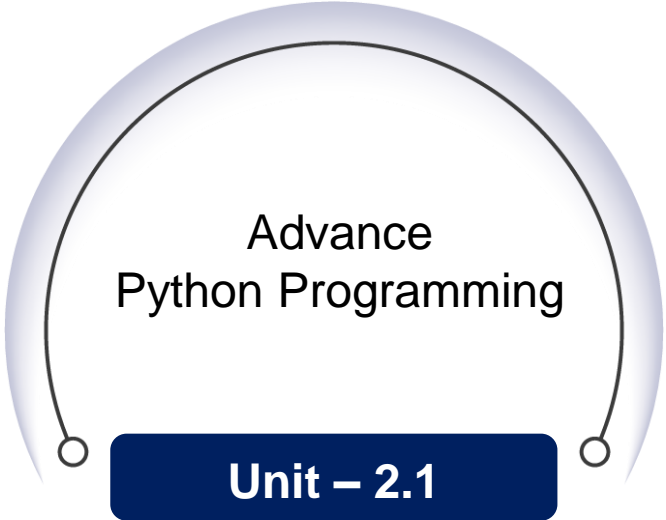


Module - 1

Introduction to Python with Data Structure



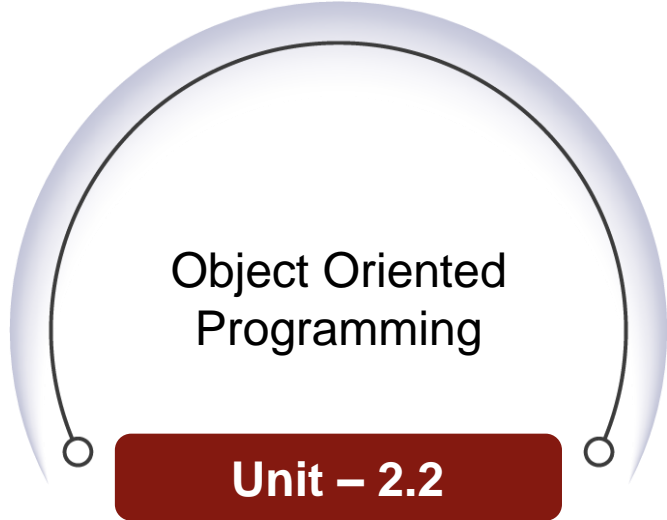
Units for Discussion



Advance
Python Programming

The diagram for Unit 2.1 consists of a light blue semi-circular arc with a black outline. Inside the arc, the text 'Advance Python Programming' is centered. At the bottom of the arc, there are two small white circles connected by a horizontal dark blue rectangular bar containing the text 'Unit – 2.1'.

Unit – 2.1



Object Oriented
Programming

The diagram for Unit 2.2 consists of a light blue semi-circular arc with a black outline. Inside the arc, the text 'Object Oriented Programming' is centered. At the bottom of the arc, there are two small white circles connected by a horizontal dark red rectangular bar containing the text 'Unit – 2.2'.

Unit – 2.2



Error & Exception
Handling

The diagram for Unit 2.3 consists of a light blue semi-circular arc with a black outline. Inside the arc, the text 'Error & Exception Handling' is centered. At the bottom of the arc, there are two small white circles connected by a horizontal purple rectangular bar containing the text 'Unit – 2.3'.

Unit – 2.3

Unit – 2.1

Advance Python programming



DISCLAIMER

The content is curated from online/offline resources and used for educational purpose only.



Quora



UBER

Learning Objectives:

- Python Conditional Statements
- Python loops
- Python range(), break and continue
- Function In Python
- Library & library installation
- Module in python
- Summary



Source :

Python Conditional Statements

There are three types of conditional statement in python

- If
- Else
- Elif

Python supports the usual logical conditions from mathematics:

