```
# Python Basics Day 02

# Operators
# Conditional Statements
# Looping Statements
# Functions
```

```
# Operators

# These are special symbols that are used to perform a specific operation.
# 1. Arithmatic operator: +, -, *, /, %, **
# 2. Assignment operator: =
# 3. Comparison operator: >,>=,<,<=,==,!=
# 4. Logical operator: and, or, not
# 5. Bitwise operator: &, |, ^
# 6. Identity operator: is, is not
# 7. Membership operator: in, not in
```

```
# 1. Arithmatic operator:

# + -> Addition
# - -> Substration
# * -> Multiplication
# / -> Division
# % -> Modulus -> Used to return the remainder of the division
# ** -> Exponential -> a**b -> a to the power of b

x = 5
y = 3

print(x+y) # 8
print(x-y) # 2
print(x*y) # 15
print(x/y) # 1.666
print(x%y) # 2
print(x**y) # 125

# c = x+y
# print(c)
```

```
8
2
15
1.6666666666666667
2
125
```

```
# 3. Comparison operator: >,>=,<,<=,==,!=

# -> The return type of comparison operator is always Boolean.

a = 5
b = 10

print(a > b) # False
print(a >= b) # False
print(a < b) # True
print(a <= b) # True
print(a == b) # Equality ---> False
print(a != b) # Not Equals ---> True
```

```
False
False
True
True
False
True
```

```
# 4. Logical operator:

# a number should be greater than 10 as well as a a even number ---> (a > 10 and a%2 ==0) ----> AND
```

```
# a number should be greater than 10 or a even number ---> (a > 10 or a%2 ==0) ----> OR

s1 = True
s2 = False

print(s1 and s2) # False
print(s1 or s2) # True
print(not s2) # True
```

```
False
True
True
```

```
# 6. Identity operator:

a = "Intellipaat"
b = "Python"
c = "Intellipaat"

print(a is b) # False
print(a is c)

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

```
False
True
138028879119472
138030211036592
138028879119472
```

```
# 7. Membership operator: in, not in

x = [1,2,3,4,5,6]
print(x)
y = 5

print(y in x) # True
print(y not in x) # False
```

```
[1, 2, 3, 4, 5, 6]
True
False
```

```
# Conditional Statements

# -> Statements are executed based on a condition.
# There are three types of conditional statments in Python:
# 1. simple if statement
# 2. if-else statement
# 3. if-elif-else statement
```

```
# 1. simple if statement

# SYNTAX for simple if statement
# if(condition):
#   Statements to be executed if the condition is True

# Check if the number 'm' is greater than 10 or not.
m = 4
if(m > 10):
  print("M is greater than 10")
```

```
# 2. if-else statement

# SYNTAX for if-else statement
# if(condition):
#   Statements to be executed if the condition is True
# else:
#   Statements to be executed if the condition is False.

```

```
# Check if the number 'm' is greater than 10 or not.
m = 15
if(m > 10):
  print("M is greater than 10")
else:
  print("M is not greater than 10")
```

```
M is greater than 10
```

```
# 3. if-elif-else statement

# SYNTAX for simle if-else statement
# if(condition):
#   Statements to be executed if the condition is True
# elif(condition):
#   Statements to be executed if the condition is True
# elif(condition):
#   Statements to be executed if the condition is True
# .
# .
# .
# .
# else:
#   Statements to be executed if the condition is False.

# Check if a number is greater than, less than or equals to 50
n = 50
if(n > 50):
  print("N is greater than 50")
elif(n < 50):
  print("N is less than 50")
elif(n == 50):
  print("N is equals to 50")
else:
  print("None of the conditions are True")

```

```
N is equals to 50
```

```
# Looping Statements

# They are used to perform the same task again and again.

# Print python 5000 times

print("Python")
print("Python")
print("Python")
print("Python")
print("Python")
```

```
Python
Python
Python
Python
Python
```

```
# There are two types of looping statements:
# 1. For Loop : used in cases wherein it involves range data.
# 2. While Loop: used in cases wherein we deal with conditional data.
```

```
# 1. For Loop : used in cases wherein it involves range data.

# range() -> inbuilt function which is used to create a range of defined values.
# SYNTAX:
# range(start, end) # NOTE: end is not included

# range(0,11) ----> 0,1,2,3,4,5,6,7,8,9,10

# SYNTAX for For-Loop:
# for iterating_variable in range(start,end):
```

```
#   task/statements

# iterating_variable ; usually defined as 'i' is used for traversing the range.

# Print python 5 times
for i in range(100,105):
  print("Python")
```

```
Python
Python
Python
Python
Python
```

```
# range(0,5) -> 0,1,2,3,4

# i   print("Python")
# 0   Python
# 1   Python
# 2   Python
# 3   Python
# 4   Python
```

```
# 2. While Loop: used in cases wherein we deal with conditional data.

# SYNTAX for While Loop:
# iterating_variable = start_value
# while(condition):
#   task/statements
#   iterating_variable = iterating_variable + 1

# Print python 5 times
i = 0
while(i < 5):
  print("Python")
  i = i + 1
```

```
Python
Python
Python
Python
Python
```

```
# i   while(i < 5)   print()    i = i + 1
# 0    TRUE          Python       1
# 1    TRUE          Python       2
# 2    TRUE          Python       3
# 3    TRUE          Python       4
# 4    TRUE          Python       5
# 5    FALSE ---------> Exit Condition.
```

```
# Functions/Methods

# These are some block of code which is used to perform a specific task.
# There are three types of functions:
# 1. Inbuilt Function
# 2. User Defined Function
# 3. Lambda Function
```

```
# 2. User Defined Function

# -> Which is used by the user to create or define a set of task.

# SYNTAX for defining a function
# def function_name(parameters/arguments):
#   task/statements

# SYNTAX to invoke a function(function call)
# function_name(value for parameters/arguments)
```

```
# Create a function that can print Python n number of times.
```

```

def printPython(n):
  for i in range(0,n):
    print("Python")
```

```
n= int(input())
printPython(n)
```

```
10
Python
Python
Python
Python
Python
Python
Python
Python
Python
Python
```

```
printPython(5)
```

```
Python
Python
Python
Python
Python
```

```
printPython(20)
```

```
Python
Python
Python
Python
Python
Python
Python
Python
Python
Python
Python
Python
Python
Python
Python
Python
Python
Python
Python
Python
```

```
# 3. Lambda Function

# function defination is stored within a variable.
# function call is made using the variable name
# only one line of code is allowed
# use the keyword lambda

# SYNTAX:
# variable_name = lambda parameters:tasks/statements
# Function Call
# variable_name(value for parameters)

# Add three numbers:
adder = lambda x,y,z:x+y+z

adder(10,50,20)
```

```
80
```

```
square = lambda a:a*a
square(4)
```

```
16
```

```
greater = lambda s,t:s if(s > t) else t
greater(10,20)
```

```
20
```

```
Start coding or generate with AI.
```