MyEduSolve x Kampus Merdeka INDONESIA JAYA

DATA ANALYST

#Future Ready: Future SkillS and Career Ready with MyEduSolve

August 2023

AGENDA

1. INTRODUCTION
2. LEARN
3. ASSIGMENT & TASK
4. QUIZ & WRAP UP
5. Q & A

INTRODUCTION

Subject Overview

Learning how to use Python operation, flow control and using modules to support Data analyst

Objectives & Learning Outcome

- **Understand how to using python operation**
- **Understand flow control**

UNIT PLANNING

Breakdown of curriculum

List of units/topics

- Evaluate expressions
- Perform Data and data type operation
- Determine the Sequences
- Select Operator
- Branching Statement
- Iteration

Key Concepts

What key concepts that want to be delivered in each meetings/units/topics

- Understand data types
- Understand operators
- Flow controls

Skills

What skills that students will gain during each meeting

- Logical Mathematics

MyEduSolve x Kampus Merdeka INDONESIA JAYA

PYTHON FOR DATA ANALYTICS

MyEduSolve

LEARNING MATERIALS

1. Operations using Data Types and Operators

2. Flow Control with Decisions and Loops

MyEduSolve x Kampus Merdeka INDONESIA JAYA

Operations using Data Types and Operators

Data types

Data types in python

(int, float, string, Boolean)

Int

Int or Integer is a data type for numeric objects in the form of positive and negative integers. For example 1,2,3 or -1,-2,-3

```
Num = 80
Print(Num)
  80
```

Float

Represents the floating-point number. Float is used to represent real numbers and is written with decimal point i.e 3.44

```
Fnum = 80.88
Print(Fnum)
  80.88
```

String

String contains sequence of characters. It represents by using single quotes or double quotes

```
Str = “80”       or    Str = “Myname”
Print(Str)             Print(Str)
  80                   > Myname
```

Boolean

Boolean is a true false statement, symbolized by 1 and 0 or True and False.