Equals: $a == b$

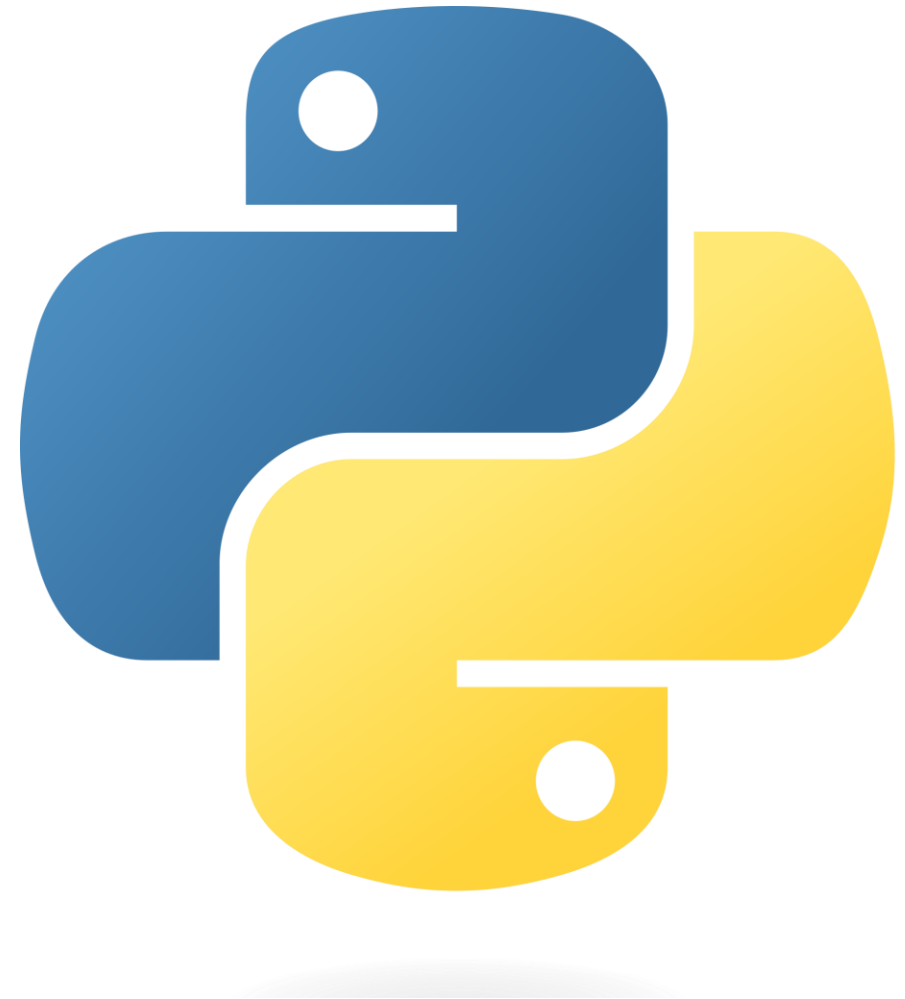
Not Equals: $a != b$

Less than: $a < b$

Less than or equal to: $a \leq b$

Greater than: $a > b$

Greater than or equal to: $a \geq b$



1. If Statement

The If statement is the most fundamental decision-making statement, in which the code is executed based on whether it meets the specified condition. It has a code body that only executes if the condition in the if statement is true. The statement can be a single line or a block of code.

Example:

```
a = 33
b = 200
if b > a:
    print("b is greater than a")
```

Output:

b is greater than a

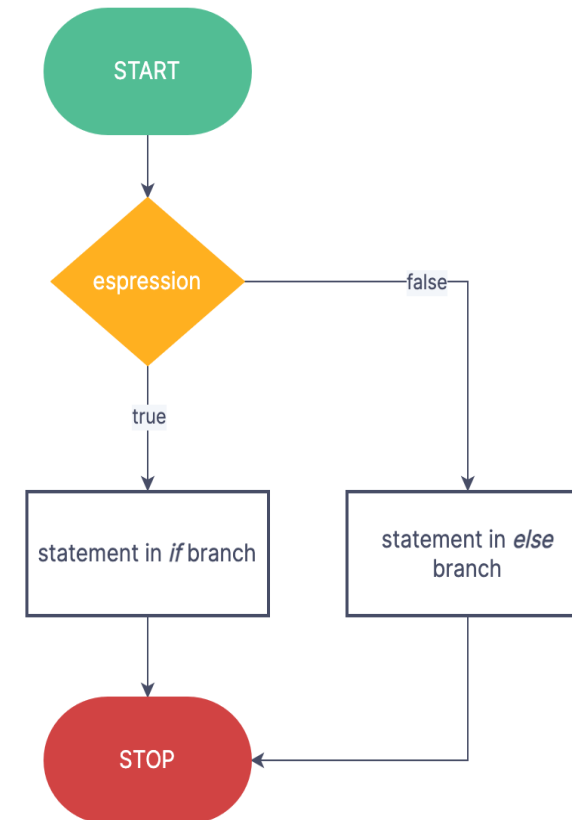
Example:

```
a = 33
b = 200
if b > a:
    print("b is greater than a")
# you will get an error
```

