation

repetition

slicing

extend

insert

# Datatypes in Python

Numbers

Strings

Tuples

Lists

Dictionaries

Sets

## Tuples are a group of values within ()

### Example

```
myDict = { 1: 'John' , 2: 'Bob' , 3: 'Alice' }
```

Key

Value

empty dictionary

Integer keys

mixed keys

pairing

accessing  
dictionary

# Datatypes in Python

Numbers

Strings

Tuples

Lists

Dictionaries

Sets

## Unordered collection of items within {}

Example

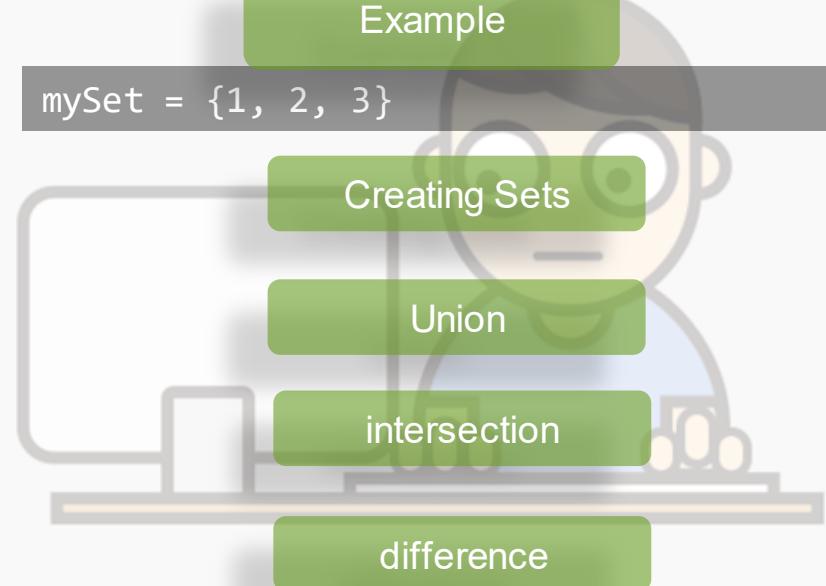
```
mySet = {1, 2, 3}
```

Creating Sets

Union

intersection

difference



# Python: Test Yourself



**Q.1** What would be the output of the following code?

Fill the missing code

```
test1 = 'PythonProgramming'  
print(test1[2:6])
```



**Note:** Post your answer in the comment section below

# Python: Flow Control



Defines the flow of the execution of  
program...