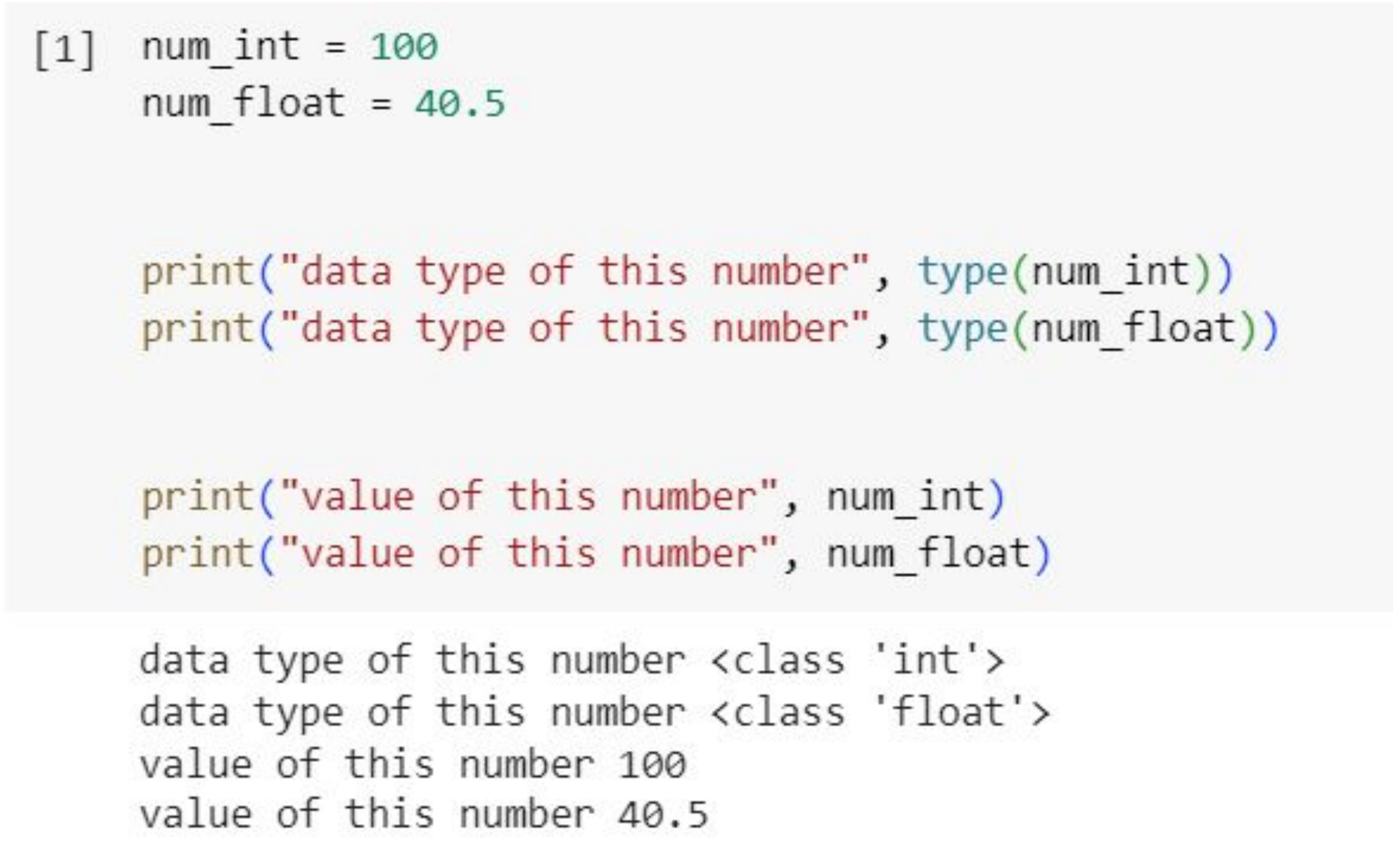
```
a = True
Print(a)
True
```

Data types operation

- **Implicit Type Conversion**
- **Explicit Type Conversion**

Implicit Type Conversion

In Implicit type conversion, python automatically converts on data type to another data type. This process doesn’t need any user involvement.



```
[1] num_int = 100
    num_float = 40.5