Output:

```
print("b is greater than a")
    ^
```

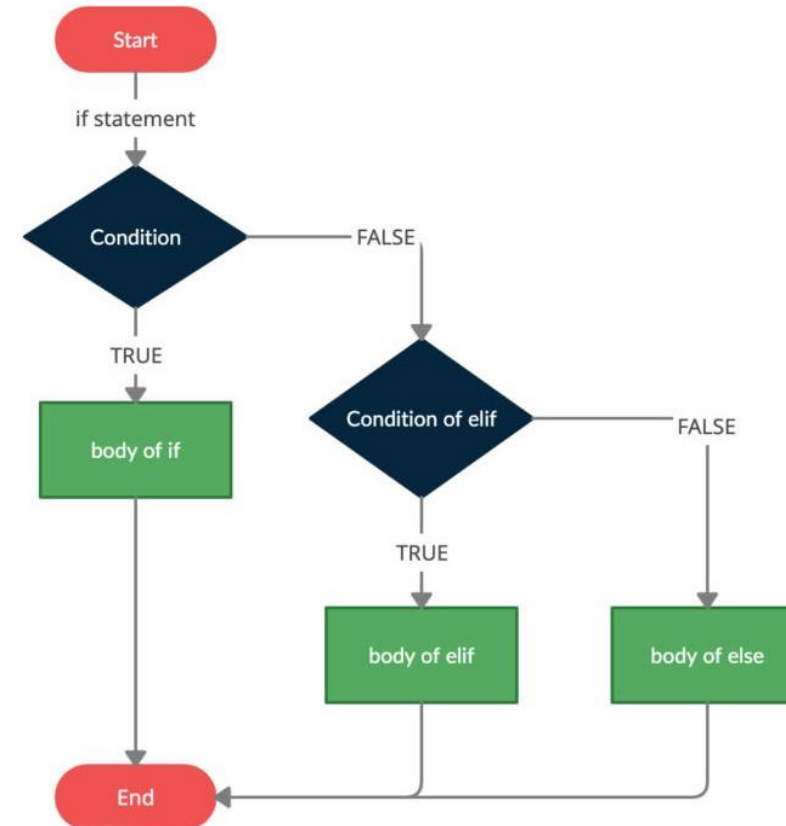
Indentation Error: expected an indented block



3. Elif Statement

In this case, the If condition is evaluated first. If it is false, the Elif statement will be executed; if it also comes false, the Else statement will be executed.

```
if condition1:  
    statements(s) # body of if  
elif condition2:  
    statements(s) # body of elif  
elif condition3:  
    statements(s) # body of elif  
else:  
    statements(s) # body of else
```



Source :

2. Else Statement

- This statement is used when both the true and false parts of a given condition are specified to be executed.
- When the condition is true, the statement inside the if block is executed; if the condition is false, the statement outside the if block is executed.

Example:

a = 200

b = 33

if b > a:

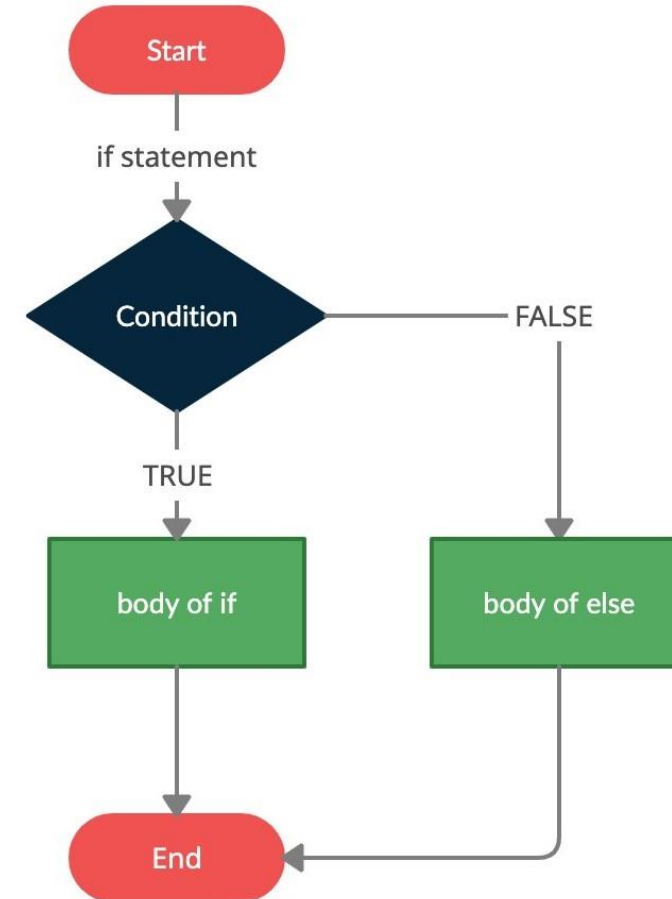
 print("b is greater than a")

else:

 print("b is not greater than a")

