Module 1

1. What are the features of python?

Python is gaining popularity in the programming community; there are many reasons behind this.

- **Interpreted Language**: Python is processed at runtime by Python Interpreter.
- **Object-Oriented Language**: It supports object-oriented features and techniques of programming.
- **Interactive Programming Language**: Users can directly interact with the Python interpreter to write programs.
- **Easy language**: Python is simple to learn, particularly for newcomers.
- **Straightforward Syntax**: The formation of Python syntax is simple, making it popular.
- **Easy to read**: Python source code is clearly defined and visible.
- **Portable:** Python codes can be run on various hardware platforms with the same interface.
- **Extendable**: Users can add low level-modules to the Python interpreter.
- **Scalable**: Python provides an improved structure for supporting large programs than shell scripts.

2. What are the executions modes in Python?

Python uses an interpreter for the execution of source code. There are two ways in which we can use the interpreter. They are as follows:

- Interactive Mode
- Script Mode

Interactive Mode

In this mode, we can execute a single statement at a time. Moreover, to use the interactive mode, we have to write the statement in front of '>>>' and press enter

This results in the output of that particular statement immediately. This mode is easy and convenient to use to see the instant output. But, at the same time, we cannot save the whole code and have to write it again and again to execute it.

Script Mode

In this mode we have to write the whole source code and save it as a Python source code file. Furthermore, we can execute this file using the interpreter. Moreover, we save the python source code file with the extension '.py'.

3. What is Python? Why Python is preferred to other languages?

Python is an advanced, interpreted programming language known for its readability and simplicity. Python is a programming language that includes features of C and Java. It provides the style of writing elegant code like C, and for object-oriented programming, it offers classes and objects like Java.

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.

- Python runs on an interpreter system, meaning that code can be executed as soon as it is written.
- Python can be treated in a procedural way, an object-oriented way or a functional way.
- Used for AI applications

4. What are the various areas where python can be used?

- Server side scripting for Web Applications
- As a software language
- Can be connected to the database and thus it will be used as a server side scripting for dynamic websites.
- It can handle big data and perform complex mathematics.
- It can be used for various statistical applications
- Used for AI applications
- Researchers for various analysis
- Rapid prototyping, or for production-ready software development.

5. What is python identifier? What are the rules for python identifier?

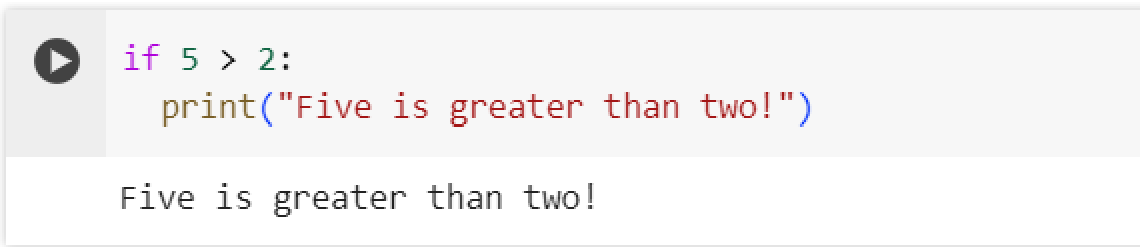
All the variables, class, object, functions, lists, dictionaries etc. in Python are together termed as Python Identifiers. Identifiers are the basis of any Python program. Almost every Python Code uses some or other identifiers.

Rules for using Python Identifiers:

- An identifier name should not be a keyword.
- An identifier name can begin with a letter or an underscore only.
- An identifier name can contain both numbers and letters along with underscores (A-z, 0-9, and _).
- An identifier name in Python is case-sensitive i.e, sum and Sum are two different identifier.

6. What is indentation in python? Explain with a example.

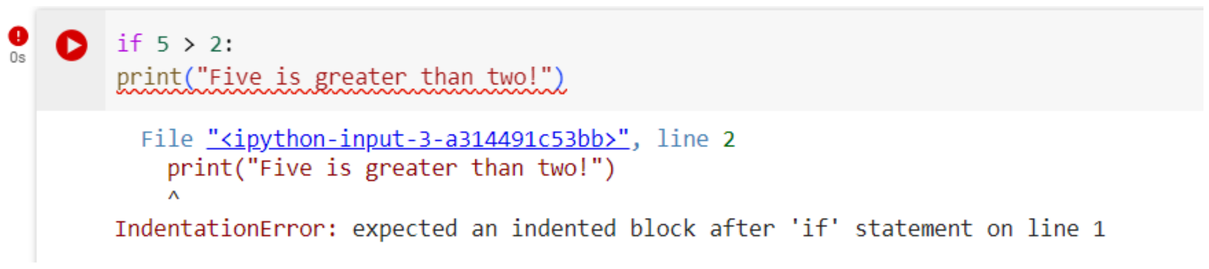
Indentation refers to the spaces at the beginning of a code line.
Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.
Python uses indentation to indicate a block of code.



```
if 5 > 2:
  print("Five is greater than two!")
```

```
Five is greater than two!
```

Python will give you an error if you skip the indentation:



```
if 5 > 2:
print("Five is greater than two!")
```

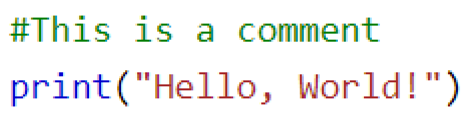
```
  File "<ipython-input-3-a314491c53bb>", line 2
    print("Five is greater than two!")
    ^
IndentationError: expected an indented block after 'if' statement on line 1
```

7. **What is comment in python? How to create single line comment and multiline comments?**

Comments can be used to explain Python code.
Comments can be used to make the code more readable.
Comments can be used to prevent execution when testing code.

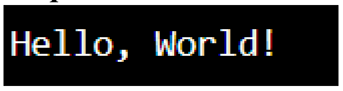
Creating a Comment

Comments starts with a #, and Python will ignore them:



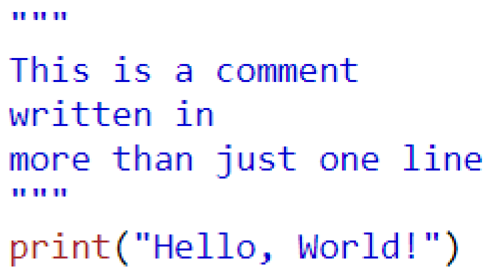
```
#This is a comment
print("Hello, World!")
```

Output

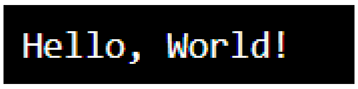


```
Hello, World!
```

Multiline comments:



```
"""
This is a comment
written in
more than just one line
"""
print("Hello, World!")
```



```
Hello, World!
```

8. **Give some examples for reserved keywords.(Any 8)**

Python has a set of keywords that are reserved words that cannot be used as variable names, function names, or any other identifiers:

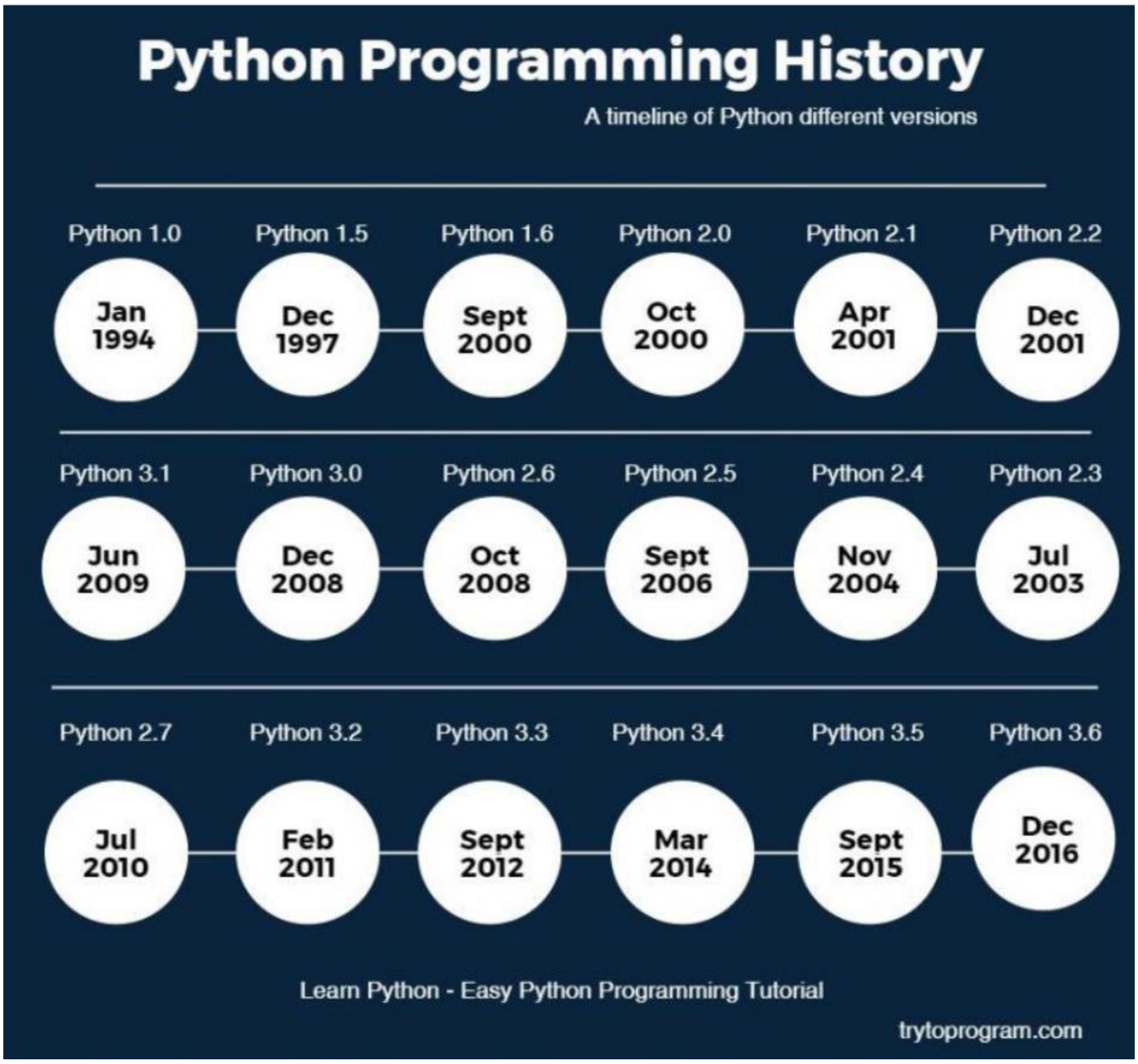
Keyword	**Description**
and	A logical operator
as	To create an alias
assert	For debugging
break	To break out of a loop
class	To define a class
continue	To continue to the next iteration of a loop
def	To define a function
del	To delete an object
elif	Used in conditional statements, same as else if
else	Used in conditional statements
except	Used with exceptions, what to do when an exception occurs

False	Boolean value, result of comparison operations
finally	Used with exceptions, a block of code that will be executed no matter if there is an
for	To create a for loop
from	To import specific parts of a module
global	To declare a global variable
if	To make a conditional statement
import	To import a module
in	To check if a value is present in a list, tuple, etc.
is	To test if two variables are equal
lambda	To create an anonymous function
None	Represents a null value
nonlocal	To declare a non-local variable

not	A logical operator
or	A logical operator
pass	A null statement, a statement that will do nothing
raise	To raise an exception
return	To exit a function and return a value
True	Boolean value, result of comparison operations
try	To make a try...except statement
while	To create a while loop
With	Used to simplify exception handling
Yield	To end a function, returns a generator

9. Explain evolution of python.

The first ever version of Python(i.e. Python 1.0) was introduced in 1991. Since its inception and introduction of Version 1, the evolution of Python has reached up to Version 3.x (till 2017). Here is the brief chart depicting the timeline of the release of different versions of Python programming language.



