

BTEC Level 3 Computing

Unit 1 - Principles of Computer Science

Programming Paradigms

R o n s

T e c h

H u b

What are Programming Paradigms

- Use of standard structures and conventions to build and develop accurate, efficient and effective computer code to fulfill identified criteria and solve problems.
- In other words.
- Writing good computer code by using common coding practices to solve problems efficiently and correctly.

R O N S

T e c h

H u b

```
mirror_mod = modifier_ob.  
# Set mirror object to mirror  
mirror_mod.mirror_object  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True
```

```
#selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
print("please select exactly  
-- OPERATOR CLASSES --
```

```
types.Operator):  
on X mirror to the selected  
object.mirror_mirror_x"  
mirror X"
```

```
context):  
context.active_object is not
```


Handling data within a program

R o n s

T e c h

H u b

What does "Handling data within a program" mean?

R o n s

T e c h

H u b



I would label this as:



Date types.



Data structures.



On page 22 of the specification there is a list.



I will focus on the main data types and a few data structures.

R o n s

T e c h

H u b

Primitive Data Types



These are the most basic.



They help make up all the others you might come across in different programming languages.

Primitive Data Types

R O N S

T e c h

H u b

- Integers are whole numbers.
- Doubles have a decimal point.
- Booleans have a true or false value.
- Chars/Strings are alphanumeric characters.



R o n s

T e c h

H u b

Primitive Data Types Example

Integer.

24.

Double.

10.98.

Boolean.

True or False.

Chars/Strings.

"RonsTechHub".

Primitive Data Type Code – Python 3

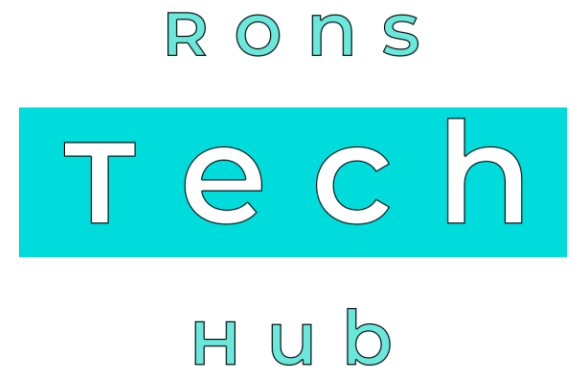
- # Primitive Data Types Example

```
# this is a string  
name = "RonTechHub"  
print("String:", name)
```

```
# this is an integer or whole number  
age = 51  
print("Integer:", age)
```

```
# this is a float or double  
height = 2.12345  
print("Float/Double:", height)
```

```
# this is a boolean  
male = True  
print("Boolean:", male)
```



What are Data Structures?

Data structures that are essential for storing, organising and manipulating data efficiently.

They are built into most programming languages.

Data Structures (four main Python)

Lists.

Tuples.

Sets.

Dictionaries.

Python List

R o n s

T e c h

H u b



```
numbers = [1, 2, 3, 4, 5]
```



```
fruits = ["apple", "banana",  
"cherry"]
```

Python Tuple

R o n s

Tech

H u b



```
numbers = (1, 2, 3, 4, 5)
```



```
fruits = ("apple", "banana",  
"cherry")
```



```
mixed_tuple = (1, "hello", 3.14,  
True)
```

Python Dictionary

R o n s

T e c h

H u b

- `person = {"name": "Alice", "age": 30, "city": "New York"}`
- `pairs = [('name', 'Charlie'), ('age', 40)]`