Output:

b is not greater than a

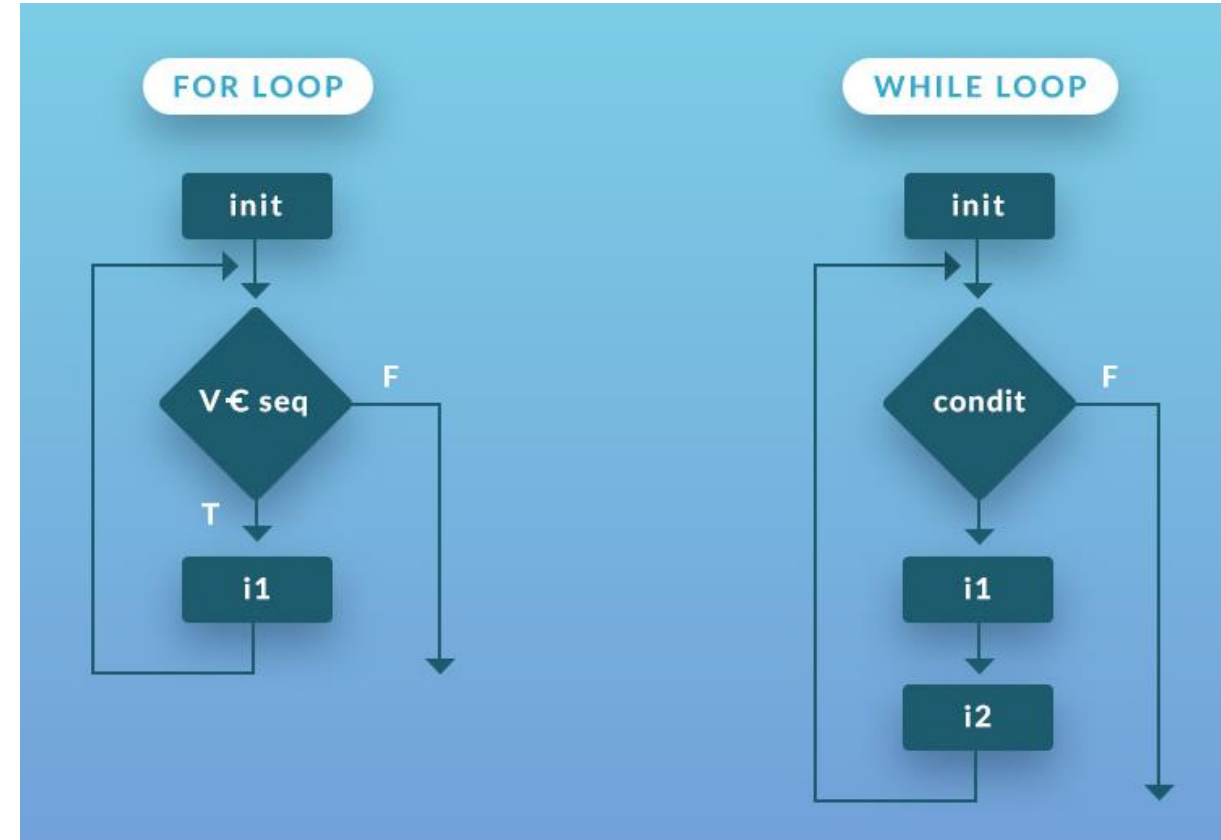


Source :

Python Loops

In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times. The diagram illustrates a loop statement –

1. While loop
2. For loop



Source :