10.How to use blank lines in python?

The print() function in Python is often used to display output on the screen. It can also be used to print an empty line.

The simplest way to do this is to call the print() function without arguments.

Code	Output
print()	

The code above calls the print() function with no arguments, resulting in an empty line being printed to the console.
This method is particularly useful if you need to include an empty line between two lines of text:

Code	Output
print("Hello, World!")	Hello, World!
print()	
print("Goodbye, World!")	Goodbye, World!

The code above will print "Hello, World!" on one line, an empty line on the next, and then "Goodbye, World!" on the following line.

11.Explain quotation in python.

Python string functions are very popular. There are two ways to represent strings in python. String is enclosed either with single quotes or double quotes. Both the ways (single or double quotes) are correct depending upon the requirement. Sometimes we have to use quotes (single or double quotes) together in the same string, in such cases, we use single and double quotes alternatively so that they can be distinguished.

```
#Gives Error
print('It's python')
```

Output:

```
    print('It's python')
                 ^
SyntaxError: invalid syntax
```

Explanation – It gives an invalid syntax error. Because a single quote after "it" is considered the end of the string and rest part is not part of a string. It can be corrected as:

```
print("It's Python !")
```

Output:

```
It's Python!
```

12.How to run python program?

1st way:

Google colab:

1. Search for google colab
2. Register and sign in
3. Open new notebook
4. Type the code and click on debug button

2nd way:

Using Text Based:

If you find that you do not have Python installed on your computer, then you can download it for free from the following website: https://www.python.org/

To Run the Python program in a command prompt

```
C:\Users\Your Name>python helloworld.py
```

```
Hello, World!
```

Using IDE:

```
C:\Users\Your Name>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
```

Multiple choice questions:

1. What is the purpose of comments in Python?
   a) To execute certain code blocks conditionally
   b) To document and explain code for better understanding
   c) To disable certain lines of code temporarily
   d) To declare variables and functions
   Correct answer: b) To document and explain code for better understanding
2. Which symbol is used to start a single-line comment in Python?
   a) //
   b) –
   c) #
   d) '''
   Correct answer: c) #
3. Which of the following statements about comments in Python is true?
   a) Comments are executed by the Python interpreter
   b) Comments can span multiple lines using // symbols
   c) Comments are ignored by the Python interpreter during execution
   d) Comments can only be used for function documentation
   Correct answer: c) Comments are ignored by the Python interpreter during execution
4. Which of the following is a correct way to write a multi-line comment in Python?
   a) /* This is a multi-line comment */
   b) <!-- This is a multi-line comment -->
   c) ''' This is a multi-line comment '''
   d) // This is a multi-line comment //
   Correct answer: c) ''' This is a multi-line comment '''
5. Comments in Python are useful for:
   a) Writing code that runs faster
   b) Hiding code from other programmers
   c) Providing additional information and explanations to the reader
   d) Creating conditional statements
   Correct answer: c) Providing additional information and explanations to the reader
6. In Python, how is a block of code defined?
   a) Using braces **{ }**
   b) Using indentation
   c) Using semicolons **;**
   d) Using keywords like "begin" and "end"
   Correct answer: b) Using indentation
7. Which of the following statements is true regarding indentation in Python?
   a) Indentation is optional in Python
   b) Indentation must always be consistent, but the number of spaces or tabs doesn't matter
   c) Indentation is used to define the beginning and end of code blocks
   d) Indentation in Python is purely for aesthetic purposes
   Correct answer: c) Indentation is used to define the beginning and end of code blocks

8. Which of the following statements is true about indentation in Python?
   a) Indentation doesn't affect the behavior of the code
   b) Indentation can be skipped if the code is simple
   c) Incorrect indentation can lead to logical errors in the code
   d) Python provides automatic indentation for better code readability
   Correct answer: c) Incorrect indentation can lead to logical errors in the code
9. Which of the following options correctly defines Python reserved keywords?
   a) Words that are reserved for future Python versions
   b) Words that are predefined and reserved for specific purposes in Python
   c) Words that are reserved for Python libraries and modules
   d) Words that cannot be used as identifiers in Python
   Correct answer: b) Words that are predefined and reserved for specific purposes in Python
10. Which of the following is not a reserved keyword in Python?
    a) pass
    b) module
    c) class
    d) break
    Correct answer: b) module
11. What happens if you try to use a reserved keyword as a variable name in Python?
    a) Python raises a SyntaxError
    b) The keyword will be automatically converted to a string
    c) It's allowed, but not recommended
    d) Python ignores the reserved status of the keyword in this context
    Correct answer: a) Python raises a SyntaxError
12. What is the purpose of reserved keywords in Python?
    a) To provide syntactic sugar for common programming tasks
    b) To define built-in functions and methods
    c) To mark special language constructs and prevent their use as identifiers
    d) To reserve memory space for critical operations
    Correct answer: c) To mark special language constructs and prevent their use as identifiers
13. Python is known for its:
    a) Simplicity and readability
    b) Complexity and verbosity
    c) High performance and speed
    d) Low-level system programming capabilities
    Correct answer: a) Simplicity and readability
14. What is the primary philosophy behind Python's design?
    a) Optimize code for performance
    b) Prioritize simplicity and readability
    c) Emphasize low-level hardware access
    d) Enable rapid development with minimal documentation
    Correct answer: b) Prioritize simplicity and readability

15. What is the purpose of Python's standard library?
    a) To provide a collection of third-party modules
    b) To include advanced features not available in the core language
    c) To extend the functionality of the Python interpreter
    d) To provide a collection of modules and packages for common tasks
    Correct answer: d) To provide a collection of modules and packages for common tasks
16. Which major company has played a significant role in the development and promotion of Python?
    a) Google
    b) Microsoft
    c) Apple
    d) IBM
    Correct answer: a) Google
17. What is one of the benefits of Python's readability?
    a) It makes debugging easier
    b) It improves code performance
    c) It reduces the need for comments
    d) It speeds up code execution
    Correct answer: a) It makes debugging easier
18. Which command is used to execute a Python script?
    a) python -i script.py
    b) python script.py
    c) python -c script.py
    d) python -e script.py
    Correct answer: b) python script.py
19. Which of the following is not a valid way to execute a Python script?
    a) Using an Integrated Development Environment (IDE)
    b) Using a text editor and a command prompt
    c) Using a web browser
    d) Using an online Python compiler
    Correct answer: c) Using a web browser
20. What happens when you execute a Python script in immediate mode?
    a) The script runs silently without any output
    b) The script is executed line by line, and the results are displayed immediately
    c) The script is executed in the background
    d) The script executes with additional debugging information
    Correct answer: b) The script is executed line by line, and the results are displayed immediately
21. What is the purpose of the Python interpreter?
    a) To convert Python code into machine code
    b) To execute Python code and produce output
    c) To optimize Python code for better performance
    d) To analyze Python code for errors and warnings
    Correct answer: b) To execute Python code and produce output

22. In Python, which of the following control structures rely on indentation?
    a) If statements
    b) While loops
    c) For loops
    d) All of the above

    Correct answer: d) All of the above
23. What error will you encounter if there's an indentation mismatch in your Python code?

    a) SyntaxError
    b) IndentationError
    c) RuntimeError
    d) ValueError

    Correct answer: b) IndentationError
24. Which of the following is not a reserved keyword in Python?
    a) If
    b) then
    c) else
    d) elif

    Correct answer: b) then
25. In Python, which keyword is used to define a loop?

    a) for
    b) loop
    c) iterate
    d) do

    Correct answer: a) for

Module 2

1. What is a variable? How to create variable? What are the rules for declaring variables?

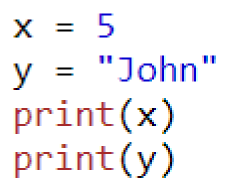
Variables

Variables are containers for storing data values.

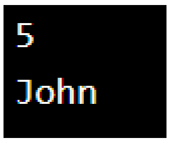
Creating variables

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.



```
x = 5
y = "John"
print(x)
print(y)
```



```
5
John
```

Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for Python variables:

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)
- A variable name cannot be any of the Python keywords.

2. What are strings? What are the functions associated with the string?

Strings in python are surrounded by either single quotation marks, or double quotation marks.

String Length

To get the length of a string, use the len() function.

Example

The len() function returns the length of a string:

```
a = "Hello, World!"
print(len(a))
```

```
13
```

Check String

To check if a certain phrase or character is present in a string, we can use the keyword in.

Use it in an if statement:

Example

Print only if "free" is present:

```
txt = "The best things in life are free!"
if "free" in txt:
  print("Yes, 'free' is present.")
```

```
Yes, 'free' is present.
```

Check if NOT

To check if a certain phrase or character is NOT present in a string, we can use the keyword not in.

Example

print only if "expensive" is NOT present:

```
txt = "The best things in life are free!"
if "expensive" not in txt:
  print("No, 'expensive' is NOT present.")
```

```
No, 'expensive' is NOT present.
```

3. **What is list? What are the functions associated with the list?**
   Lists are used to store multiple items in a single variable.

List Length

To determine how many items a list has, use the len() function:

Example

Print the number of items in the list:

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

```
3
```

Append Items

To add an item to the end of the list, use the append() method:

Example

Using the append() method to append an item:

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

```
['apple', 'banana', 'cherry', 'orange']
```

Insert Items

To insert a list item at a specified index, use the insert() method.

The insert() method inserts an item at the specified index:

Example

Insert an item as the second position:

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)
```

```
['apple', 'orange', 'banana', 'cherry']
```

Remove Specified Item

The remove() method removes the specified item.

Example

Remove "apple":

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("apple")
print(thislist)
```

```
['banana', 'cherry']
```

Remove Specified Index

The pop() method removes the specified index.

Example

Remove the second item:

```
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)
```

4. What is tuple? What are the functions associated with the tuple?

Tuples are used to store multiple items in a single variable.

Tuple Length

To determine how many items a tuple has, use the len() function:

Example

Print the number of items in the tuple:

```
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
```

```
3
```

Add Items

Since tuples are immutable, they do not have a built-in append() method, but there are other ways to add items to a tuple.

1. **Convert into a list**: Just like the workaround for changing a tuple, you can convert it into a list, add your item(s), and convert it back into a tuple.