    print("data type of this number", type(num_int))
    print("data type of this number", type(num_float))


    print("value of this number", num_int)
    print("value of this number", num_float)
```

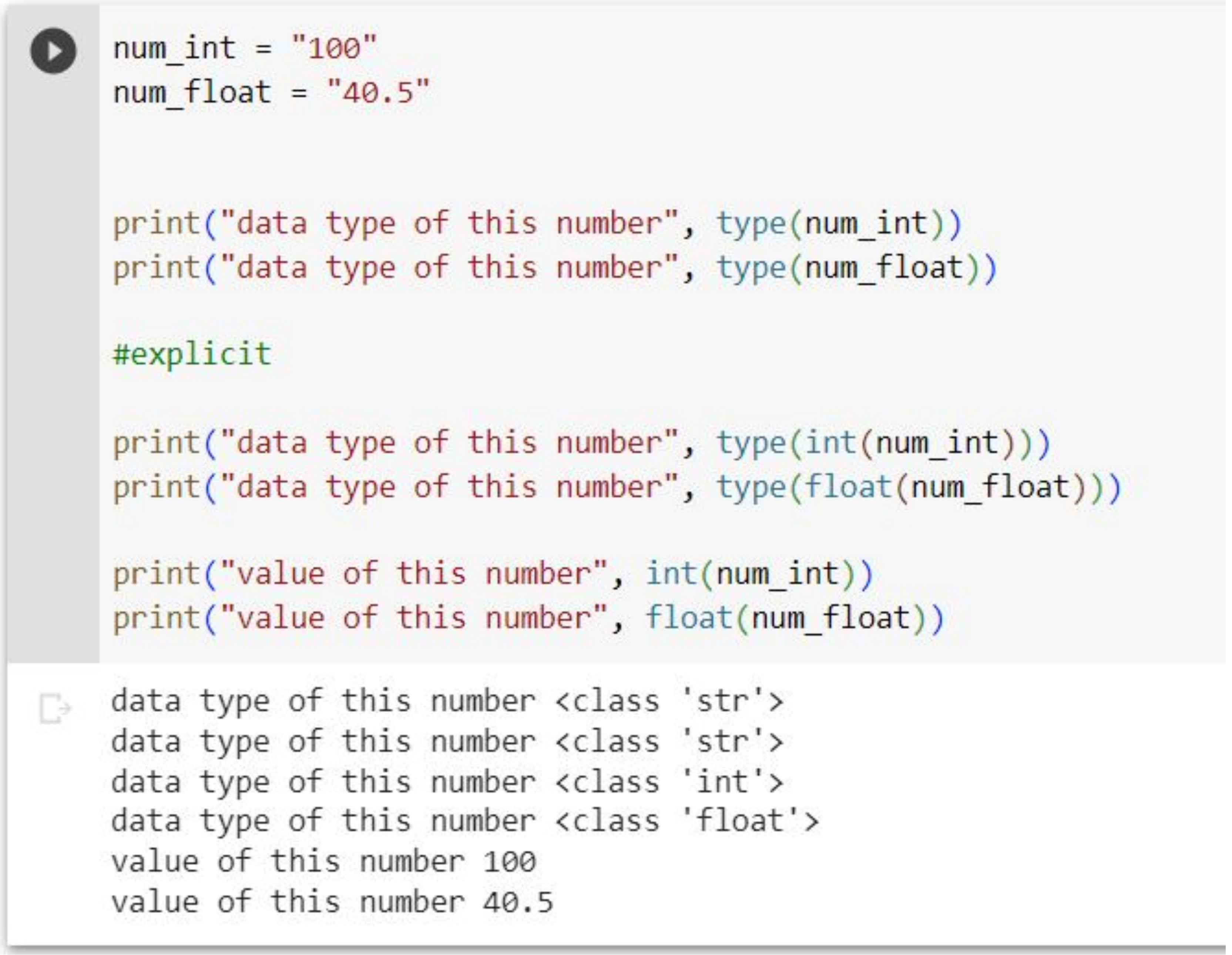
```
data type of this number <class 'int'>
data type of this number <class 'float'>
value of this number 100
value of this number 40.5
```