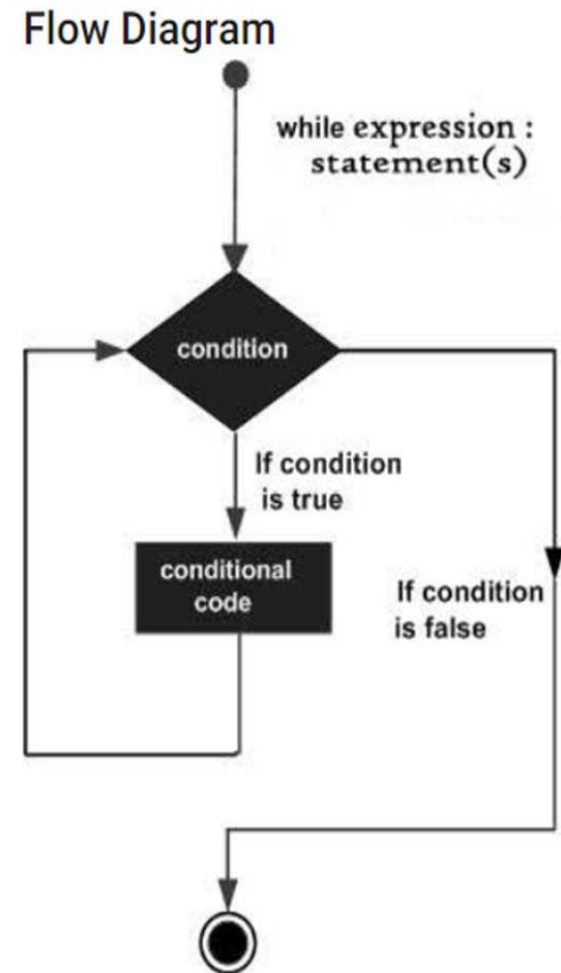
Types of Loops

1. While Loop

A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

Syntax :

```
while expression: statement(s)
```



Source :

Types of Loops

1. While Loop

Example

```
count = 0
while (count < 9):
    print('The count is:', count)
    count = count + 1
print("Good bye!")
```

Output

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8
Good bye!
```

While Loop With Else

- Python supports to have an else statement associated with a loop statement.
- If the else statement is used with a while loop, the else statement is executed when the condition becomes false.

Example

```
count = 0
while count < 5:
    print(count, " is less than 5")
    count = count + 1
else:
    print(count, " is not less than 5")
```

Output

```
0 is less than 5
1 is less than 5
2 is less than 5
3 is less than 5
4 is less than 5
5 is not less than 5
```

Source :

Types of Loops

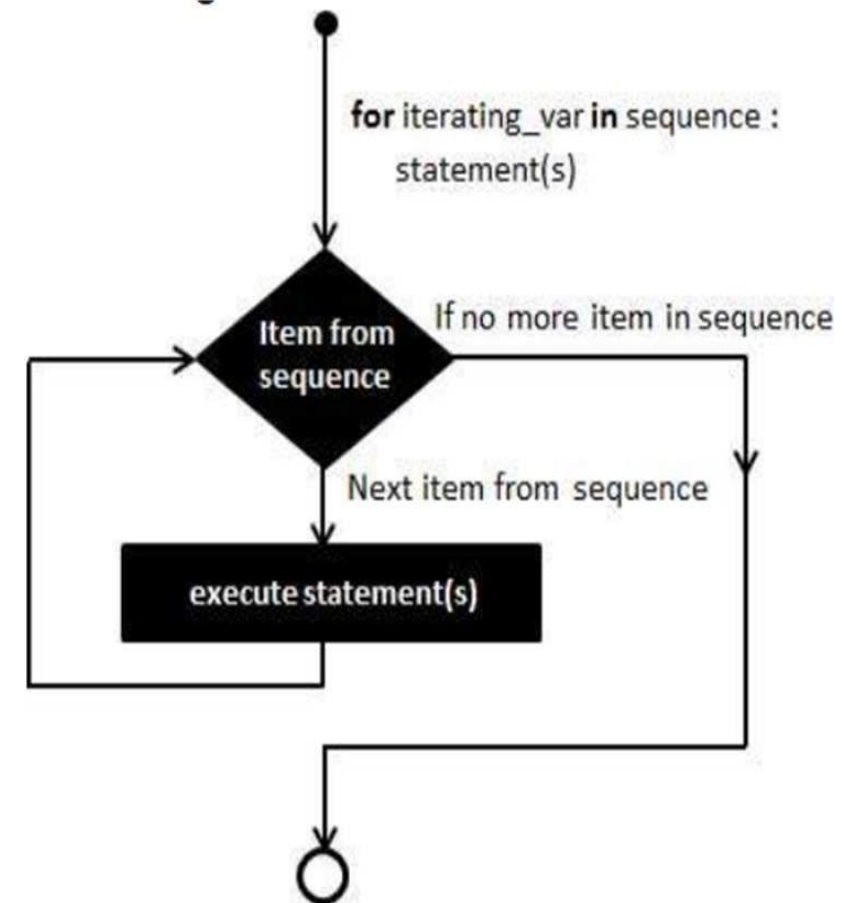
2. For Loop

It could iterate over the items of any sequence, such as a list or a string.

Syntax :

for iterating_var **in** sequence:

statements(s)



Source :

Types of Loops

2. For Loop

Example

```
for letter in 'Python':    # First Example
    print('Current Letter :', letter)

fruits = ['banana', 'apple', 'mango']
for fruit in fruits:       # Second Example
    print('Current fruit :', fruit)

print("Good bye!")
```

Output

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
Current fruit : banana
Current fruit : apple
Current fruit : mango
Good bye!
```


For Loop With Else

- Python supports to have an else statement associated with a loop statement.
- If the else statement is used with a for loop, the else statement is executed when the loop has exhausted iterating the list.