if-else

Nested if-else

for

while

break

continue

# Python: Flow Control

if-else

Nested if-else

for

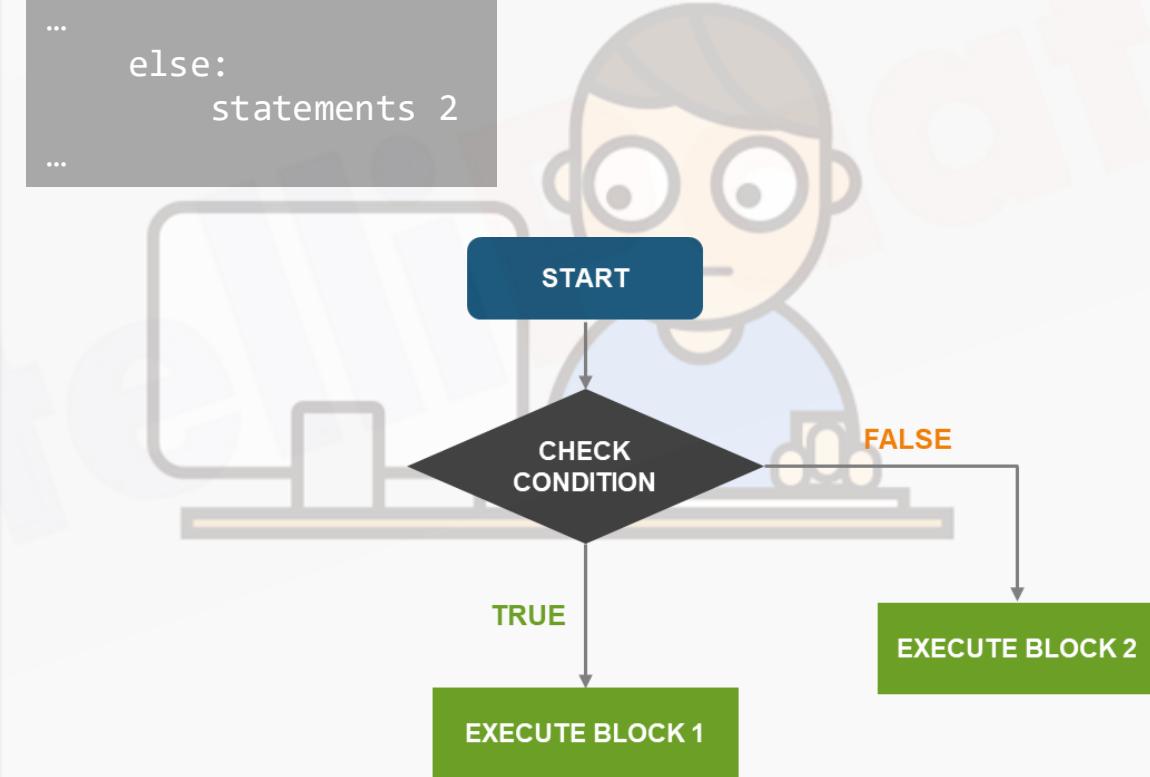
while

break

continue

## Syntax

```
if (condition):  
    statements 1  
...  
else:  
    statements 2  
...
```



# Python: Flow Control

if-else

Nested if-else

for

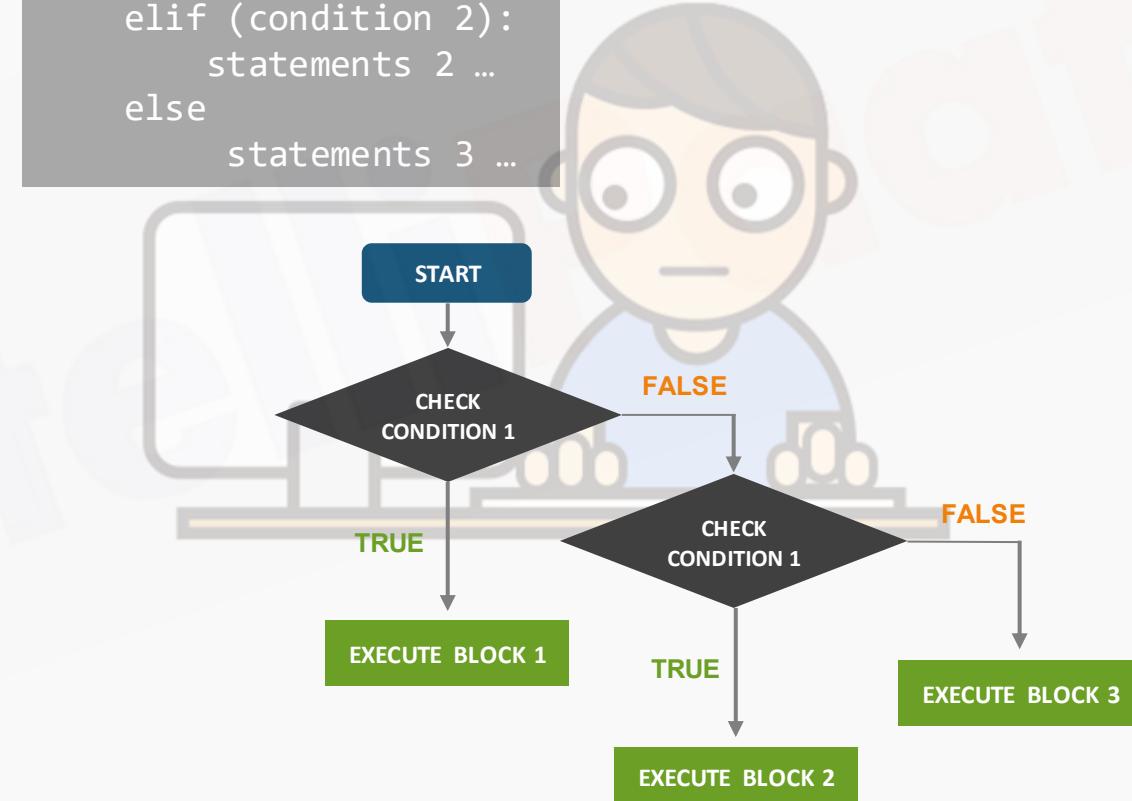
while

break

continue

## Syntax

```
if (condition 1):  
    statements 1 ...  
elif (condition 2):  
    statements 2 ...  
else  
    statements 3 ...
```