Data types operation

- **Implicit Type Conversion**
- **Explicit Type Conversion**

Explicit Type Conversion

In Explicit type conversion, users convert the data type of an object to required data type. We use the predefined function like int(), float(), str(). example

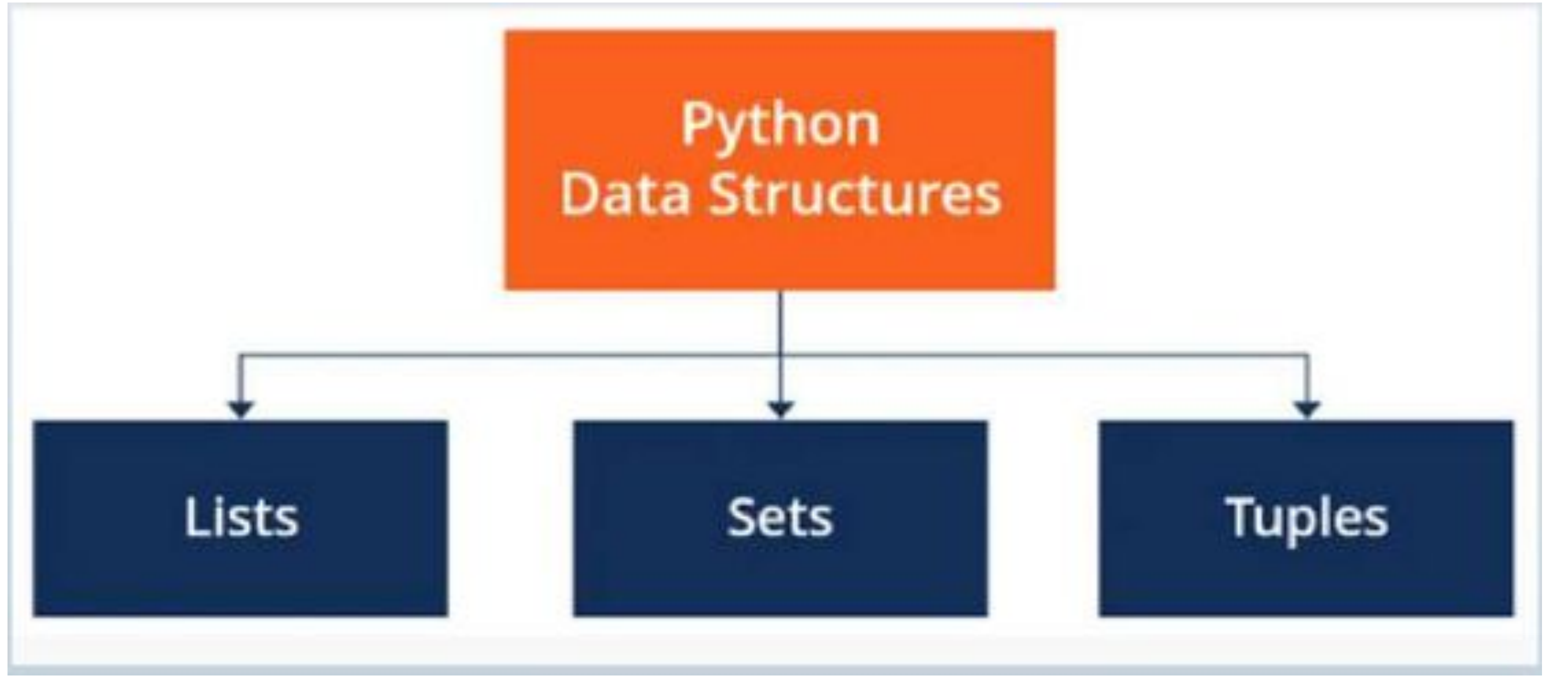


```python
num_int = "100"
num_float = "40.5"


print("data type of this number", type(num_int))
print("data type of this number", type(num_float))

#explicit

print("data type of this number", type(int(num_int)))
print("data type of this number", type(float(num_float)))

print("value of this number", int(num_int))
print("value of this number", float(num_float))
```

```
data type of this number <class 'str'>
data type of this number <class 'str'>
data type of this number <class 'int'>
data type of this number <class 'float'>
value of this number 100
value of this number 40.5
```

