Introduction to Python

Agenda

- Python Fundamentals Quiz
- Variables, Operators, and Data Structures
- Conditional Statements
- Looping Statements
- Functions

Let's begin the discussion by answering a few questions on the fundamentals of Python programming

Python Fundamentals Quiz

Which of the following is true regarding variables in Python?

A Values assigned to variables can be modified

B Variables can store data structures such as arrays and dictionaries

C Variables can only store integer and floats

D Variables can store integer, floats, strings and booleans

Python Fundamentals Quiz

Which of the following is true regarding variables in Python?

A — Values assigned to variables can be modified

B — Variables can store data structures such as arrays and dictionaries

C — Variables can only store integer and floats

D — Variables can store integer, floats, strings and booleans

Variables

Used to **store any type of data**

Single values (integer, float, string, boolean, etc.)

Data structures (arrays, lists, dictionaries, etc.)

Can be **created by assigning a value to it with the "=" operator** (assignment operator)

`num = 100` => creates a variable *`num`* and stores the value *`100`* in it

Can be **modified to store a different value**

`num = 3.14` => variable *`num`* gets modified, now stores *`3.14`* instead of *`100`*

Python Fundamentals Quiz

Which of the following combinations accurately matches mathematical symbols with their respective operations?

A + for addition, - for subtraction

B * for exponentiation, / for modulus

C % for multiplication, ** for division

D ** for multiplication, % for subtraction

Python Fundamentals Quiz

Which of the following combinations accurately matches mathematical symbols with their respective operations?

[✓ A + for addition, - for subtraction]

B * for exponentiation, / for modulus

C % for multiplication, ** for division

D ** for multiplication, % for subtraction

Operators

Symbol	Operation	Example	Output
+, -	Addition, Subtraction	1 + 7 9 - 4	8 5
*, /, %	Multiplication, Division, Modulus	3 * 4 6 / 2 6 % 2	12 3.0 0
**	Exponentiation	2 ** 4	16
==, !=, >, >=, <, <=	Comparison	5 <= 4	False
in, not in	Membership	5 in [1, 2, 3, 4, 5]	True
()	Grouping	(1+2) * (2+3)	3 * 5 = 15

Python Fundamentals Quiz

Consider a customer database for managing Netflix subscriptions. Each customer has a unique ID, name, email, and subscription type, which remain constant once registered.

What data structure would be the best choice to store this information?

A Only List

B Only Tuple

C List and Tuple

D List and Dictionary

Python Fundamentals Quiz

Consider a customer database for managing Netflix subscriptions. Each customer has a unique ID, name, email, and subscription type, which remain constant once registered.

What data structure would be the best choice to store this information?

- A Only List
- [✓ B Only Tuple]
- C List and Tuple
- D List and Dictionary

Data Structures

List	Tuple	Dictionaries
A collection of items of any data type	A collection of items of any data type	A collection of key-value pairs
Mutable (can be modified)	Immutable (cannot be modified)	Mutable (can be modified)
Syntax: `mylist = ["Element 1", "Element 2", "Element 3"]`	Syntax: `mytuple = ("Element 1", "Element 2", "Element 3")`	Syntax: `mydict = {1: 'Element 1', 2: 'Element 2', 3: 'Elements 3'}`
Example: `X=["a", 2, True, "b"]`	Example: `X=("a", 2, True, "b")`	Example: `X={1:'Jan', 2:'Feb', 3:'Mar'}`

Python Fundamentals Quiz

Which of the following will retrieve the two middle elements from the list?

```
my_list = [1, "two", 3.5, "four", 5, 6.0, "seven", 8.2]
```

A `my_list[3:5]`

B `my_list[4:6]`

C `my_list[-5:-3]`

D `my_list[-4:-6]`

Python Fundamentals Quiz

Which of the following will retrieve the two middle elements from the list?

```
my_list = [1, "two", 3.5, "four", 5, 6.0, "seven", 8.2]
```

A `my_list[3:5]`

B `my_list[4:6]`

C `my_list[-5:-3]`

D `my_list[-4:-6]`

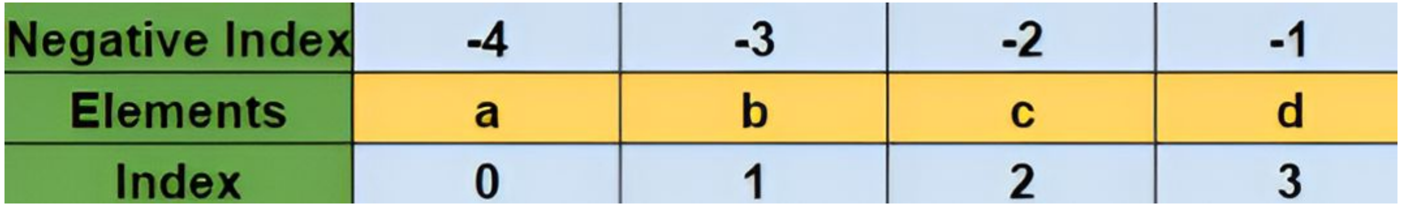
Lists - Slicing

To retrieve a specific **subset of elements from a list**, one can **slice the list by index**

The format for list slicing is as follows:

```
<list_name>[Start Index : End Index : Index-Jump]
```

Consider a list `my_list = ['a', 'b', 'c', 'd']`



Negative Index	-4	-3	-2	-1
Elements	a	b	c	d
Index	0	1	2	3

Python Fundamentals Quiz

A supermarket offers discounts as per a customer's purchase amount based on the following rules:

1. If the total purchase is $5000 or more, the discount is 20%.
2. If the total purchase is between $3000 (incl.) and $5000 (excl.), the discount is 15%.
3. If the total purchase is less than $3000, the discount is 10%.