# Python: Flow Control

if-else

Nested if-else

for

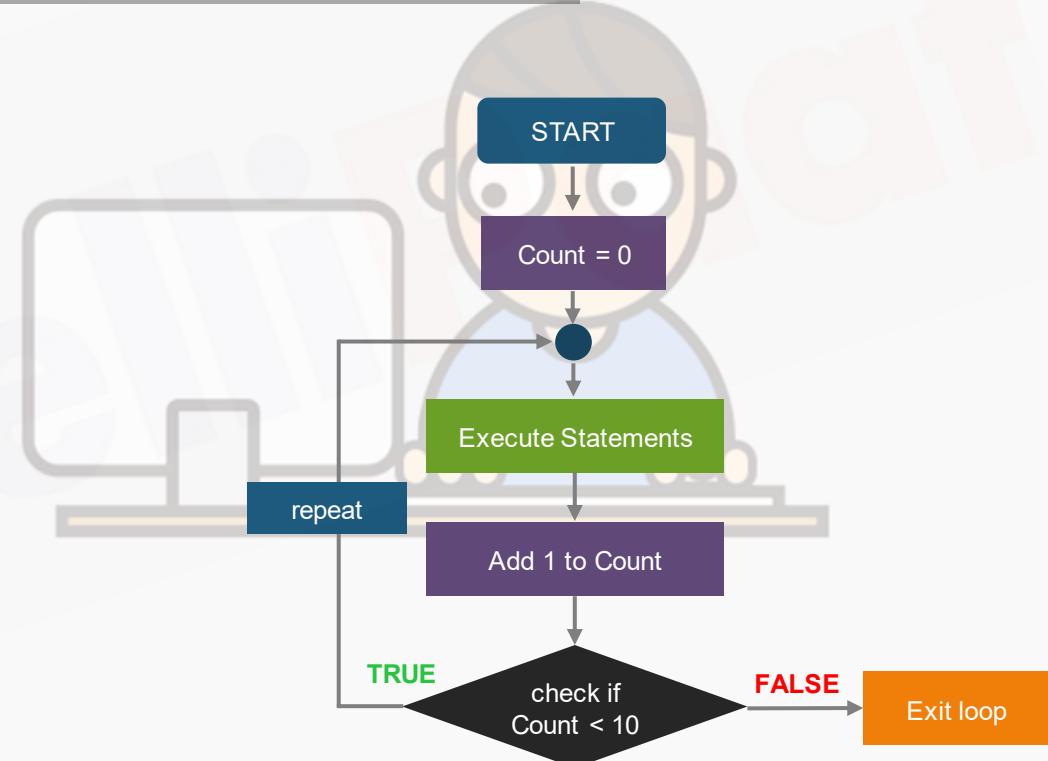
while

break

continue

## Syntax

```
for iterating_var in sequence:  
    execute statements
```



# Python: Flow Control

if-else

Nested if-else

for

while

break

continue

## code

```
1 fruits = ["apple", "banana", "cherry"]  
2 for x in fruits:  
3     print(x)
```

## output



apple  
banana  
cheery

## variable

```
x in fruit[0] = apple  
x in fruit[1] = banana  
x in fruit[2] = cherry
```



# Python: Flow Control

if-else

Nested if-else

for

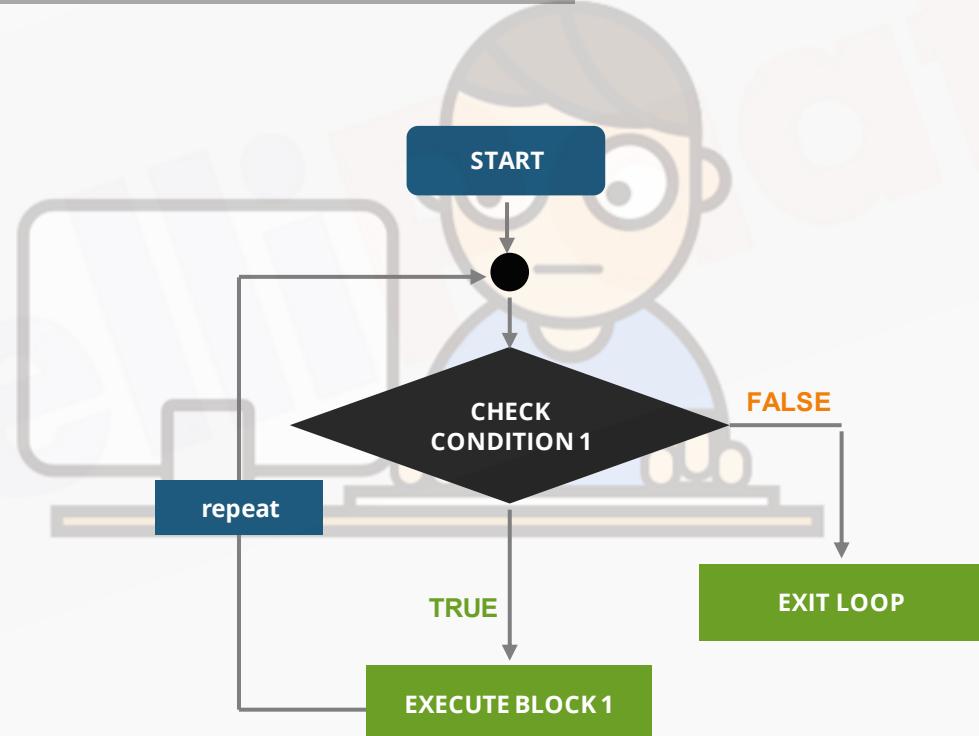
while

break

continue

## Syntax

```
while (condition is True):  
    statements1...
```



# Python: Flow Control

if-else

Nested if-else

for

while

break

continue

## code

```
1 a = 1
2 while a<5:
3     print(a)
4     a+=2
```

## output

1  
3

FINAL

## variable

a = 1 is less than 5

a = a+2= 1+2= 3

a = 3 is less than 5

a = 5 is less than 5

# Python: Flow Control

if-else

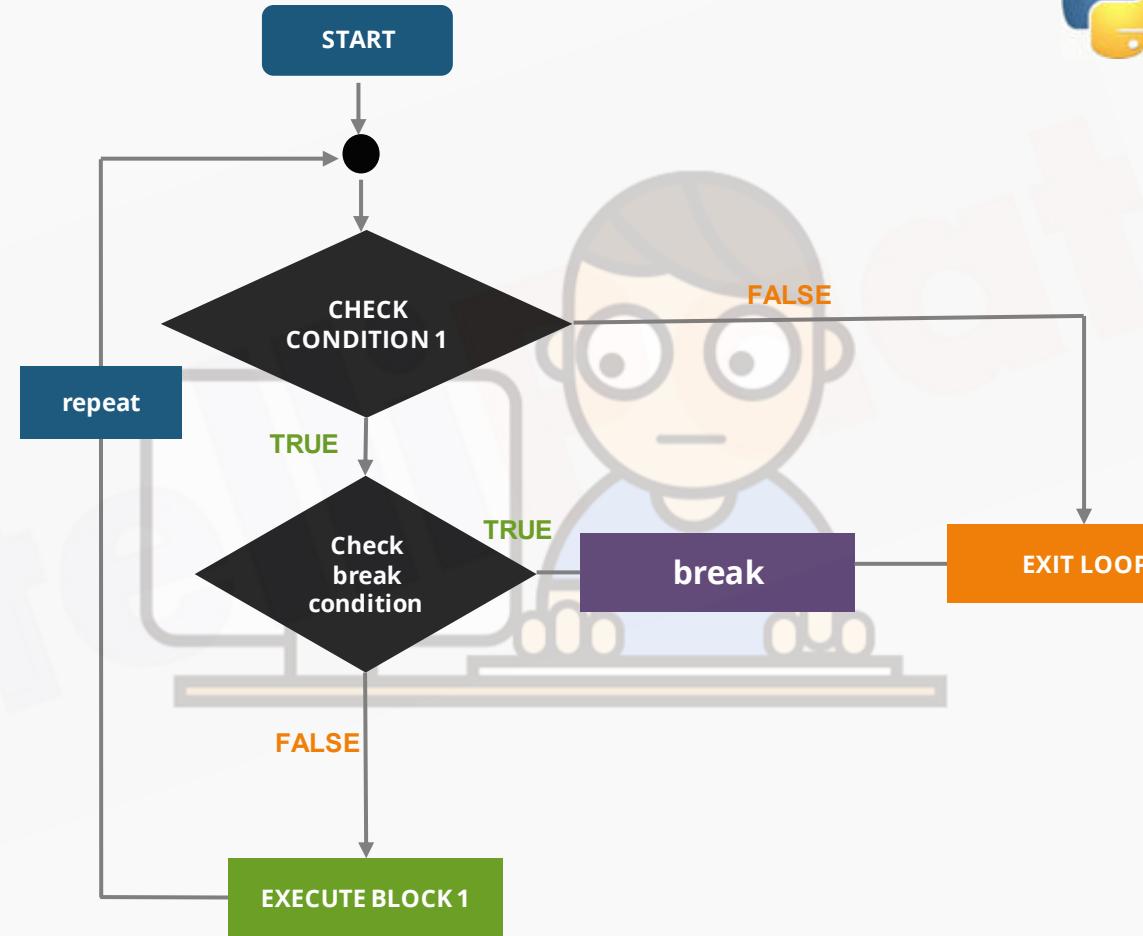
Nested if-else

for

while

break

continue



# Python: Flow Control

if-else