Python Data Structure

The Basic Python structures in Python include list, set , tuples and dictionary. Each of the data structures is in its own . Data structures are “containers” that organize and group data according to type.

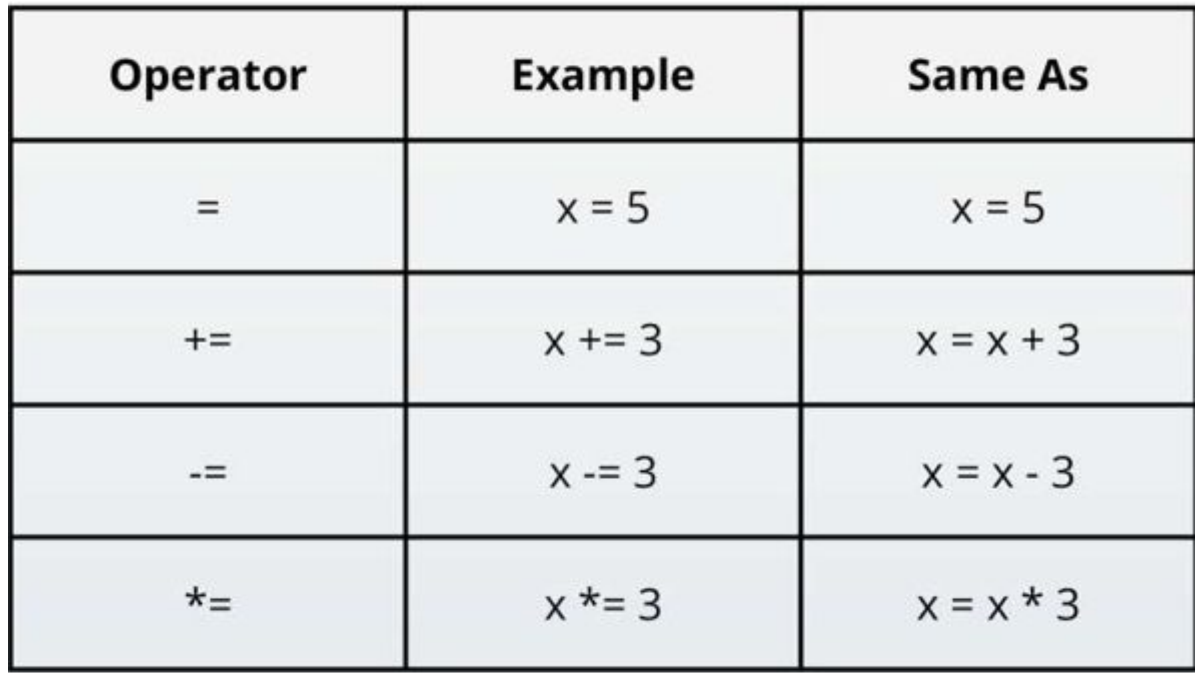


The data structures differ based on mutability and order. Mutability refers to the ability to change an object after its creation. Mutable objects can be modified, added, deleted after they have been created, while immutable objects cannot be modified after their creation. Order, in this context relates to whether the position of an element can be used to access the element.

Python Data Operators

Assignment Operators

An assignment statement evaluates the expression list(remember that this can be a single expression or a comma-separated list, the latter yielding a tuple) and assign the single result



Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3

Comparison Operators

Python has six comparison operators, which are as follows:

- Less Than (<)
- Less Than or equal to (<=)
- Greater Than (>)
- Greater Than Equal (>=)
- Equal to (==)
- Not Equal to (!=)

These Comparison operators compare two values and return Boolean value, either True or False.