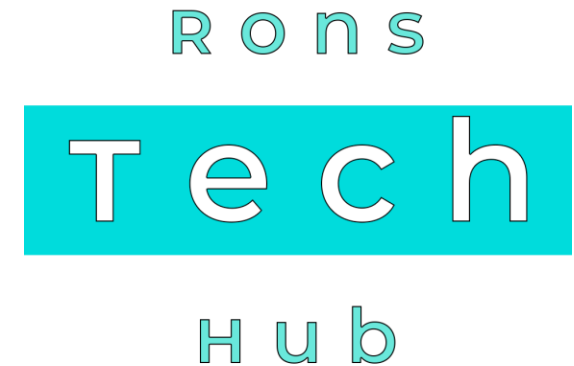
Data Structures Code – Python 3

- # List
- `my_list = [1, 2, 3, "apple", "banana"]`
- `print("List:", my_list)`

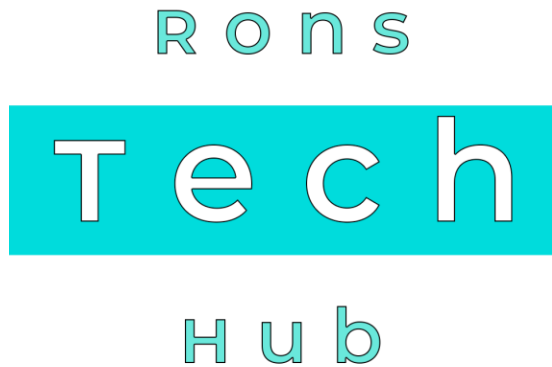
- # Tuple
- `my_tuple = (10, 20, "hello")`
- `print("Tuple:", my_tuple)`

- # Set
- `my_set = {5, 6, 7, 7, 8}` # Note: duplicate 7 is removed
- `print("Set:", my_set)`

- # Dictionary
- `my_dict = {"name": "Alice", "age": 30, "city": "New York"}`
- `print("Dictionary:", my_dict)`



Variables



- Local Variables and Global Variables.
- This is described as the scope of a variable.
- Local is only usable in that section of the code.
- Global can be used anywhere.
- This will make more sense when you see more examples.

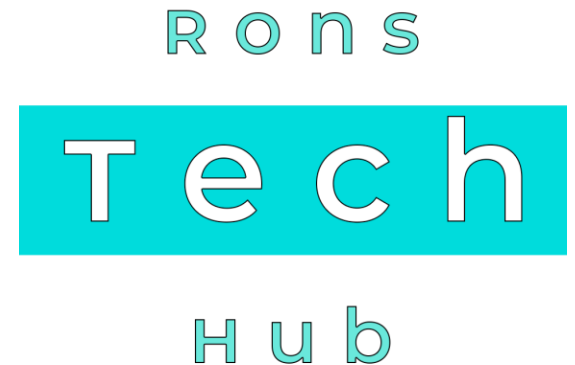
Python Variables Example

```
username = "RonsTechHub"    # username is the variable name

age = 15                     # age is the variable name

height = 1.56                # height is the variable name

dooraccess = True            # dooraccess is the variable name
```



Local Variable Example

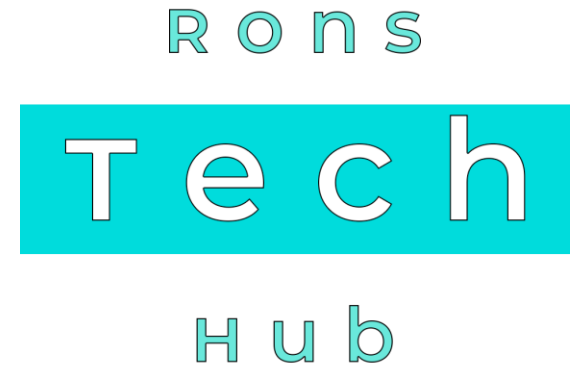
- `def my_function():`
- `local_variable = 10 # This is a local variable`
- `print("Inside the function:", local_variable)`

R O N S

T e c h

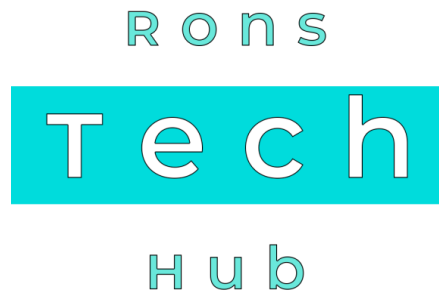
H u b

Global Variable Example



- `global_variable = 20 # This is a global variable`
- `def my_function():`
 - `print("Inside the function:", global_variable)`
- `def modify_global():`
 - `global global_variable # Needed to modify the global variable`
 - `global_variable = 30`
 - `print("Inside modify_global:", global_variable)`

Arithmetic/Math Operations



Addition.