Example

Convert the tuple into a list, add "orange", and convert it back into a tuple:

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.append("orange")
thistuple = tuple(y)
```

```
('apple', 'banana', 'cherry', 'orange')
```

Remove Items

Note: You cannot remove items in a tuple.

Tuples are **unchangeable**, so you cannot remove items from it, but you can use the same workaround as we used for changing and adding tuple items:

Example

Convert the tuple into a list, remove "apple", and convert it back into a tuple:

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.remove("apple")
thistuple = tuple(y)
```

```
('banana', 'cherry')
```

Or you can delete the tuple completely:

Example

The del keyword can delete the tuple completely:

```
thistuple = ("apple", "banana", "cherry")
del thistuple
print(thistuple) #this will raise an error because the tuple no longer exists
```

```
Traceback (most recent call last):
  File "demo_tuple_del.py", line 3, in <module>
    print(thistuple) #this will raise an error because the tuple no longer exists
NameError: name 'thistuple' is not defined
```

5. Explain slicing and string format method.

Slicing

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

Example

Get the characters from position 2 to position 5 (not included):

```
b = "Hello, World!"
print(b[2:5])
```

```
llo
```

Note: The first character has index 0.

Slice From the Start

By leaving out the start index, the range will start at the first character:

Example

Get the characters from the start to position 5 (not included):

```
b = "Hello, World!"
print(b[:5])
```

```
Hello
```

Slice To the End

By leaving out the end index, the range will go to the end:

Example

Get the characters from position 2, and all the way to the end:

```
b = "Hello, World!"
print(b[2:])
```

```
llo, World!
```

String Format

We can combine strings and numbers by using the format() method!

The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are:

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
```

```
I want 3 pieces of item 567 for 49.95 dollars.
```

6. Explain data type conversion in python.

You can convert from one type to another with the int(), float(), and complex() methods:

```
x = 1    # int
y = 2.8  # float
z = 1j   # complex

#convert from int to float:
a = float(x)

#convert from float to int:
b = int(y)

#convert from int to complex:
c = complex(x)

print(a)
print(b)
print(c)

print(type(a))
print(type(b))
print(type(c))
```

```
1.0
2
(1+0j)
<class 'float'>
<class 'int'>
<class 'complex'>
```

7. **Explain**
    a. **Identity operators**
    b. **Membership operators**

Python Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Python Membership Operators

Membership operators are used to test if a sequence is presented in an object:

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

8. **Explain**
    a. **Break statement**
    b. **Continue statement**

The break Statement

With the break statement we can stop the loop before it has looped through all the items:

Example

Exit the loop when x is "cherry":

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

```
apple
banana
cherry
```

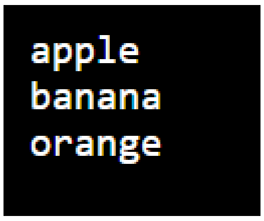
The continue Statement

With the continue statement we can stop the current iteration of the loop, and continue with the next:

Example

Do not print cherry:

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



```
apple
banana
orange
```

9. **Explain the working of if condition along with elif and else with example?**

   An "if statement" is written by using the if keyword.

Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

If statement, without indentation will raise an error

The elif keyword is Python's way of saying "if the previous conditions were not true, then try this condition".

The else keyword catches anything which isn't caught by the preceding conditions.

Example

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
a is greater than b
```

In this example a is greater than b, so the first condition is not true, also the elif condition is not true, so we go to the else condition and print to screen that "a is greater than b".

10.Explain for loop? How to use for loop to loop through a string?

For Loops

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

Example

Print each fruit in a fruit list:

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

```
apple
banana
cherry
```

The for loop does not require an indexing variable to set beforehand.

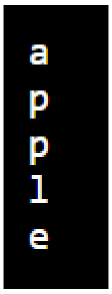
Looping Through a String

Even strings are iterable objects, they contain a sequence of characters:

Example

Loop through the letters in the word "apple":

```
for x in "apple":
  print(x)
```



```
a
p
p
l
e
```

11.Explain the concept of random number.

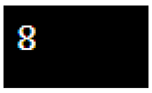
Random Number

Python does not have a random() function to make a random number, but Python has a built-in module called random that can be used to make random numbers:

Example

Import the random module, and display a random number between 1 and 9:

```
import random

print(random.randrange(1, 10))
```



```
8
```

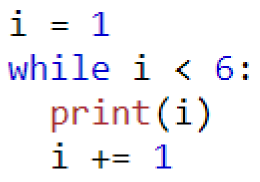
12.Explain while loop with an example.

The while Loop

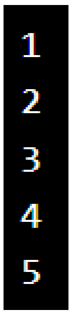
With the while loop we can execute a set of statements as long as a condition is true.

Example

Print i as long as i is less than 6:



```python
i = 1
while i < 6:
  print(i)
  i += 1
```



```
1
2
3
4
5
```

Note: remember to increment i, or else the loop will continue forever.

The while loop requires relevant variables to be ready, in this example we need to define an indexing variable, i, which we set to 1.

Multiple choice questions.

1. What is the data type of the variable 'x' in Python, if x = 5?
   A) **Integer**
   B) String
   C) Float
   D) Boolean
2. What data type would you use to store a sequence of characters in Python?
   A) Integer
   B) Float
   C) **String**
   D) Boolean
3. What will be the data type of the result of 10 * 2.5 in Python?
   A) Integer
   B) **Float**
   C) String
   D) Boolean
4. What is the data type of the result of the expression "Hello" + "World" in Python?
   A) Integer
   B) Float
   C) **String**
   D) Boolean
5. What will be the data type of the result of 10 > 5 in Python?
   A) Integer
   B) Float
   C) String
   D) **Boolean**

6. What function in Python is used to determine the length of a string?
   A) **len**()
   B) length()
   C) str_len()
   D) strlen()
7. Which of the following methods in Python is used to convert a string to uppercase?
   A) to_upper()
   B) uppercase()
   C) **upper**()
   D) toUpperCase()
8. Which method in Python is used to remove leading and trailing whitespace characters from a string?
   A) trim()
   B) **strip**()
   C) clean()
   D) remove()
9. What function in Python is used to replace occurrences of a substring within a string with another substring?
   A) **replace**()
   B) swap()
   C) substitute()
   D) modify()
10. What function in Python is used to format strings with placeholders?
    A) **format**()
    B) interpolate()
    C) insert()
    D) merge()
11. What is the result of 5 + 3 in Python?
    A) **8**
    B) 15
    C) 53
    D) Error
12. Which operator in Python is used for string concatenation?
    A) &
    B) **+**
    C) *
    D) %
13. What is the result of the expression 10 > 5 in Python?
    A) **True**
    B) False
    C) 1
    D) Error
14. Which looping statement in Python executes a block of code repeatedly as long as a condition is true?
    A) for loop
    B) **while loop**
    C) do-while loop
    D) until loop

15. In Python, how can you exit from a loop prematurely?
    A) **Using the break statement**
    B) Using the stop statement
    C) Using the exit statement
    D) Using the end statement
16. What does the 'continue' statement do in Python?
    A) Exits the loop
    B) **Skips the current iteration and proceeds to the next iteration**
    C) Stops the program
    D) Restarts the loop from the beginning
17. Which loop in Python is typically used when you know the exact number of iterations?
    A) while loop
    B) **for loop**
    C) do-while loop
    D) until loop
18. What keyword is used for making decisions in Python?
    A) decide
    B) choose
    C) **if**
    D) switch
19. Which of the following is the correct syntax for an if statement in Python?
    A) if condition { }
    B) **if condition: (with indentation for block)** C) if condition then { }
    D) if condition; { }
20. What does the 'elif' keyword signify in Python?
    A) **It is short for "else if" and is used to check multiple conditions**.
    B) It indicates the start of a loop.
    C) It means "else" in Python.
    D) It indicates a comment.
21. In Python, what is the purpose of the 'else' statement?
    A) It is used for error handling.
    B) It is used for input/output operations.
    C) It is used to execute a block of code if the condition is true.
    D) **It is used to execute a block of code if the condition is false.**
22. How do you define a tuple in Python?
    A) Using square brackets [ ]
    B) **Using parentheses ( )**
    C) Using curly braces { }
    D) Using angle brackets < >
23. How do you define a list in Python?
    A) **Using square brackets [ ]**
    B) Using parentheses ( )
    C) Using curly braces { }
    D) Using angle brackets < >
24. How do you define an empty dictionary in Python?
    A) empty_dict = []
    B) **empty_dict = {}**
    C) empty_dict = ()
    D) empty_dict = None

25. What is the main purpose of using dictionaries in Python?
    A) To store elements in a sorted order
    B) To store elements with duplicate values
    C) **To store elements as key-value pairs**
    D) To store elements in a stack

Module 3

1. Explain module in python.

Module is same as a code library.

A file containing a set of functions we want to include in our application.

Create a Module

To create a module just save the code in a file with the file extension .py:

Example

Save this code in a file named mymodule.py

```
def greeting(name):
  print("Hello, " + name)
```

Use a Module

We can use the module we just created, by using the import statement:

Example

Import the module named mymodule, and call the greeting function:

```
import mymodule

mymodule.greeting("Jonathan")
```

```
Hello, Jonathan
```

Note: When using a function from a module, use the syntax: *module_name.function_name*.

2. How to create pie chart using matplotlib

With Pyplot, you can use the pie() function to draw pie charts:

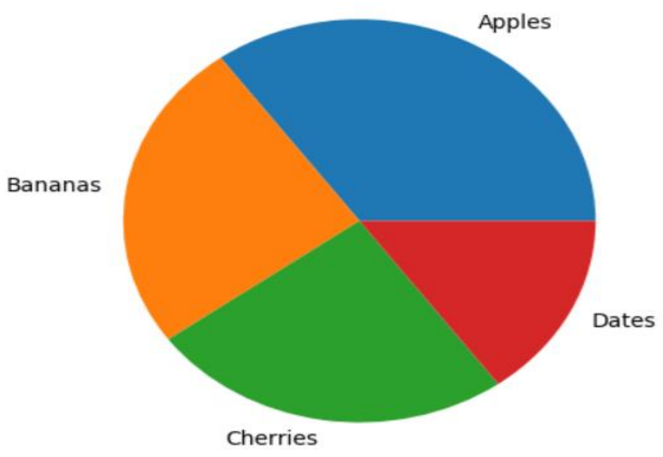
Add labels to the pie chart with the labels parameter.

The labels parameter must be an array with one label for each wedge:

Example

A simple pie chart:

```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

plt.pie(y, labels = mylabels)
plt.show()
```



3. How to create bars using matplotlib

With Pyplot, you can use the bar() function to draw bar graphs

The bar() function takes arguments that describes the layout of the bars.