Example

```
for num in range(10,20):      #to iterate between 10 to 20
    for i in range(2,num):    #to iterate on the factors of the number
        if num%i == 0:        #to determine the first factor
            j=num/i            #to calculate the second factor
            print('%d equals %d * %d' % (num,i,j))
            break              #to move to the next number, the #first FOR
    else:                      # else part of the loop
        print(num, 'is a prime number')
        break
```

Output

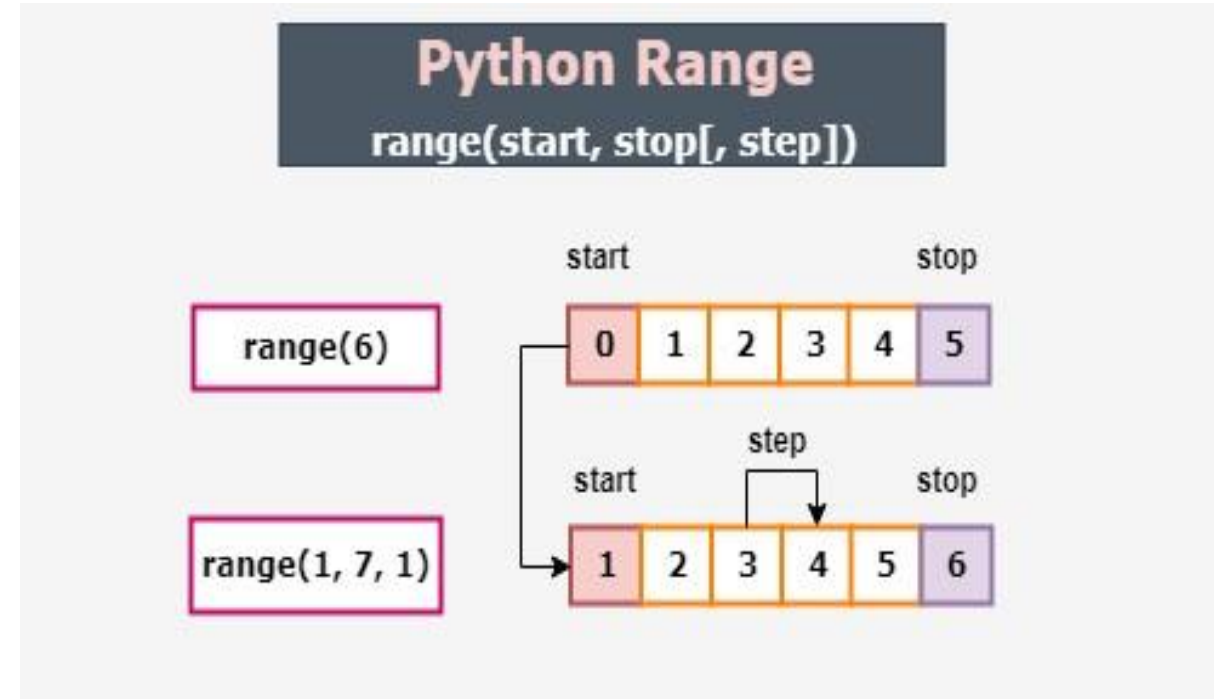
```
10 equals 2 * 5
12 equals 2 * 6
14 equals 2 * 7
16 equals 2 * 8
18 equals 2 * 9
```

Python range()

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

Syntax :

`range(start, stop, step)`



Source :

Range Function

The `range()` function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

Syntax:

`range(start, stop, step)`

Example1:

```
for n in range(4):  
    print(n)
```

Output:

0
1
2
3

Example2:

```
x = range(1, 9, 2)  
for n in x:  
    print(n)
```

Output:

1
3
5
7

Pass Statement

The pass statement is used as a placeholder for future code. When the pass statement is executed, nothing happens, but you avoid getting an error when empty code is not allowed. Empty code is not allowed in loops, function definitions, class definitions, or in if statements.

Example: This will give no any error/output because of pass statement

```
a = 33  
b = 200  
if b > a:  
    pass
```

Python Delete Statement

The **del keyword** is used to delete objects. In Python everything is an object, so the **del keyword** can also be used to delete variables, lists, or parts of a list etc.