Subtraction.

Multiplication.

Division.

Remainder Division/Modulo.

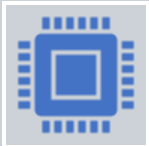
Programming is



Logical.



Math is logical.



The same operators and comparators we use in math we use in programming.

Arithmetic Operations – Python 3

- The code will be provided.
- ```
Addition
print(10 + 5) # Output: 15

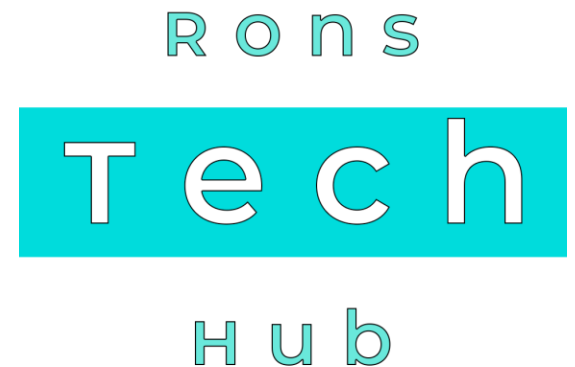
Subtraction
print(10 - 5) # Output: 5

Multiplication
print(10 * 5) # Output: 50

Division
print(10 / 5) # Output: 2.0 (Note: results in a float)

Integer Division (Floor Division)
print(10 // 3) # Output: 3

Remainder (Modulo)
print(10 % 3) # Output: 1
```





# Relational Operators

Relational operators are used in programming to compare two values or expressions.

These operators evaluate the relationship between the values and return a Boolean result (True or False).

R o n s

T e c h

H u b



# List of Operators (Python 3)



R O N S

T e c h

h u b



Equal to (==)

Not equal to  
(!=)

Not equal to  
(!=)

Less than (<)

Greater than  
or equal to  
(>=)

Less than or  
equal to (<=)

# Relational Operators (Python 3)

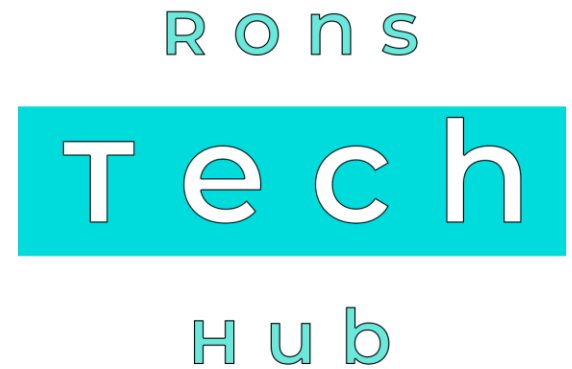
- Code provided.

- # Numerical comparisons

```
print(5 == 5) # True (equal to)
print(5 != 10) # True (not equal to)
print(10 > 5) # True (greater than)
print(5 < 10) # True (less than)
print(10 >= 10) # True (greater than or equal to)
print(5 <= 10) # True (less than or equal to)
```

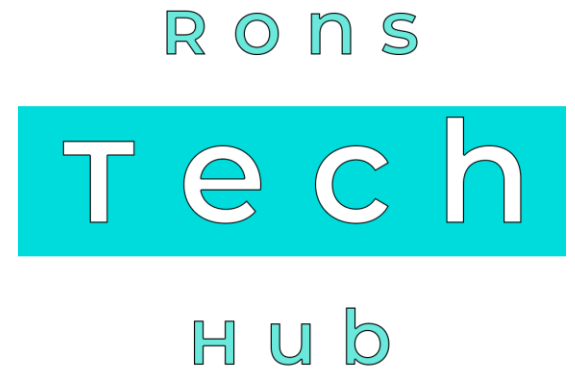
```
String comparisons (case-sensitive)
```

```
print("apple" == "apple") # True
print("apple" != "Apple") # True
print("apple" < "banana") # True (alphabetical order)
```