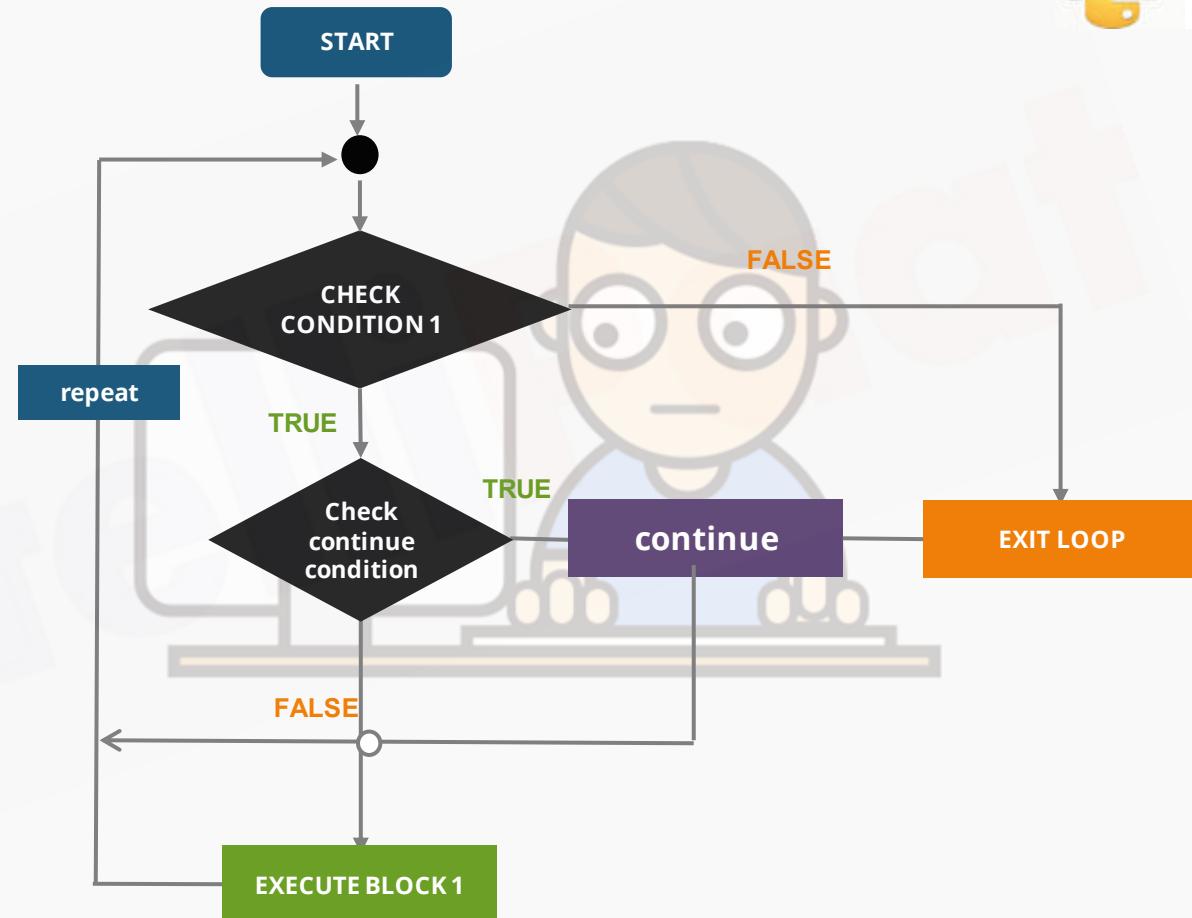
Nested if-else

for

while

break

continue

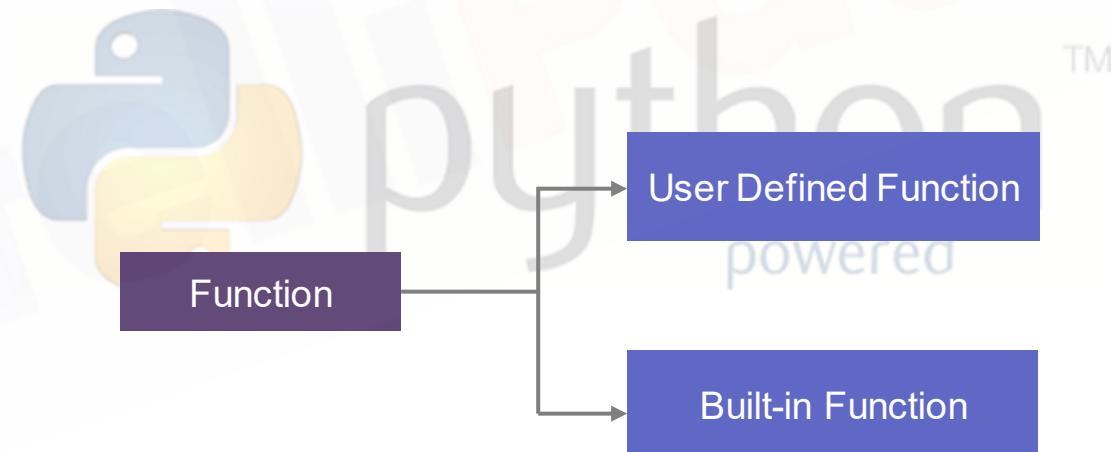


# Python: Function



## What are functions?

A function is a block of organized, reusable sets of instructions that is used to perform some related actions



# Python: Function

User Defined Function

Built-in Function

## How to define a function?

### Syntax

```
def func_name (arg1,  
arg2, arg3, ....):
```

```
    statements...  
    return
```

```
[expression]
```



### Example

```
def add ( a, b )  
    sum = a + b  
    return sum
```

# Python: Function

User Defined Function

Built-in Function

## How to define a function?

- `abs()`: Returns the absolute value of a number
- `all()`: Returns True if all items in an iterable object are true
- `any()`: Returns True if any item in an iterable object is true
- `ascii()`: Returns a readable version of an object. Replaces non-ascii characters with escape character
- `bin()`: Returns the binary version of a number
- `bool()`: Returns the Boolean value of the specified object

# Python: Function Call



Calling the defined function...

Pass by Value

Pass by Reference

# Python: Function Call

Pass by Value

Pass by Reference

## Calling a function by passing the value...

### Example

```
>>>a=10
>>>def ChangeIt(b):
...     print("value of b is", b)
...     b=100
...     print("New value of b is", b)
...
>>>ChangeIt(a)
```

# Python: Function Call

Pass by Value

Pass by Reference

## Calling a function by passing the value...

### Example

```
>>>c=[10,20,30]
>>>def ChangeThem(d):
...     print("value of d is", d)
...     d[0]=99
...     d[1]=98
...     print("New value of d is", d)
...
>>>ChangeThem(c)
```

# Python: Function Call

Pass by Value

Pass by Reference

*pass by reference*

cup = 

fillCup( )

*pass by value*

cup = 

fillCup( )

# Python: Test yourself



## Q.2

Following function takes two parameters and print the first one. Fill the missing code:

Fill the missing code

```
def my_function(fname, lname):  
    print(.....)
```



**Note:** Post your answer in the comment section below

# Python: Test Yourself



## Q.3

A function which return the x parameter + 10.