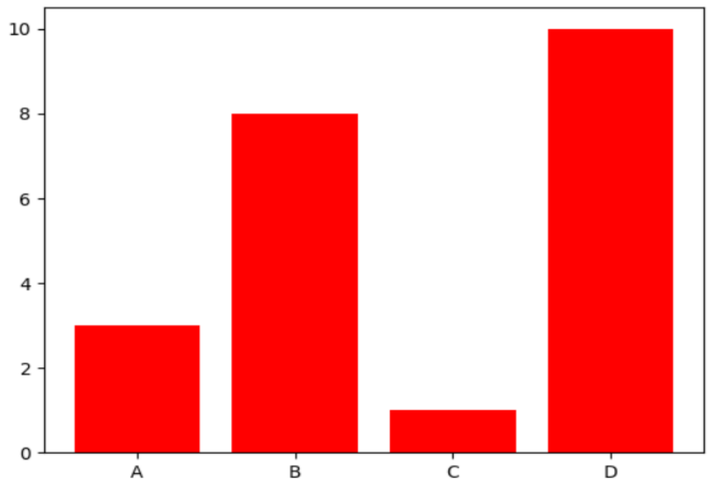
The categories and their values represented by the *first* and *second* argument as arrays.

The keyword argument color to set the color of the bars

Example

Draw 4 red bars:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x, y, color = "red")
plt.show()
```



4. How to create Subplot using matplotlib

The subplot() function takes three arguments that describes the layout of the figure.

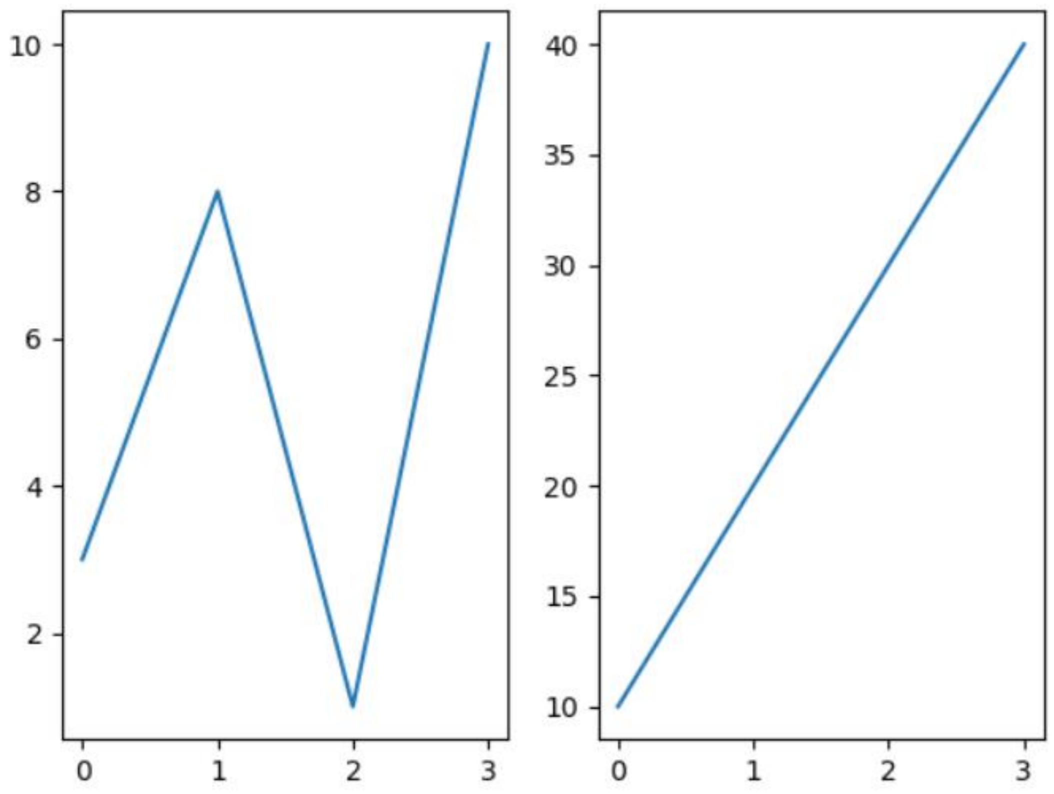
The layout is organized in rows and columns, which are represented by the *first* and *second* argument.

The third argument represents the index of the current plot.

```
plt.subplot(1, 2, 1)
#the figure has 1 row, 2 columns, and this plot is the first plot.
```

```
plt.subplot(1, 2, 2)
#the figure has 1 row, 2 columns, and this plot is the second plot.
```

```python
import matplotlib.pyplot as plt
import numpy as np

#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(1, 2, 1)
plt.plot(x,y)

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(1, 2, 2)
plt.plot(x,y)

plt.show()
```



5. Explain opening a file with syntax and example.

The key function for working with files in Python is the open() function.

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

There are four different methods (modes) for opening a file:

"r" - Read - Default value. Opens a file for reading

"a" - Append - Opens a file for appending

"w" - Write - Opens a file for writing

"x" - Create - Creates the specified file

In addition you can specify if the file should be handled as binary or text mode

"t" - Text - Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

Syntax

To open a file for reading it is enough to specify the name of the file:

```
f = open("demofile.txt")
```

6. Explain closing the file with syntax and example.

It is a good practice to always close the file when you are done with it.

Example

Close the file when you are finish with it:

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

```
Hello! Welcome to demofile.txt
```

7. How to write to an file?

To write to a 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

demofile2.txt

Hello! Welcome to demofile2.txt

This file is for testing purposes.

Good luck!

Example 1

Open the file "demofile2.txt" and append content to the file:

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()

#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
```

```
Hello! Welcome to demofile2.txt
This file is for testing purposes.
Good Luck!Now the file has more content!
```

Example 2

Open the file "demofile2.txt" and overwrite the content:

```
f = open("demofile2.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()

#open and read the file after the overwriting:
f = open("demofile2.txt", "r")
print(f.read())
```

```
Woops! I have deleted the content!
```

Note: the "w" method will overwrite the entire file.

8. Explain

a) Create a new file

b) Delete a file

Create

To create a new file in Python, use the open() method, with one of the following parameters:

"x" - Create - will create a file

Example

Create a file called "myfile.txt":

```
f = open("myfile.txt", "x")
```

Result: a new empty file is created!

Delete

To delete a file, you must import the OS module, and run its os.remove() function:

Example

Remove the file "demofile.txt":

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

9. What is data frame? How to create a data frame?

A Pandas Data Frame is a 2-dimensional data structure, like a 2-dimensional array, or a table with rows and columns.

Example

Create a simple Pandas Data Frame:

```
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

#load data into a DataFrame object:
df = pd.DataFrame(data)

print(df)
```

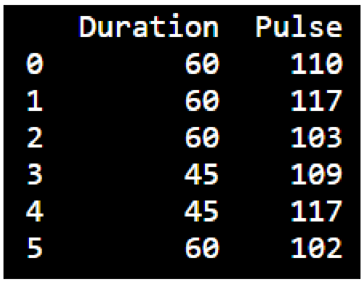
```
   calories  duration
0       420        50
1       380        40
2       390        45
```

10. What are the two methods for viewing the data of data frame?

1. **head() method:** The head() method returns the headers and a specified number of rows, starting from the top.

   Note: if the number of rows is not specified, the head() method will return the top 5 rows.

data.csv

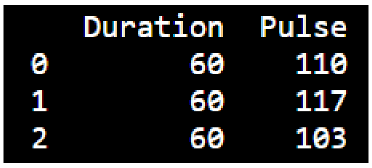


```
   Duration  Pulse
0        60    110
1        60    117
2        60    103
3        45    109
4        45    117
5        60    102
```

Example

Print the first 3 rows of the DataFrame:

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df.head(3))
```



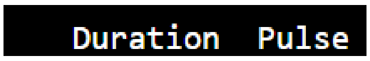
```
   Duration  Pulse
0        60    110
1        60    117
2        60    103
```

2. **tail() method:** The tail() method returns the headers and a specified number of rows, starting from the bottom.

   Example

Print the last 3 rows of the DataFrame:

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df.tail(3))
```



```
   Duration  Pulse
```

```
3          45     109
4          45     117
5          60     102
```

Note: if the number of rows is not specified, the `tail()` method will return the bottom 5 rows.

11.How to create a 1-D array and access the elements of it?

Create a 1-D array containing the values 1,2,3,4,5:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print(arr)
```

```
[1 2 3 4 5]
```

Access Array Elements

We can access an array element by referring to its index number.

The indexes in NumPy arrays start with 0, meaning that the first element has index 0, and the second has index 1 etc.

Get the first element from the above array:

```
print(arr[0])
```

```
1
```

12.Explain slicing of arrays.

Slicing arrays

Slicing in python means taking elements from one given index to another given index.

We pass slice instead of index like this: [*start*:*end*].

Slice elements from index 1 to index 5 from the following array:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5])
```

```
[2 3 4 5]
```

Multiple choice questions.

1. How do you import a module named "example_module" in Python?
   A) **import example_module**
   B) include example_module
   C) load example_module
   D) use example_module
2. Which statement accurately describes the import statement in Python?
   A) **It loads the module only once, even if imported multiple times in a program.**
   B) It loads the module every time it is imported in a program.
   C) It is used to export functions and variables from a module.
   D) It imports all functions and variables from a module automatically.
3. How do you access a function named "my_function" from an imported module named "my_module"?
   A) my_function.my_module
   B) **my_module.my_function**
   C) my_function->my_module
   D) my_module::my_function

4. What happens when you import a module in Python?
   A) **It executes the module's code.**
   B) It makes the module available for use in the current program.
   C) It prints the module's documentation.
   D) It initializes the module's variables.
5. How can you alias a module named "long_module_name" to use it with a shorter name?
   A) alias long_module_name as lm
   B) from long_module_name import as lm
   C) **import long_module_name as lm**
   D) use long_module_name as lm
6. How can you create a module in Python?
   A) **Write Python code in a text file with a .py extension.**
   B) Compile Python code into a shared library.
   C) Use a built-in Python function to generate a module.
   D) Modules can only be created by Python core developer
7. What is NumPy?
   a) **A scientific computing library in Python**
   b) A machine learning framework
   c) A data visualization tool
   d) A web development framework

8. Which of the following is the correct way to import NumPy in Python?
   a) **import numpy as np**
   b) from numpy import *
   c) import numpy
   d) from numpy import numpy
9. What is the primary data structure in NumPy?
   a) **Array**
   b) List
   c) Dictionary
   d) Set
10. What function is used to generate random numbers in NumPy?
   a) random.random()
   b) np.rand()
   c) **np.random()**
   d) random.randint()
11. What is a DataFrame in Python Pandas?
   a) A graphical representation of data
   b) **A two-dimensional labeled data structure**
   c) A function for plotting data
   d) A machine learning model
12. Which library in Python is commonly used for working with DataFrames?
   a) NumPy
   b) **Pandas**
   c) Matplotlib
   d) Scikit-learn