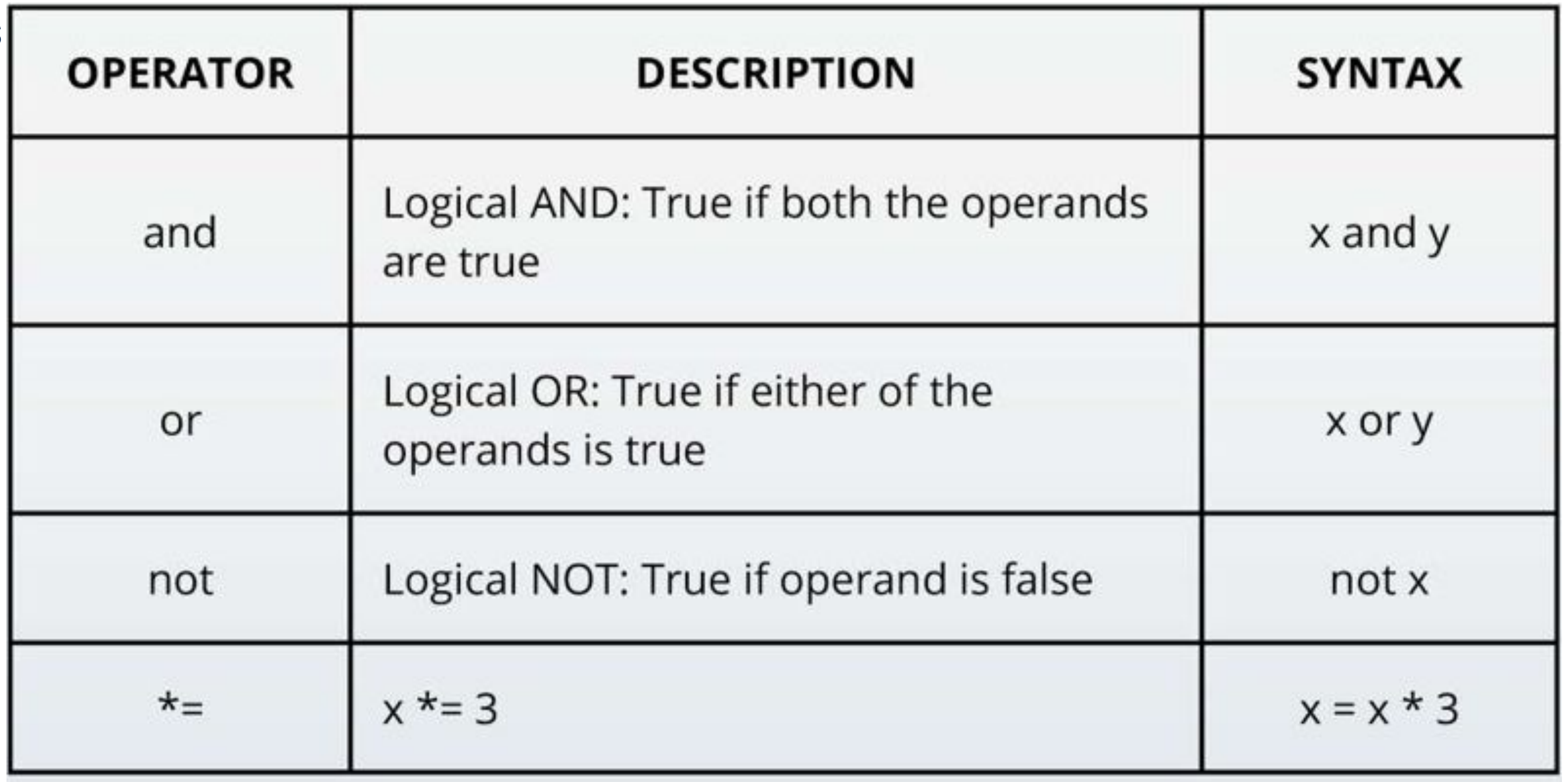
Python Data Operators

Logical Operators

Logical operators are used on conditional statements (either True or False). They perform Logical AND, logical OR, and Logical Not operators. This operator supports short-circuit evaluation, which mean that if the first argument is False the second is never evaluated.

Example Code

```
X = 2
Y = 1
IF X > 2 and Y<1:
   Print(True)
Else:
   Print(False)
```

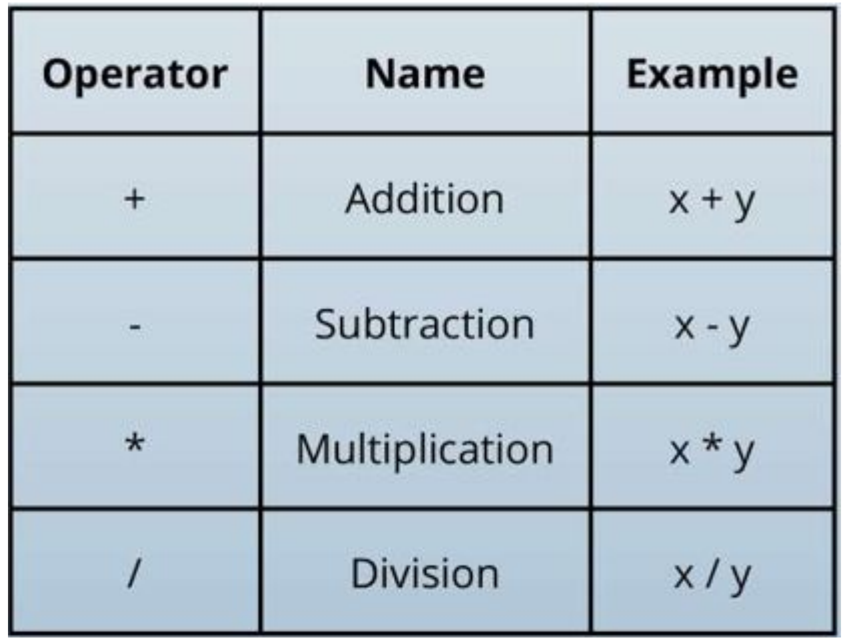


OPERATOR	DESCRIPTION	SYNTAX
and	Logical AND: True if both the operands are true	x and y
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if operand is false	not x
*=	x *= 3	x = x * 3

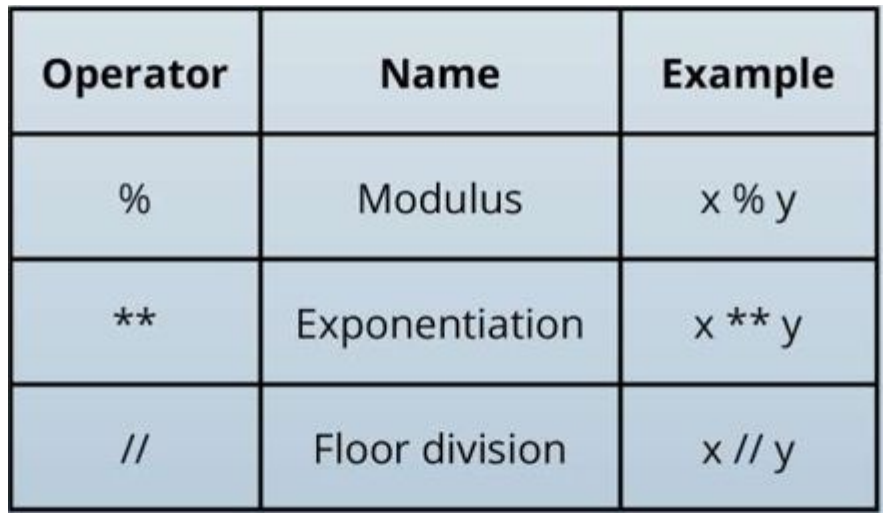
Python Data Operators

Assignment Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:



Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y



Operator	Name	Example
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Example Code

```
X = 2
Y = 1
Sum = X + Y
  3
Subs = X-Y
  1
```

HANDS ON SESSION

Google Collab

MyEduSolve x Kampus Merdeka INDONESIA JAYA

Flow Control with Decisions and Loops

Code Segments:

Branching

- IF Statement
- ELIF Statement
- ELSE Statement
- Nested IF Statement
- Compound I Conditional Statement

IF Statement

A selection statement that allows more than one possible flow of c

```
X = 2
Y = 1
IF X == 2 and Y ==1:
   Print(“Correct”)
  Correct
```

ELIF Statement

Elif is short for "else if" and is used when the first if statement isn't true, but you want to check for another condition

```
X = 2
IF X == 0:
   Print(“False”)
Elif X >= 2:
    Print(“Correct”)
  Correct
```

ELSE Statement

The ELSE statement specifies that alternate processing is to take place when the conditions of the matching IF state

```
Y = 1
IF  Y ==2:
   Print(“Correct”)
Else:
   Print(“FALSE”)
  FALSE
```

Code Segments: Branching

- **IF Statement**
- **ELIF Statement**
- **ELSE Statement**
- **Nested IF Statement**
- **Compound Conditional Statement**

NESTED IF Statement

Nested if is a decision-making statement that works similar to other decision-making statements such as if..else..if..else..etc

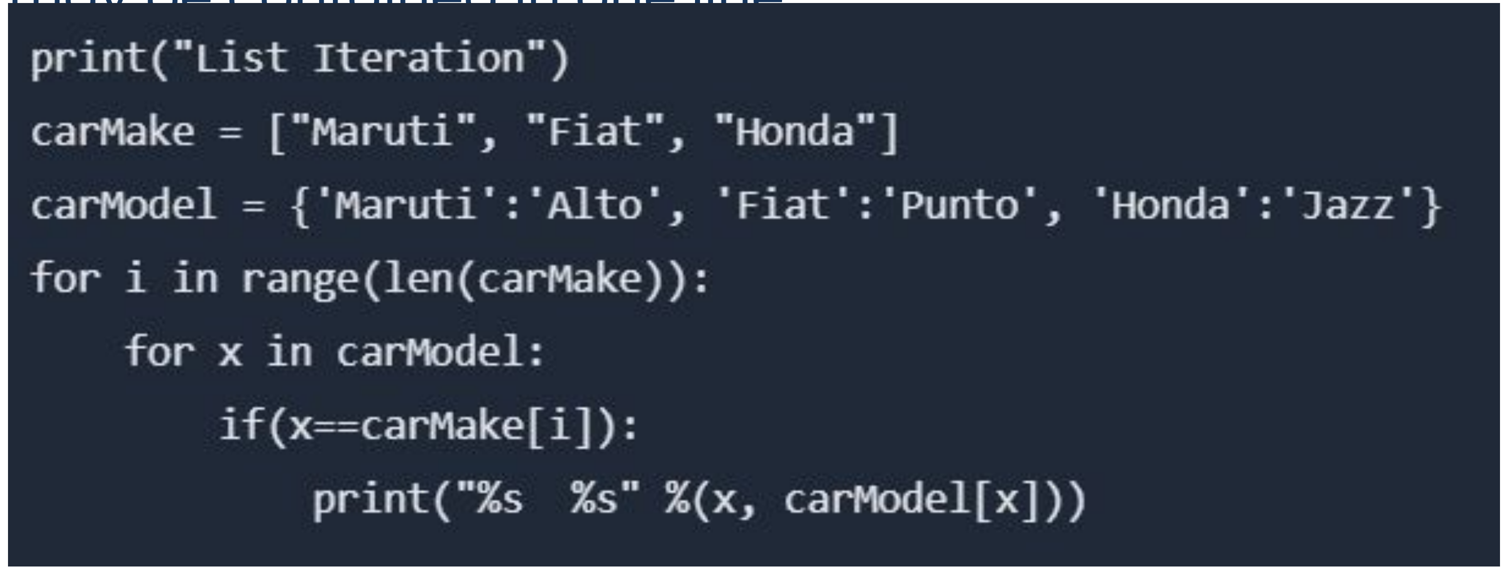
```
var = 100
if var <  200:
   print "Expression value is less than 200"
   if var == 150:
      print "Which is 150"
   elif var == 100:
      print "Which is 100"
   elif var == 50:
      print "Which is 50"
   elif var < 50:
      print "Expression value is less than 50"
else:
   print "Could not find true expression"
   Expression value is less than 200
   Which is 100
```