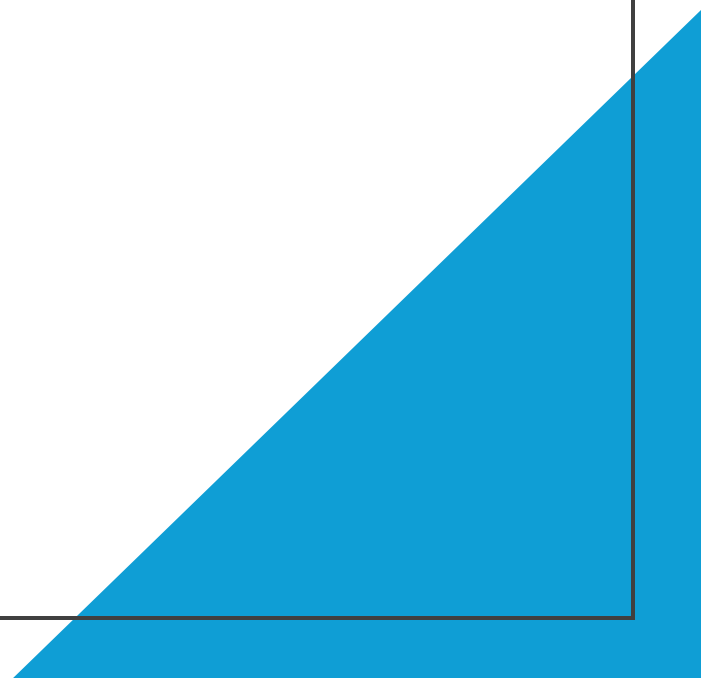




# Boolean Operators (NOT Boolean)



- Boolean operators are used to combine or modify Boolean values (True or False) in logical expressions.
- They help in decision-making in a program by evaluating conditions.
- The results will give a boolean value of True or False.



# List of Boolean Operators

R o n s

Tech

Hub

AND.

OR.

NOT.



## Boolean Operator "AND"

Returns True if **both** conditions are True.

Otherwise, it returns False.

# Boolean Operator "OR"

- Returns True if **at least one** of the conditions is True.
  - Returns False only if **both** conditions are False.
-

# Boolean Operator "NOT"

R O N S

T e c h

H u b



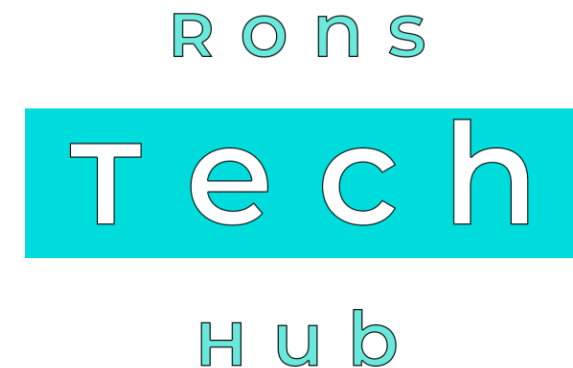
Reverses the value of a condition.



If the condition is True, not makes it False, and vice versa.

# Boolean Operators (Python 3)

- Code provided.
- # AND (both must be true)  
print(5 > 3 and 10 > 5) # True  
print(5 > 3 and 10 < 5) # False  
  
# OR (at least one must be true)  
print(5 > 3 or 10 < 5) # True  
print(5 < 3 or 10 < 5) # False  
  
# NOT (reverses the truth value)  
print(not 5 > 3) # False  
print(not 5 < 3) # True







# DateTime

R O N S



T e c h

H u b



Every single piece of meta data uses date and time.



Every single database uses date and time.



It is so important it has its own category, which makes it easier to track and manipulate.

# Date/Time (Python 3)

- Code provided as a file.
- `from datetime import date`  
`today = date.today()`  
`print(today) # e.g., 2025-01-18`

R o n s

T e c h

H u b





+  
o •

# Next Time

Built In Functions

R o n s

T e c h

H u b

+

• o