Fill the missing code:

Fill the missing code

```
def my_function(x):
```

..... . .

..... . .

.....



**Note:** Post your answer in the comment section below

# Python: Test Yourself



## Q.4 What is the output of ..

Output??

```
fruits = ("apple", "banana", "cherry")
if "mango" in fruits:
    print(3+4)
else
    print(2*2)
```



**Note:** Post your answer in the comment section below

# Python: Lambda



## What is Lambda Function?

Anonymous function, i.e a function having no name

### Syntax

```
lambda arguments : expression
```



### Example

```
x = lambda a : a + 10  
print(x(5))
```



**Note:** Lambda function cannot contain more than one expression

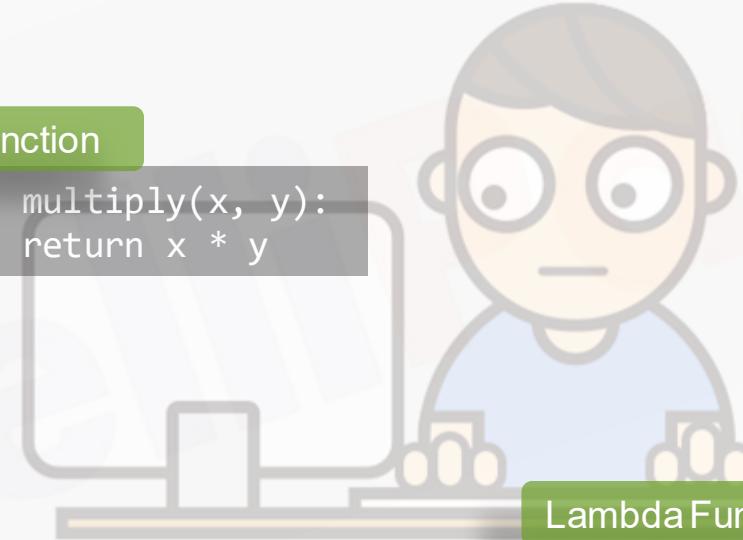
# Python: Lambda



## Function vs Lambda Function

Function

```
def multiply(x, y):  
    return x * y
```



Lambda Function

```
r = lambda x, y: x * y  
r(12, 3) # call the  
lambda function
```

# Python: Lambda

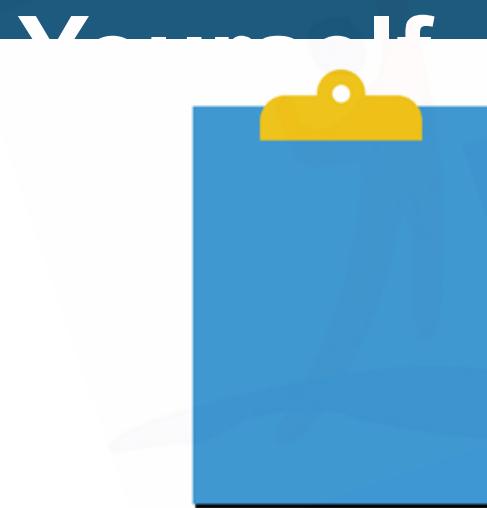


## Power of Lambda: Anonymous Function inside another function...

### Example

```
def myfunc(n):  
    return lambda a : a + n  
mySum = myfunc(3)  
print(mySum(10))
```

# Python: Test



**Q.5** Write a lambda function that sums arguments a, b, and c and print the result



**Note:** Post your answer in the comment section below

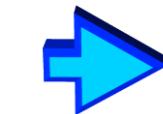
# Python: Test Yourself



## Q.6 What is the output of ..

Output??

```
def myfunc(n):  
    return lambda a : a * n  
  
mydoubler = myfunc(2)  
print(mydoubler(11))
```



**Note:** Post your answer in the comment section below

## What is an Array?

Used to store multiple values in one single variable.

Storing in multiple variable

```
car1 = "Ford";  
car2 = "Volvo";  
car3 = "BMW";
```

Using Array

```
cars = ["Ford", "Volvo", "BMW"]
```



**Note:** Python does not have built-in support for Arrays, but Python lists can be used instead.

# Python: Arrays

## Accessing the element

```
>>>x = cars[0]  
Ford
```

## Modifying the element

```
>>>cars[0] = "Honda"  
>>>print(cars[0])  
Honda
```

## Length of array

```
>>>x = len(cars)  
3
```

## Looping the Array

```
for x in cars:  
    print(x)
```

## Adding the element

```
>>>cars.append("Opel")  
>>>print(len(cars))  
4
```

## Removing one element

```
>>>cars.pop(1)
```

## Removing specific element

```
cars.remove("Volvo")
```

# Python: Classes/Object

## What is a class and an object in Python?

- Python is an object-oriented programming language
- Almost everything in Python is an object, with its properties and methods
- A Class is like a "blueprint" for creating objects