Code Segments: Branching

- **IF Statement**
- **ELIF Statement**
- **ELSE Statement**
- **Nested IF Statement**
- **Compound Conditional Statement**

Compound Conditional Statement

Compound statements contain (groups of) other statements; they affect or control the execution of those other statements in some way. In general, compound statements span multiple lines, although in simple incarnations a whole compound statement may be contained in one line.



```
print("List Iteration")
carMake = ["Maruti", "Fiat", "Honda"]
carModel = {'Maruti':'Alto', 'Fiat':'Punto', 'Honda':'Jazz'}
for i in range(len(carMake)):
    for x in carModel:
        if(x==carMake[i]):
            print("%s  %s" %(x, carModel[x]))
```

Code Segments: Iteration

- While
- For
- Break
- Continue
- Pass
- Nested Loop

While

The while statement is one of the control flow statements in C# that enables the execution of a sequence of logic multiple times in a loop until a specific condition is false

```
X = 1
While X < 5:
   Print(“Correct”)
    X += 1
```

For

The for statement lets you repeat a statement or compound statement a specified number of times

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

Code Segments: Iteration

- **While**
- **For**
- **Break**
- **Continue**
- **Pass**
- **Nested Loop**

Break

A break statement in Python alters the flow of a loop by terminating it once a specified condition is met

```
for i in range(5):
    if i == 3:
        break
    print(i)
```

Continue

The continue statement in Python is used to skip the remaining code inside a loop for the current iteration only

```
for i in range(5):
    if i == 3:
        continue
    print(i)
```

Code Segments: Iteration

- **While**
- **For**
- **Break**
- **Continue**
- **Pass**
- **Nested Loop**

Pass

The pass statement does nothing in Python, which is helpful for using as a placeholder in if statement branches, functions, and classes. In layman's terms, pass tells Python to skip this line and do nothing.

```
n = 10
# use pass inside if statement
if n > 10:
    pass
print('Hello')
```

Nested Loop

A nested loop refers to a loop within a loop, an inner loop within the body of an outer one.

```
x = [1, 2]
y = [4, 5]
for i in x:
  for j in y:
    print(i, j)
```

HANDS ON SESSION

Google Collab

MyEduSolve

GLOSSARY

NO	Terms	Definition
1	Indent	adding white space before a statement to a particular block of code
2	Indexing	the process of accessing an element in a sequence using its position in the sequence (its index)
3	Slicing	a feature that enables accessing parts of sequences like strings, tuples, and lists
4	Variable	a symbolic name that is a reference or pointer to an object
5	Infinity Loop	used to run a certain program or block of code an unspecified number of times

MyEduSolve

ASSIGNMENT/TASK

TASK 1

Create looping to identify the even and odd numbers

TASK 2

Create a loop to give a remarks of score. The remarks as follows:

A+ = 95-100

A=90-94

B+=85-89

B=80-84

C+=75-79

C= 70-74

Other score than that “Retake the course”

REFERRENCE / RESOURCES

https://www.w3schools.com/python/python_conditions.asp

https://www.w3schools.com/python/python_operators.asp

https://www.programiz.com/python-programming/operators

https://www.tutorialspoint.com/python/python_operators.htm

Quiz / Wrap Up Questions

Q & A
SESSION

Kampus Merdeka
INDONESIA JAYA

MyEduSolve