13. How do you create a DataFrame from a dictionary **data** in Pandas?
   a) **pd.DataFrame(data)**
   b) DataFrame(data)
   c) pd.createDataFrame(data)
   d) DataFrame.from_dict(data)
14. What function is used to display the first few rows of a DataFrame in Pandas?
   a) display()
   b) show()
   c) **head**()
   d) top()
15. How can you access a specific column named 'column_name' in a DataFrame **df**?
   a) df.column_name
   b) df.get_column('column_name')
   c) **df['column_name']**
   d) df.column['column_name']
16. What is Matplotlib?
   a) A machine learning library
   b) A data manipulation library
   c) **A data visualization library**
   d) A web development framework
17. How do you import Matplotlib in Python?
   a) import matplot
   b) **import matplotlib.pyplot as plt**
   c) import matplotlib as mpl

d) import plt.matplotlib

18. Which of the following is the correct way to create a basic line plot using Matplotlib?
    a) **plt.plot()**
    b) plt.line()
    c) plt.create_plot()
    d) plt.line_plot()
19. What function is used to add a title to a Matplotlib plot?
    a) plt.add_title()
    b) **plt.title()**
    c) plt.plot_title()
    d) plt.set_title()
20. How can you add labels to the x-axis and y-axis in a Matplotlib plot?
    a) plt.add_labels()
    b) **plt.x_label() and plt.y_label()**
    c) plt.xlabel() and plt.ylabel()
    d) plt.set_xlabel() and plt.set_ylabel()
21. What function is used to create a scatter plot in Matplotlib?
    a) **plt.scatter()**
    b) plt.plot()
    c) plt.create_scatter()
    d) plt.scatter_plot()
22. How do you create subplots in Matplotlib?
    a) plt.create_subplots()
    b) plt.add_subplot()
    c) plt.subplot()
    d) **plt.subplots()**
23. How can you add a legend to a Matplotlib plot?
    a) **plt.legend()**
    b) plt.add_legend()
    c) plt.create_legend()
    d) plt.set_legend()
24. How do you create a histogram in Matplotlib?
    a) **plt.hist()**
    b) plt.create_histogram()
    c) plt.plot_hist()
    d) plt.histogram()
25. Which method is used to add grid lines to a Matplotlib plot?
    a) **plt.grid()**
    b) plt.add_grid()
    c) plt.create_grid()
    d) plt.set_grid()

Module 4

1. **What is machine learning? What is data set?**

Machine Learning

Machine Learning is making the computer learn from studying data and statistics.

Machine Learning is a step into the direction of artificial intelligence (AI).

Machine Learning is a program that analyses data and learns to predict the outcome.

Data Set

In the mind of a computer, a data set is any collection of data. It can be anything from an array to a complete database.

Example of an array:

```
[99,86,87,88,111,86,103,87,94,78,77,85,86]
```

Example of a database:

Carname	Color	Age	Speed	AutoPass
BMW	red	5	99	Y
Volvo	black	7	86	Y
VW	gray	8	87	N
VW	white	7	88	Y
Ford	white	2	111	Y
VW	white	17	86	Y
Tesla	red	2	103	Y
BMW	black	9	87	Y
Volvo	gray	4	94	N
Ford	white	11	78	N
Toyota	gray	12	77	N
VW	white	9	85	N
Toyota	blue	6	86	Y

2. **What are the data types in machine learning?**

To analyze data, it is important to know what type of data we are dealing with.

We can split the data types into three main categories:

- **Numerical**
- **Categorical**
- **Ordinal**

Numerical data are numbers, and can be split into two numerical categories:

- Discrete Data
  - counted data that are limited to integers. Example: The number of cars passing by.
- Continuous Data
  - measured data that can be any number. Example: The price of an item, or the size of an item

Categorical data are values that cannot be measured up against each other. Example: a color value, or any yes/no values.

Ordinal data are like categorical data, but can be measured up against each other. Example: school grades where A is better than B and so on.

3. **Explain how to use mean with an example?**

Mean

The mean value is the average value.

To calculate the mean, find the sum of all values, and divide the sum by the number o0f values:

```
(99+86+87+88+111+86+103+87+94+78+77+85+86) / 13 = 89.77
```

Example

Use the NumPy mean() method to find the average speed:

```
import numpy

speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]

x = numpy.mean(speed)

print(x)
```

4. **Explain how to use median with an example?**

1st example:

The median value is the value in the middle, after you have sorted all the values:

```
77, 78, 85, 86, 86, 86, 87, 87, 88, 94, 99, 103, 111
```

It is important that the numbers are sorted before you can find the median.

The NumPy module has a method for this:

Example

Use the NumPy median() method to find the middle value:

```
import numpy

speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]

x = numpy.median(speed)

print(x)
```

```
87.0
```

2nd example:

If there are two numbers in the middle, divide the sum of those numbers by two.

```
77, 78, 85, 86, 86, 86, 87, 87, 94, 98, 99, 103

(86 + 87) / 2 = 86.5
```

Example

Using the NumPy module:

```
import numpy

speed = [99,86,87,88,86,103,87,94,78,77,85,86]

x = numpy.median(speed)

print(x)
```

```
86.5
```

5. **Explain how to use mode with an example?**

The Mode value is the value that appears the most number of times:

```
99, 86, 87, 88, 111, 86, 103, 87, 94, 78, 77, 85, 86 = 86
```

The SciPy module has a method for this. Learn about the SciPy module in our SciPy Tutorial.

Example

Use the SciPy mode() method to find the number that appears the most:

```
from scipy import stats

speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]

x = stats.mode(speed)

print(x)
```

```
ModeResult(mode=array([86]), count=array([3]))
```

6. **Explain how to use standard deviation with example.**

Standard deviation is a number that describes how spread out the values are.

The NumPy module has a method to calculate the standard deviation:

Example

Use the NumPy std() method to find the standard deviation:

```
import numpy

speed = [86,87,88,89]

x = numpy.std(speed)

print(x)
```

```
1.118033988749895
```

1. Calculate the mean:

   $$\text{Mean} = \frac{88+87+89+86}{4} = \frac{350}{4} = 87.5$$

2. Find the squared difference between each number and the mean:

$$(88-87.5)^2 = (0.5)^2 = 0.25$$
$$(87-87.5)^2 = (-0.5)^2 = 0.25$$
$$(89-87.5)^2 = (1.5)^2 = 2.25$$
$$(86-87.5)^2 = (-1.5)^2 = 2.25$$

1. Calculate the mean of those squared differences:

   $$\text{Mean of squared differences} = \frac{0.25+0.25+2.25+2.25}{4} = \frac{5}{4} = 1.25$$

2. Take the square root of the result:

   $$\text{Standard deviation} = \sqrt{1.25} \approx 1.118$$

7. Explain how to use variance with example.

Variance is another number that indicates how spread out the values are.

1. Find the mean:

   Mean = (86 + 87 + 88 + 89) / 4

   = 350 / 4

   = 87.5
2. Find the deviation from the mean for each number:

   Deviation from mean for 86 = 86 - 87.5 = -1.5

   Deviation from mean for 87 = 87 - 87.5 = -0.5

   Deviation from mean for 88 = 88 - 87.5 = 0.5

   Deviation from mean for 89 = 89 - 87.5 = 1.5
3. Square each deviation:

   (-1.5)^2 = 2.25

   (-0.5)^2 = 0.25

   (0.5)^2 = 0.25

   (1.5)^2 = 2.25
4. Find the mean of the squared deviations:

   Variance = (2.25 + 0.25 + 0.25 + 2.25) / 4

   = 5 / 4

   = 1.25

```
import numpy

speed = [86,87,88,89]

x = numpy.var(speed)

print(x)
```

```
1.25
```

8. What do you mean by class and object? Explain with example.

A class is considered a blueprint of objects. A class contains the blueprints or the prototype from which the objects are being created.

We use the `class` keyword to create a class in Python. For example,

```
class Bike:
    name = ""
    gear = 0
```

Here,

1. Bike - the name of the class
2. name/gear - variables inside the class with default values "" and **0** respectively.

An object is called an instance of a class.

```
# create class
class Bike:
    name = ""
    gear = 0

# create objects of class
bike1 = Bike()
```

Here, bike1 is the object of the class.

9. Explain scatterplot with an example

Scatter Plot

A scatter plot is a diagram where each value in the data set is represented by a dot.

The Matplotlib module has a method for drawing scatter plots, it needs two arrays of the same length, one for the values of the x-axis, and one for the values of the y-axis:

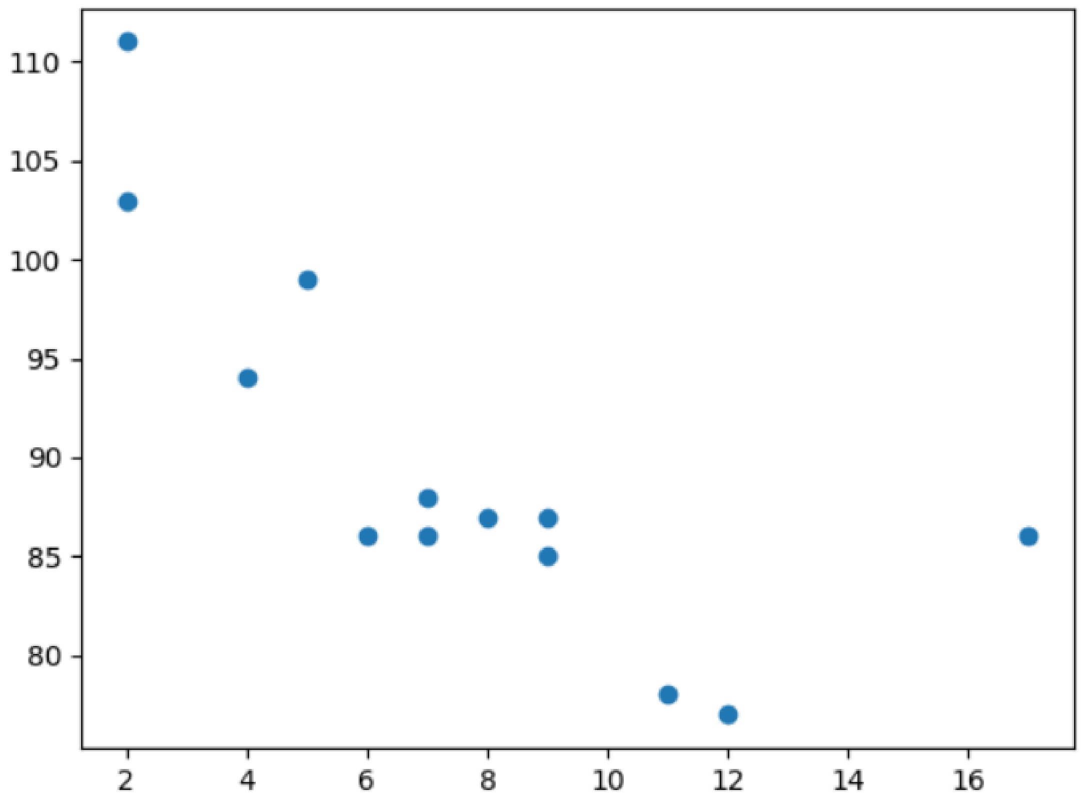
The x array represents the age of each car.

The y array represents the speed of each car.

Example

Use the scatter() method to draw a scatter plot diagram:

```
import matplotlib.pyplot as plt

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

plt.scatter(x, y)
plt.show()
```



Scatter Plot Explained

The x-axis represents ages, and the y-axis represents speeds.