Class

```
class MyClass:  
    x = 5
```

Object

```
obj1 = MyClass()  
print(obj1.x)
```

# Python: File Handling

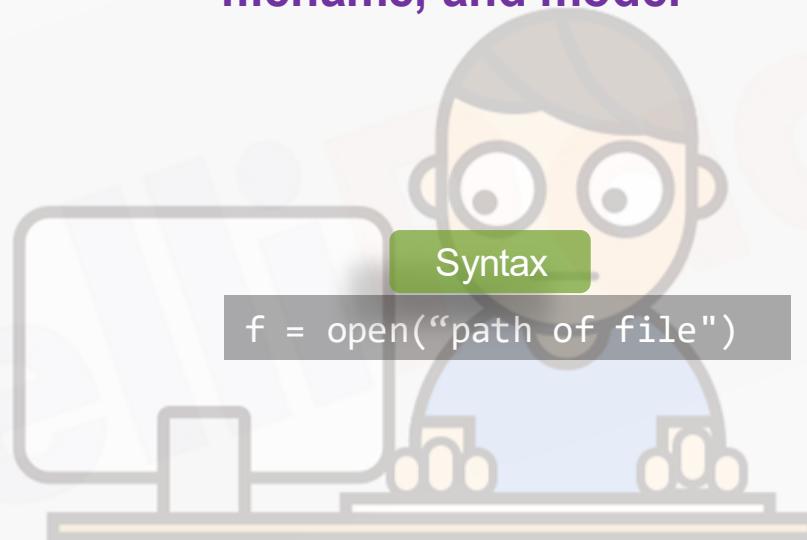
Open

Read

Write/Create

Delete

The `open()` function takes two parameters;  
filename, and mode.



# Python: File Handling

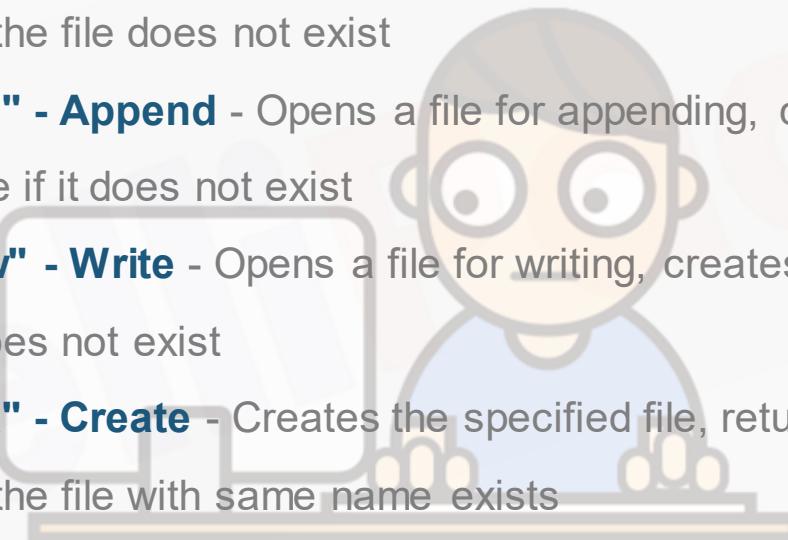
Open

Read

Write/Create

Delete

- **"r" - Read** - Default value. Opens a file for reading, error if the file does not exist
- **"a" - Append** - Opens a file for appending, creates the file if it does not exist
- **"w" - Write** - Opens a file for writing, creates the file if it does not exist
- **"x" - Create** - Creates the specified file, returns an error if the file with same name exists



Example

```
f = open("path of file","rt")
```

# Python: File Handling

Open

Read

Write/Create

Delete



## Example

```
f = open("demofile.txt", "r")
print(f.read())
```

## Reading parts of file

```
f = open("demofile.txt", "r")
print(f.read(5)) #Return the 5 first
characters of the file
```

## Reading first line

```
f = open("demofile.txt", "r")
print(f.readline()) #readline() is
used to return one line
```



# Python: File Handling

Open

Read

Write/Create

Delete

Reading first line

```
f = open("demofile.txt", "r")
print(f.readline())
print(f.readline()) # using readline()
twice prints first two line
```

Loop through the file

```
#Read the file line by line
f = open("demofile.txt", "r")
for x in f:
    print(x)
```

# Python: File Handling

Open

Read

Write/Create

Delete

To write to an existing file, you must add a parameter to the open() function

- **"a" - Append** – will append to the end of the file
- **"w" - Write** – will overwrite any existing content

Example: Append

```
f = open("demofile.txt", "a")
f.write("Now the file has one more line!")
```

Example: Overwrite

```
f = open("demofile.txt", "w")
f.write("Woops! I have deleted the content!")
```