In [15]:

```
1 x = ["physics", "chemistry", "maths"]
2
3 del x[0]
4
5 print(x)
```

```
['chemistry', 'maths']
```

Python Return Statement

"A return statement is used to end the execution of the function call and “returns” the result (value of the expression following the return keyword) to the caller." The statements after the return statements are not executed. If the return statement is without any expression, then the special value None is returned. A return statement is overall used to invoke a function so that the passed statements can be executed.

```
1  # A Python program to return multiple values from a method using list
2
3  def fun():
4      str = "Hello world"
5      x = 10
6      return [str, x];
7
8  # Driver code to test above method
9  list = fun()
10 print(list)
```

```
['Hello world', 10]
```

PYTHON IMPORT STATEMENT

The import keyword is used to import modules.
For example, we are importing math module and want to use sqrt function.

```
1 import math
2
3 x = math.sqrt(64)
4
5 print(x)
```

8.0

Python Continue And Break Statement

Break Statement

We use the break statement when we want to break the loop after specific iteration

```
i = 1
while i < 9:
    print(i)
    if i == 3:
        break
    i += 1
```

1
2
3

Continue Statement

The continue statement is used when we want to skip any specific iteration.

```
i = 0
while i < 10:
    i += 1
    if i == 3:
        continue
    print(i)
```

1
2
4
5
6
7
8
9
10

Lab - 1

Python Control Statements

Function In Python

In Python **function is just a block of code**, a function is a group of related statements that perform a specific task. Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable.

Syntax of Function

```
def function_name(parameters):  
    """docstring"""  
    statement(s)
```

Example of a Function

```
def greet(name):  
    """  
    This function greets to  
    the person passed in as  
    a parameter  
    """  
    print("Hello, " + name + ". Good morning!")
```

Defining And Calling A Function

Function in Python is defined by the "def" keyword followed by the function name and parentheses (). To call a function we simply type the function name with appropriate parameters.

```
# Example to Illustrate Function
# Function Definition

def show(n):
    print("Value is",n)
a=10
show(a)                # Calling a Function

def greetings(s):
    print(f"Hello Mr. {s},Good Morning!")
greetings("S.C.Bose")    # Calling a Function
```

```
Value is 10
Hello Mr. S.C.Bose,Good Morning!
```

Scope and Variables

1. Local variable

Variable which are inside the scope

```
# Example of Local variable
def test1():
    a = 10 # here a is local variable
    print("Value of a inside function or local variable is: ", a)

# outside the function
a = 20
test1()
print("Value fo a Outside Function", a)
```

```
Value of a inside function or local variable is:  10
Value fo a Outside Function 20
```

2. Global Variables

A variable declared outside of the function or in global scope is known as global variable.

```
# example of global variable  
  
# global variable as it is outside the scope  
x = "awesome"  
  
# scope start  
def myfunc():  
    print("Python is " + x)  
  
myfunc()  
# scope ended
```

Python is awesome

Parameters and Arguments

Parameters in python

A parameter is nothing just a **variable** which is defined under the parentheses during function defining. Simply they are written when we declare a function. For example: `sum(a,b)`

Arguments in python

Arguments nothing just the **values** of the parameters like if a,b are the parameters then (1,2) will be the values of a,b respectively. For example: `sum(1,2)`

Example:

Here a,b are the parameters

```
def sum(a,b):
```

```
    print(a+b)
```

```
sum(1,2)
```

1. Arbitrary Arguments - *Args

We use it when there are unknown number of arguments are there

Example:

```
In [20]: 1 def my_function(*Student):  
          2     print("The brilliant student is " + Student[2])  
          3  
          4 my_function("Ravi", "kartik", "vinay")
```

```
The brilliant student is vinay
```

2. Keyword Arguments (**Kwargs)

We use this when there are unknown number of keywords and arguments in our code.

Example: In the following program we are using (**kwargs) for taking keyword arguments

```
1 def my_function(**Student):  
2     print("The brilliant student is " + Student["lname"])  
3  
4 my_function(fname = "Ravi", lname = "Sharma")
```

The brilliant student is Sharma

Lamda Function

Python Lambda Functions are anonymous function means that the function is without a name. As we already know that the def keyword is used to define a normal function in Python. Similarly, the lambda keyword is used to define an anonymous function in Python.