What we can read from the diagram is that the two fastest cars were both 2 years old, and the slowest car was 12 years old.

10. Explain linear regression with an example.
11. Explain polynomial regression with an example.
12. Explain multiple regression with an example.

Multiple choice questions:

1. In Python, which library provides a function to calculate the mean of a list of numbers?
   a) NumPy
   b) Pandas
   c) Math
   d) SciPy
   Answer: a) NumPy
2. Which of the following functions in Python's NumPy library is used to calculate the mean?
   a) np.mean()
   b) np.average()
   c) np.median()
   d) np.mode()
   Answer: a) np.mean()
3. Which of the following expressions correctly calculates the mean of a list named **data** in Python using NumPy?
   A) **mean(data)**
   B) **np.mean(data)**
   C) **.mean(data)**
   D) All of the above
   Answer: C) np.mean(data)
4. Which of the following is an advantage of using NumPy's **mean()** function over Python's built-in **sum()** and **len()** functions to calculate the mean?
   A) NumPy is faster for large datasets.
   B) NumPy can handle only integer data types.
   C) NumPy does not support multidimensional arrays.
   D) NumPy requires additional libraries for installation.
   Answer: A) NumPy is faster for large datasets.
5. What does the **mean()** function in Python calculate?
   A) Median
   B) Mode
   C) Mean
   D) Range
   Answer: C) Mean
6. Which of the following statements best describes the median in statistics?
   A) The most frequently occurring value in a dataset.
   B) The arithmetic average of a dataset.
   C) The middle value in a sorted dataset.
   D) The difference between the maximum and minimum values in a dataset.
   Answer: C) The middle value in a sorted dataset.
7. In Python, which library is commonly used for calculating the median?
   A) NumPy
   B) Matplotlib
   C) Pandas
   D) SciPy
   Answer: A) NumPy

8. What is the result if the number of elements in a dataset is even and there is no single middle value?
   A) The average of the two middle values.
   B) The sum of the two middle values.
   C) The first middle value encountered.
   D) An error is raised.
   Answer: A) The average of the two middle values.
9. Which of the following is true regarding the calculation of median in Python?
   A) It always requires the dataset to be sorted.
   B) It works only with integer data types.
   C) It cannot handle missing values.
   D) It returns the middle value of the dataset.
   Answer: A) It always requires the dataset to be sorted.
10. In Python, which function is used to find the median of a list of numbers using NumPy?
   A) median()
   B) mean()
   C) mode()
   D) mid()
   Answer: A) median()
11. What does the mode represent in a dataset?
   A) The average value
   B) The middle value when the data is sorted
   C) The most frequent value
   D) The total number of values
   Answer: C) The most frequent value
12. In Python, which function is used to find the mode of a list of numbers?
   A) mode()
   B) median()
   C) mean()
   D) max()
   Answer: A) mode()
13. What is the mode of the following dataset: [12, 15, 17, 20, 17, 25, 17]?
   A) 17
   B) 20
   C) 15
   D) 25
   Answer: A) 17
14. What does standard deviation measure in a dataset?
   A) The average value
   B) The spread or dispersion of the data
   C) The most frequent value
   D) The middle value when the data is sorted
   Answer: B) The spread or dispersion of the data
15. Which Python library is commonly used to calculate the standard deviation of a dataset?
   A) NumPy
   B) Pandas
   C) Matplotlib

D) SciPy
Answer: A) NumPy

16. In Python, which function is used to find the standard deviation of a list of numbers using NumPy?
A) std_dev()
B) stdev()
C) standard_deviation()
D) std()
Answer: D) std()
17. What does variance measure in a dataset?
A) The spread or dispersion of the data
B) The average value
C) The most frequent value
D) The middle value when the data is sorted
Answer: A) The spread or dispersion of the data
18. Which Python library is commonly used to calculate the variance of a dataset?
A) NumPy
B) Pandas
C) Matplotlib
D) SciPy
Answer: A) NumPy
19. In Python, which function is used to find the variance of a list of numbers using NumPy?
A) variance()
B) var()
C) compute_variance()
D) std()
Answer: B) var()
20. The variance is the square of:
A) The mean
B) The median
C) The standard deviation
D) The mode
Answer: C) The standard deviation
21. What is a class in Python?
A) A function that returns objects
B) A collection of variables and functions
C) A built-in data type
D) A loop structure
Answer: B) A collection of variables and functions
22. Which keyword is used to define a class in Python?
A) class
B) def
C) new
D) obj
Answer: A) class
23. What is an object in Python?
A) An instance of a class
B) A built-in data type

C) A reserved keyword
D) A loop structure
Answer: A) An instance of a class

24. How do you create an object of a class in Python?
A) Using the **new** keyword
B) Using the **def** keyword
C) Using the **object()** function
D) Using the class name followed by parentheses
Answer: D) Using the class name followed by parentheses

Module 5

Question and Answer:

1. How to create database?

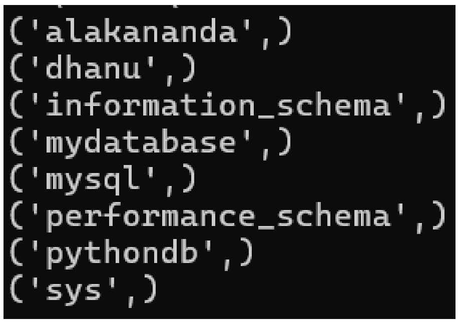
To create a database in MySQL, use the "CREATE DATABASE" statement:

create a database named "mydatabase":

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="password"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE DATABASE mydatabase")
```

Check if Database Exists

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="password"
)
mycursor = mydb.cursor()

mycursor.execute("SHOW DATABASES")
for x in mycursor:
    print(x)
```



```
('alakananda',)
('dhanu',)
('information_schema',)
('mydatabase',)
('mysql',)
('performance_schema',)
('pythondb',)
('sys',)
```

2. How to create table?

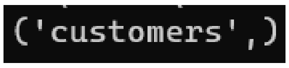
To create a table in MySQL, use the "CREATE TABLE" statement.

Create a table named "customers":

```python
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="password",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))")
```

Check if Table Exists

```python
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="password",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SHOW TABLES")

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



```
('customers',)
```

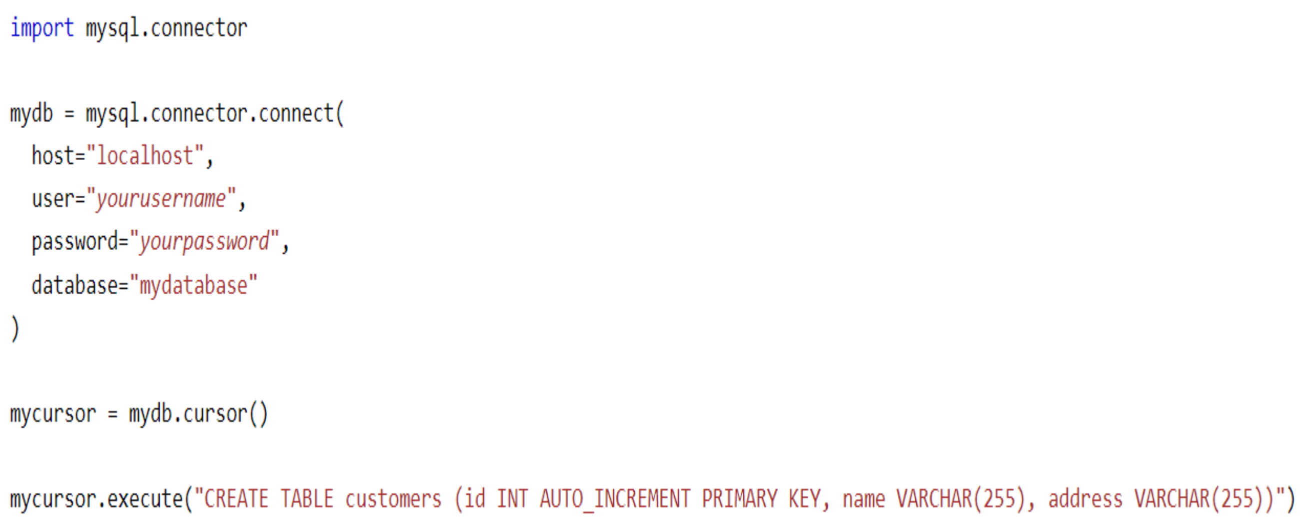
3. How to create primary key for a table? Explain with example.

When creating a table, you should also create a column with a unique key for each record.

This can be done by defining a PRIMARY KEY.

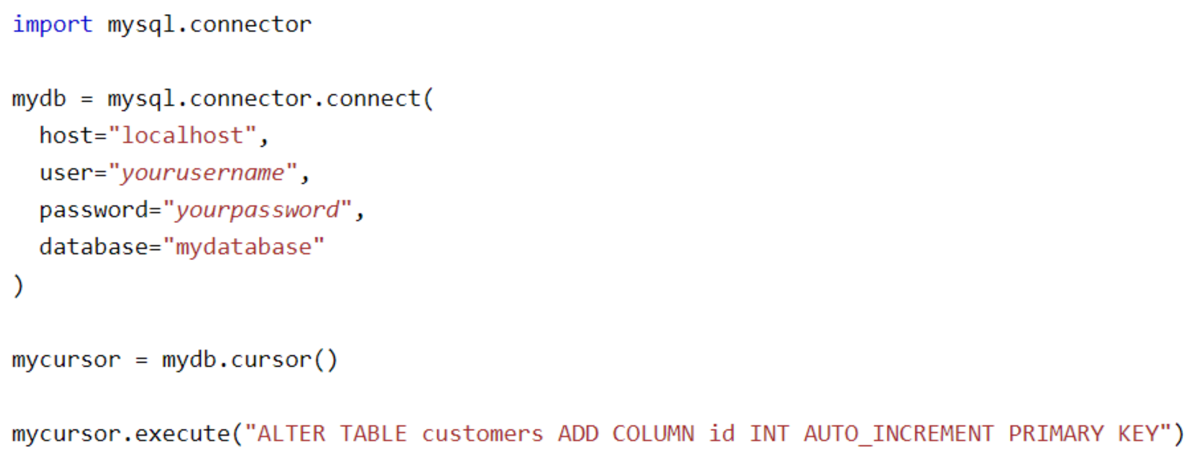
We use the statement "INT AUTO_INCREMENT PRIMARY KEY" which will insert a unique number for each record. Starting at 1, and increased by one for each record.

Create primary key when creating the table:



```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(255), address VARCHAR(255))")
```

Create primary key on an existing table:



```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("ALTER TABLE customers ADD COLUMN id INT AUTO_INCREMENT PRIMARY KEY")
```

4. Explain insert statement with example.

To fill a table in MySQL, use the "INSERT INTO" statement.

Insert a record in the "customers" table:

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = ("John", "Highway 21")
mycursor.execute(sql, val)

mydb.commit()

print(mycursor.rowcount, "record inserted.")
```

```
1 record inserted.
```

Insert Multiple Rows

To insert multiple rows into a table, use the executemany() method.