# Python: File Handling

Open

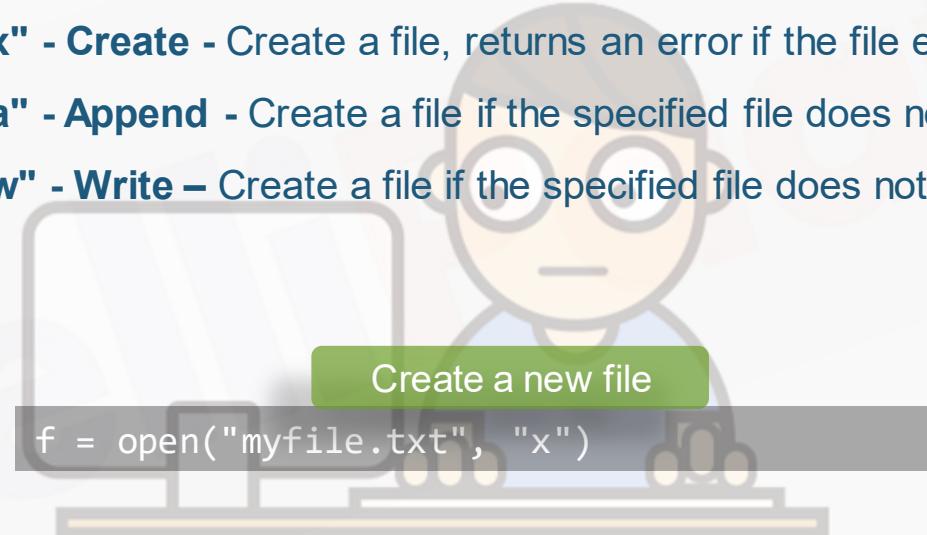
Read

Write/Create

Delete

## Creating a new file...

- **"x"** - **Create** - Create a file, returns an error if the file exist
- **"a"** - **Append** - Create a file if the specified file does not exist
- **"w"** - **Write** – Create a file if the specified file does not exist



# Python: File Handling

Open

Read

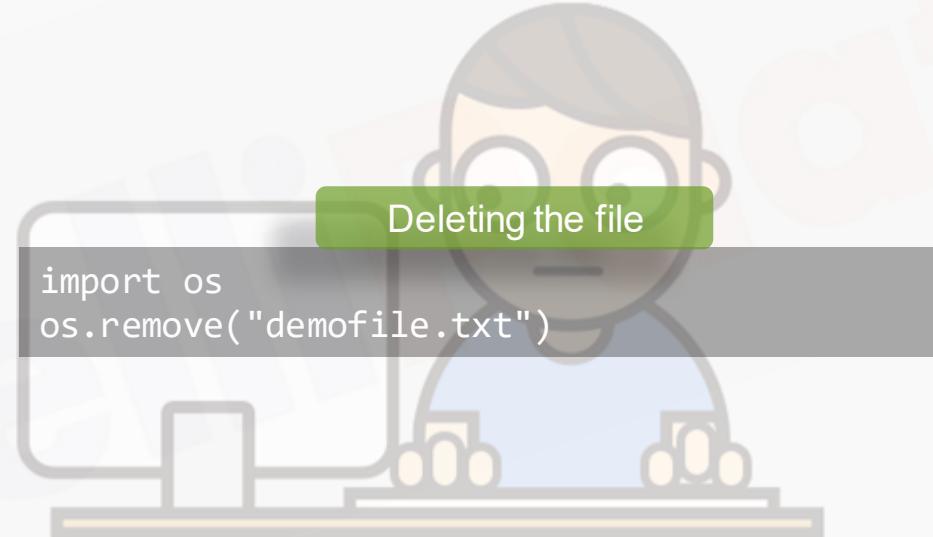
Write/Create

Delete

Import OS module to delete a file...

Deleting the file

```
import os  
os.remove("demofile.txt")
```



# QUIZ

# Quiz 1

Where did the Python name come from?

A

Python Snake

B

Monty Python's Circus Flying

C

Monty Python's Flying Circus

D

None of the above



# Answer 1

1. Where did the Python name come from?

A

a) Python Snake

B

a) Monty Python's Circus Flying

C

a) Monty Python's Flying Circus

D

a) None of the above



# Quiz 2

1. Which among the following can be an identifier in Python?

A

a)@HEll0

B

a)1hello

C

a)\${Hello1

D

a)\_Hello1



# Answer 2

1. Which among the following can be an identifier in Python?

A

a)@HEllo

B

a)1hello

C

a)\${Hello1

D

a)\_Hello1



# Quiz 3

1.What is the output of the following

2.4 | 6

A

a)4

B

a)6

C

a)10

D

a)1



# Answer 3

1.What is the output of the following

2.4 | 6

A

a)4

B

a)6

C

a)10

D

a)1



# Quiz 4

1.What is the output of the following

`10<<1`

A

a)10

B

a)1

C

a)20

D

a)None of the above



# Answer 4

1.What is the output of the following

`10<<1`

A

a)10

B

a)1

C

a)20

D

a)None of the above



# Quiz 5

1. Which of the following is going to throw an exception out of the following?

A

a) `A = 10`

B

a) `int A = 10`

C

a) `Name = "Noah"`

D

a) `Name = 'Noah'`



# Answer 5

1. Which of the following is going to throw an exception out of the following?

A

a) `A = 10`

B

a) `int A = 10`

C

a) `Name = "Noah"`

D

a) `Name = 'Noah'`



# Thank You



[www.intellipaat.com](http://www.intellipaat.com)



India : +91-7847955955

US : 1-800-216-8930 (TOLL FREE)

[sales@intellipaat.com](mailto:sales@intellipaat.com)