Which logical operator should be used in the missing spaces (___) to correctly implement the discount logic?

```
if total_purchase ___ 5000:
    discount = 0.20
elif total_purchase ___ 3000:
    discount = 0.15
else:
    discount = 0.10
```

A `>= , >`

B `> , >=`

C `>= , >=`

D `> , >`

Python Fundamentals Quiz

A supermarket offers discounts as per a customer's purchase amount based on the following rules:

1. If the total purchase is $5000 or more, the discount is 20%.
2. If the total purchase is between $3000 (incl.) and $5000 (excl.), the discount is 15%.
3. If the total purchase is less than $3000, the discount is 10%.

Which logical operator should be used in the missing spaces (___) to correctly implement the discount logic?

```
if total_purchase ___ 5000:
    discount = 0.20
elif total_purchase ___ 3000:
    discount = 0.15
else:
    discount = 0.10
```

A >= , >

B > , >=

C [✓ >= , >=]

D > , >

Conditional Statements

Used to **make decisions based on specified rules**

A **single decision** can be made using **`if-else` construct**

```
if (test expression):
    <Body of if> this is executed if the expression is True
else:
    <Body of else> this is executed if the expression is False
```

More than one decision can be made using the **`if-elif-else` construct**

```
if (test expression 1):
    <Body of if> run this if test expression 1 is True
elif (test expression 2):
    <Body of elif> run this if test expression 2 is True
else:
    <Body of else> run this if none of the above expressions are True
```

Python Fundamentals Quiz

Which of the following statements is true regarding loops in Python?

A The "for" loop requires a condition to be evaluated before execution

B The "for" loop iterates through a sequence, executing on each element

C The "while" loop requires a condition to be evaluated before execution

D The "while" loop iterates through a sequence, executing on each element

Python Fundamentals Quiz

Which of the following statements is true regarding loops in Python?

- A The "for" loop requires a condition to be evaluated before execution
- [✓ B] The "for" loop iterates through a sequence, executing on each element
- [✓ C] The "while" loop requires a condition to be evaluated before execution
- D The "while" loop iterates through a sequence, executing on each element

Looping Statements

Used to **repeat a single statement or a set of statements**

Looping statement	Syntax	Example	Output

`for`	`for iter_var in seq:`
	`    statements(s)`
		`for i in range(1, 5):`
		`    print(i)`
			Prints all integers from 1 till 5 (excluded)

`while`	`while condition:`
	`    statement(s)`
		`i = 1`
		`while i < 5:`
		`    print(i)`
		`    i += 1`
			Prints all integers from 1 till 5 (excluded)

Python Fundamentals Quiz

What is the purpose of functions in Python?

A To execute a specific task only once in a program

B To combine multiple variables into a single variable

C To break code into modular chunks for reusability and organization

D To declare built-in variables that are predefined in Python

Python Fundamentals Quiz

What is the purpose of functions in Python?

- A To execute a specific task only once in a program
- B To combine multiple variables into a single variable
- [✓ C To break code into modular chunks for reusability and organization]
- D To declare built-in variables that are predefined in Python

Functions

Block of instructions that performs a specific task

Break code into **modular chunks** which can be reused **reused later**

Makes the code more organized and manageable

There are two types of functions in Python

Built-in Functions: Pre-defined in Python (`print()`, `len()`, `sum()`, etc)

User-defined Functions: Defined by users to perform a specific task

Python Fundamentals Quiz

The total amount payable at a supermarket after applying a relevant discount based on a customer's total purchase amount is defined by the following function:

```
def calculate_amount_payable(total_purchase):
    if total_purchase >= 5000:
        discount = 0.20
    elif total_purchase >= 3000:
        discount = 0.15
    else:
        discount = 0.10
    return total_purchase*(1 - discount)
```

Which of the following is the correct way to call this function and store the final price in a variable?

A
```
final_price =
calculate_amount_payable(4000)
```

B
```
final_price =
calculate_amount_payable()
```

C
```
final_price =
calculate_amount_payable(4000, 15)
```

D
```
calculate_amount_payable =
final_price(4000)
```

Python Fundamentals Quiz

The total amount payable at a supermarket after applying a relevant discount based on a customer's total purchase amount is defined by the following function:

```
def calculate_amount_payable(total_purchase):
    if total_purchase >= 5000:
        discount = 0.20
    elif total_purchase >= 3000:
        discount = 0.15
    else:
        discount = 0.10
    return total_purchase*(1 - discount)
```

Which of the following is the correct way to call this function and store the final price in a variable?

A `final_price = calculate_amount_payable(4000)`

B `final_price = calculate_amount_payable()`

C `final_price = calculate_amount_payable(4000, 15)`

D `calculate_amount_payable = final_price(4000)`

User-defined Functions

Syntax for a user defined function:

```
def function_name(parameters):
    statement(s)
    return statement
```

The `return` **statement is optional** and can be used when one or more values have to be returned by the function

A function can have **multiple return statements**

One can **store the output of a function** by **assigning it to a variable**

```
variable_name = function_name(parameters)
```

Great Learning

Happy Learning !