The second parameter of the executemany() method is a list of tuples, containing the data you want to insert:

Fill the "customers" table with data:]

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = [
  ('Peter', 'Lowstreet 4'),
  ('Amy', 'Apple st 652'),
  ('Hannah', 'Mountain 21'),
  ('Michael', 'Valley 345'),
  ('Sandy', 'Ocean blvd 2'),
  ('Betty', 'Green Grass 1'),
  ('Richard', 'Sky st 331'),
  ('Susan', 'One way 98'),
  ('Vicky', 'Yellow Garden 2'),
  ('Ben', 'Park Lane 38'),
  ('William', 'Central st 954'),
  ('Chuck', 'Main Road 989'),
  ('Viola', 'Sideway 1633')
]

mycursor.executemany(sql, val)

mydb.commit()

print(mycursor.rowcount, "was inserted.")
```

```
13 record was inserted.
```

5. Explain SELECT statement with example.

To select from a table in MySQL, use the "SELECT" statement:

Example

Select all records from the "customers" table, and display the result:

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchall()

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

```
(1, 'John', 'Highway 21')
(2, 'Peter', 'Lowstreet 27')
(3, 'Amy', 'Apple st 652')
(4, 'Hannah', 'Mountain 21')
(5, 'Michael', 'Valley 345')
(6, 'Sandy', 'Ocean blvd 2')
(7, 'Betty', 'Green Grass 1')
(8, 'Richard', 'Sky st 331')
(9, 'Susan', 'One way 98')
(10, 'Vicky', 'Yellow Garden 2')
(11, 'Ben', 'Park Lane 38')
(12, 'William', 'Central st 954')
(13, 'Chuck', 'Main Road 989')
(14, 'Viola', 'Sideway 1633')
(15, 'Michelle', 'Blue Village')
```

Using the fetchone() Method

If you are only interested in one row, you can use the fetchone() method.

The fetchone() method will return the first row of the result:

Example

Fetch only one row:

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchone()

print(myresult)
```

```
(1, 'John', 'Highway 21')
```

6. **Explain how to**
    a. **Select particular columns**
    b. **Select With a Filter**

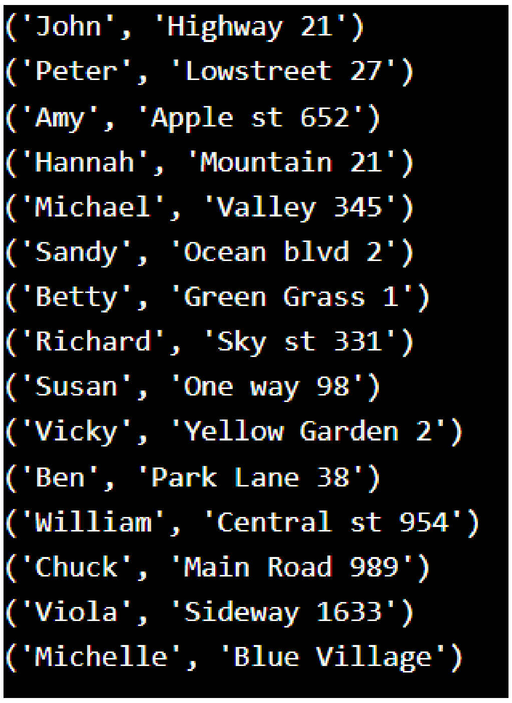
Selecting Columns

To select only some of the columns in a table, use the "SELECT" statement followed by the column name(s):

Example

Select only the name and address columns:

```python
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT name, address FROM customers")

myresult = mycursor.fetchall()

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



```
('John', 'Highway 21')
('Peter', 'Lowstreet 27')
('Amy', 'Apple st 652')
('Hannah', 'Mountain 21')
('Michael', 'Valley 345')
('Sandy', 'Ocean blvd 2')
('Betty', 'Green Grass 1')
('Richard', 'Sky st 331')
('Susan', 'One way 98')
('Vicky', 'Yellow Garden 2')
('Ben', 'Park Lane 38')
('William', 'Central st 954')
('Chuck', 'Main Road 989')
('Viola', 'Sideway 1633')
('Michelle', 'Blue Village')
```

Select using filter:

You can filter the selection by using the "WHERE" statement:

Example

Select record(s) where the address is "Park Lane 38": result:

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * FROM customers WHERE address ='Park Lane 38'"

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

```
(11, 'Ben', 'Park Lane 38')
```

7. Explain the use of ORDER BY with example.

Sort the Result

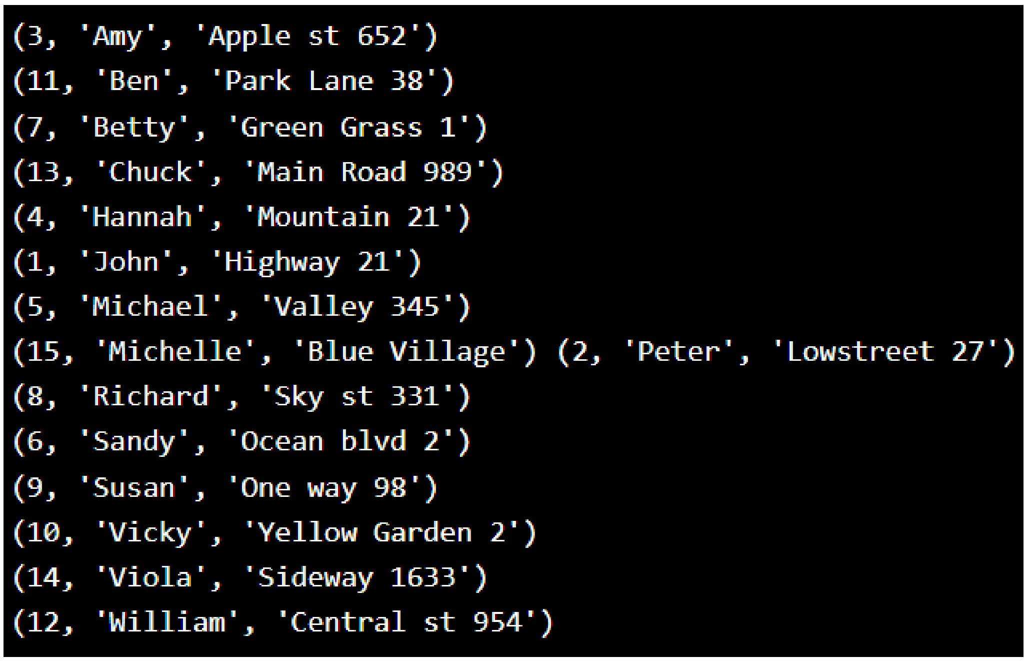
Use the ORDER BY statement to sort the result in ascending or descending order.

The ORDER BY keyword sorts the result ascending by default. To sort the result in descending order, use the DESC keyword.

Example

Sort the result alphabetically by name: result:

```python
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * FROM customers ORDER BY name"

mycursor.execute(sql)

myresult = mycursor.fetchall()

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



```
(3, 'Amy', 'Apple st 652')
(11, 'Ben', 'Park Lane 38')
(7, 'Betty', 'Green Grass 1')
(13, 'Chuck', 'Main Road 989')
(4, 'Hannah', 'Mountain 21')
(1, 'John', 'Highway 21')
(5, 'Michael', 'Valley 345')
(15, 'Michelle', 'Blue Village') (2, 'Peter', 'Lowstreet 27')
(8, 'Richard', 'Sky st 331')
(6, 'Sandy', 'Ocean blvd 2')
(9, 'Susan', 'One way 98')
(10, 'Vicky', 'Yellow Garden 2')
(14, 'Viola', 'Sideway 1633')
(12, 'William', 'Central st 954')
```

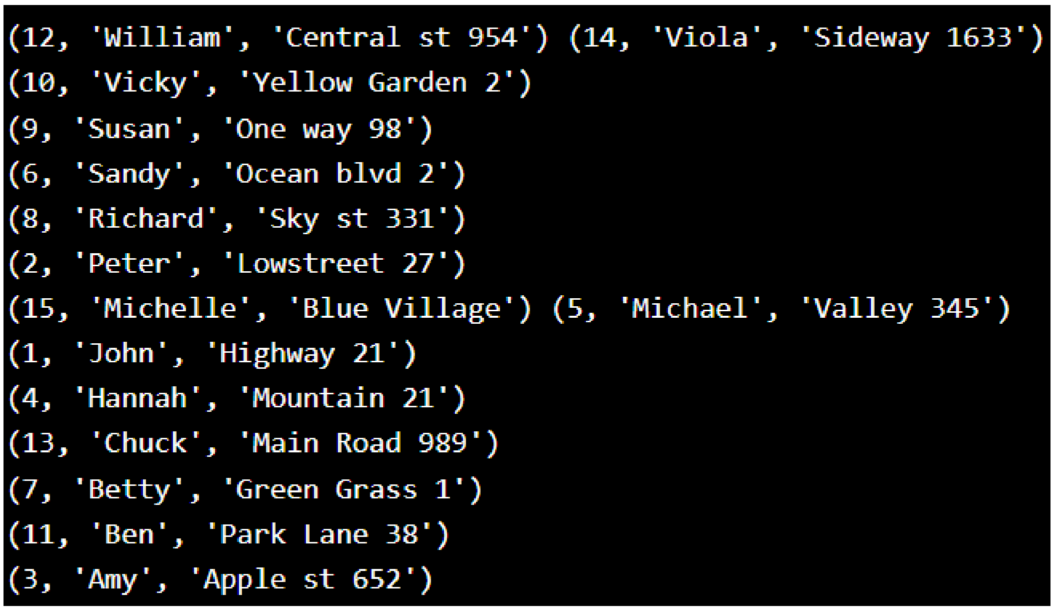
ORDER BY DESC

Use the DESC keyword to sort the result in a descending order.

Example

Sort the result reverse alphabetically by name:

```python
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "SELECT * FROM customers ORDER BY name DESC"

mycursor.execute(sql)

myresult = mycursor.fetchall()

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



```
(12, 'William', 'Central st 954') (14, 'Viola', 'Sideway 1633')
(10, 'Vicky', 'Yellow Garden 2')
(9, 'Susan', 'One way 98')
(6, 'Sandy', 'Ocean blvd 2')
(8, 'Richard', 'Sky st 331')
(2, 'Peter', 'Lowstreet 27')
(15, 'Michelle', 'Blue Village') (5, 'Michael', 'Valley 345')
(1, 'John', 'Highway 21')
(4, 'Hannah', 'Mountain 21')
(13, 'Chuck', 'Main Road 989')
(7, 'Betty', 'Green Grass 1')
(11, 'Ben', 'Park Lane 38')
(3, 'Amy', 'Apple st 652')
```

8. **Explain the following by using examples.**
   a. **UPDATE**
   b. **DELETE**

Update Table

You can update existing records in a table by using the "UPDATE" statement:

Example

Overwrite the address column from "Valley 345" to "Canyon 123":

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley 345'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) affected")
```

```
1 record(s) affected
```

Delete Record

You can delete records from an existing table by using the "DELETE FROM" statement:

Example

Delete any record where the address is "Mountain 21":

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE address = 'Mountain 21'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) deleted")
```

```
1 record(s) deleted
```

9. **Explain the following with examples.**
   a. **Commit**
   b. **Rollback**

The commit() method:

The commit() method is used to make sure the changes made to the database are consistent. It basically provides the database confirmation regarding the changes made by a user or an application in the database.

Syntax:

```
comm.commit() #comm refers to the database connection object
```

The rollback() method:

The rollback() method is used to revert the last changes made to the database. If a condition arises where one is not satisfied with the changes made to the database or a database transaction fails, the rollback() method can be used to retrieve the original data that was changed through the commit() method.

Syntax:

```
comm.rollback() #comm refers to the database connection object
```

Example:

```
sql = "DELETE from customers where address = 'Mountain 21'"

try:

    mycursor.execute(sql)
    mydb.commit()

except:

    mydb.rollback()

mydb.close()
```

10. Explain

a. **Python MongoDB Create database**

b. **Python MongoDB Create collection**

Creating a Database

To create a database in MongoDB, start by creating a MongoClient object, then specify a connection URL with the correct ip address and the name of the database you want to create.

Example

Create a database called "mydatabase":

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]
```

Creating a Collection

A **collection** in MongoDB is the same as a **table** in SQL databases.

To create a collection in MongoDB, use database object and specify the name of the collection you want to create.

Example

Create a collection called "customers":

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]

mycol = mydb["customers"]
```

11.Explain Python MongoDB Insert

A **document** in MongoDB is the same as a **record** in SQL databases.

To insert a record, or *document* as it is called in MongoDB, into a collection, we use the insert_one() method.

The first parameter of the insert_one() method is a dictionary containing the name(s) and value(s) of each field in the document you want to insert.

```
import pymongo

myclient = pymongo.MongoClient('mongodb://localhost:27017/')
mydb = myclient['mydatabase']
mycol = mydb["customers"]

mydictionary = { "name": "Peter", "address": "Lowstreet 27" }

x = mycol.insert_one(mydictionary)

print(x.inserted_id)
```

```
5b1910482ddb101b7042fcd7
```

Insert Multiple Documents

To insert multiple documents into a collection in MongoDB, we use the insert_many() method.

The first parameter of the insert_many() method is a list containing dictionaries with the data you want to insert:

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]

mylist = [
  { "name": "Amy", "address": "Apple st 652"},
  { "name": "Hannah", "address": "Mountain 21"}
]

x = mycol.insert_many(mylist)

#print list of the _id values of the inserted documents:

print(x.inserted_ids)
```

```
[ObjectId('5b19112f2ddb101964065487'), ObjectId('5b19112f2ddb101964065488')
```

12.Explain Python MongoDB Find

Just like the **SELECT** statement is used to find data in a table in a MySQL database.

Find One

To select data from a collection in MongoDB, we can use the find_one() method.

The find_one() method returns the first occurrence in the selection.

Example

Find the first document in the customers collection:

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]

x = mycol.find_one()

print(x)
```

```
{'_id': 1, 'name': 'John', 'address': 'Highway37'}
```

Find All

To select data from a table in MongoDB, we can also use the find() method.

The find() method returns all occurrences in the selection.

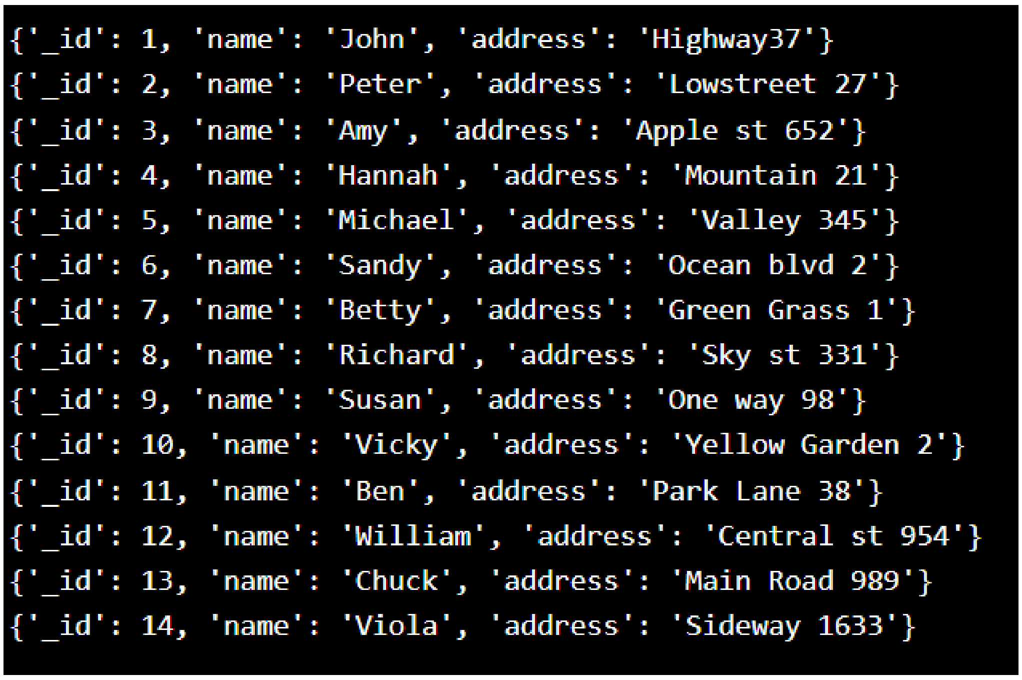
The first parameter of the find() method is a query object. In this example we use an empty query object, which selects all documents in the collection.

No parameters in the find() method gives you the same result as **SELECT *** in MySQL.

Example

Return all documents in the "customers" collection, and print each document:

```python
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]

for x in mycol.find():
  print(x)
```



```
{'_id': 1, 'name': 'John', 'address': 'Highway37'}
{'_id': 2, 'name': 'Peter', 'address': 'Lowstreet 27'}
{'_id': 3, 'name': 'Amy', 'address': 'Apple st 652'}
{'_id': 4, 'name': 'Hannah', 'address': 'Mountain 21'}
{'_id': 5, 'name': 'Michael', 'address': 'Valley 345'}
{'_id': 6, 'name': 'Sandy', 'address': 'Ocean blvd 2'}
{'_id': 7, 'name': 'Betty', 'address': 'Green Grass 1'}
{'_id': 8, 'name': 'Richard', 'address': 'Sky st 331'}
{'_id': 9, 'name': 'Susan', 'address': 'One way 98'}
{'_id': 10, 'name': 'Vicky', 'address': 'Yellow Garden 2'}
{'_id': 11, 'name': 'Ben', 'address': 'Park Lane 38'}
{'_id': 12, 'name': 'William', 'address': 'Central st 954'}
{'_id': 13, 'name': 'Chuck', 'address': 'Main Road 989'}
{'_id': 14, 'name': 'Viola', 'address': 'Sideway 1633'}
```

Multiple choice questions:

1. What is PyMySQL?
   A) A Python package for sending emails
   B) A Python library for working with PostgreSQL databases
   C) **A Python library for interacting with MySQL databases**
   D) A Python framework for web development
2. How do you execute a query using PyMySQL?
   A) run_query()
   B) execute_query()
   C) **execute()**
   D) query()
3. What method is used to fetch all rows from a result set in PyMySQL?
   A) fetch_rows()
   B) **fetch_all()**
   C) fetch()
   D) fetch_row()
4. How do you retrieve the number of rows affected by an SQL statement in PyMySQL?
   A) fetch_row_count()
   B) **affected_rows()**
   C) count_rows()
   D) get_row_count()

5. How do you commit changes to the database in PyMySQL?
   A) **commit()**
   B) apply_changes()
   C) save_changes()
   D) execute_changes()
6. Which method is used to close a cursor object in PyMySQL?
   A) **close()**
   B) end()
   C) finish()
   D) shutdown()
7. How do you handle exceptions in PyMySQL?
   A) PyMySQL does not support exception handling
   B) **Using the try-except block**
   C) Using the **catch** statement
   D) Using the **handle_exception()** function
8. What method is used to rollback changes made to the database in PyMySQL?
   A) **rollback()**
   B) undo()
   C) revert()
   D) cancel()
9. How do you close a connection to the MySQL database in PyMySQL?
   A) **Using the close() method on the connection object**
   B) Using the **shutdown()** method
   C) Connections are automatically closed when the program terminates
   D) Using the **disconnect()** method
10. Which exception is raised when there is a connection error in PyMySQL?
    A) ConnectionError

B) PyMySQLConnectionError
C) **MySQLConnectionError**
D) DatabaseError

11. To install PyMySQL, you can use:
    a) pip install pymysql
    b) pip3 install pymysql
    c) easy_install pymysql
    d) **All of the above**
12. What does the **commit()** method do in PyMySQL?
    a) **Commits the current transaction to the database**
    b) Rolls back the current transaction
    c) Closes the cursor
    d) Reconnects to the database server
13. Which method is used to insert data into a MySQL table using PyMySQL?
    a) insert()
    b) insert_data()
    c) **execute()**
    d) execute_insert()
14. Which method is used to delete data from a MySQL table using PyMySQL?
    a) delete()
    b) remove()
    c) execute_delete()
    d) **execute()**
15. What is MongoDB?
    a) SQL database
    b) **NoSQL database**
    c) Relational database
    d) None of the above
16. Which of the following is a correct way to install PyMongo, the Python driver for MongoDB?
    a) **pip install pymongo**
    b) pip install mongo-python
    c) pip install mongo
    d) pip install mongopy
17. What is PyMongo?
    a) A graphical interface for MongoDB
    b) **A Python package for interacting with MongoDB**
    c) A MongoDB server written in Python
    d) A Python-based query language for MongoDB
18. How do you connect to a MongoDB server using PyMongo?
    a) connect()
    b) create_connection()
    c) connect_to_mongo()
    d) **MongoClient()**
19. Which of the following is used to create a new database in MongoDB using PyMongo?
    a) db_create()
    b) **create_database()**

c) database.create()
d) client.database

20. What does the insert_one() method in PyMongo do?
a) Updates a single document in the collection
b) **Inserts a single document into the collection**
c) Deletes a single document from the collection
d) Retrieves a single document from the collection

21. Which method is used to find documents in a MongoDB collection using PyMongo?
a) find_one()
b) **find()**
c) search()
d) get()

22. What does the delete_one() method in PyMongo do?
a) **Deletes a single document from the collection**
b) Deletes all documents from the collection
c) Deletes the entire collection
d) Deletes a single field from the document

23. How do you sort documents in PyMongo?
a) **sort()**
b) order()
c) order_by()
d) sort_by()

24. Which method is used to limit the number of documents returned in a PyMongo query?
a) **limit()**
b) fetch()
c) get_limit()
d) retrieve()

25. In PyMongo, which method is used to drop a collection?
a) drop_collection()
b) delete_collection()
c) remove_collection()
d) **drop()**