Syntax

lambda arguments: expression

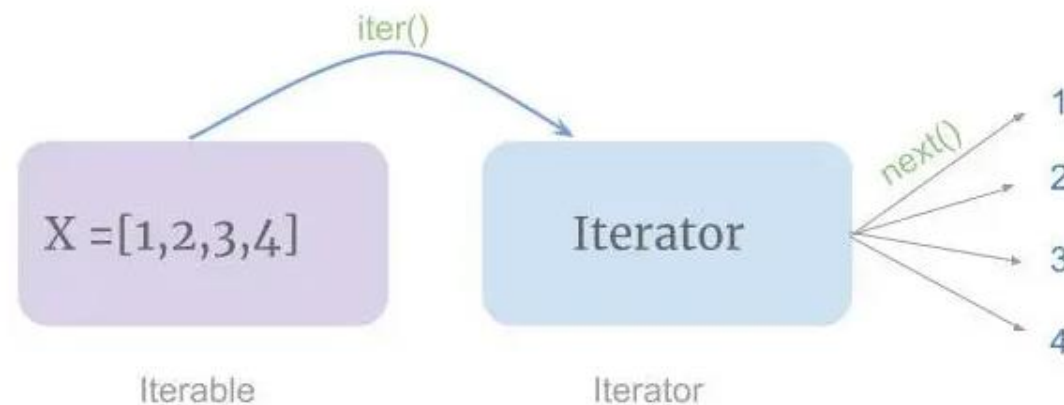
```
x = lambda a: a + 10  
print(x(5))
```

15

Iterator

An iterator is an object that can be iterated upon. This means that you can traverse through all the values in the object. Iterators are used in both for loops and while loops.

An iterator in Python is an object that contains a countable number of elements that can be iterated upon. In simpler words, we can say that Iterators are objects that allow you to traverse through all the elements of a collection and return one element at a time.



Generator

In Python, a generator is a function that returns an iterator that produces a sequence of values when iterated over.

Generators are useful when we want to produce a large sequence of values, but we don't want to store all of them in memory at once.

```
def generator_name(arg):  
    # statements  
    yield something
```

Here, the yield keyword is used to produce a value from the generator.

Decorator

A Python decorator is a function that takes in a function and returns it by adding some functionality.

In fact, any object which implements the special `__call__()` method is termed callable. So, in the most basic sense, a decorator is a callable that returns a callable.

Basically, a decorator takes in a function, adds some functionality and returns it.

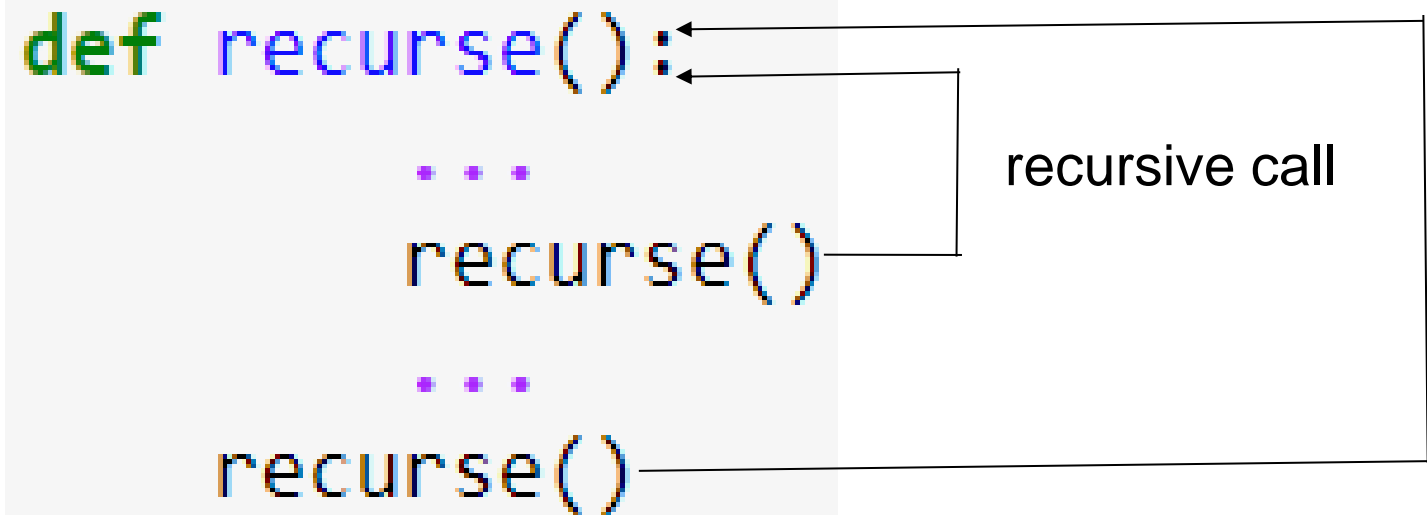
```
def make_pretty(func):  
    def inner():  
        print("I got decorated")  
        func()  
    return inner  
  
def ordinary():  
    print("I am ordinary")  
  
# Output: I am ordinary
```

Recursive Functions

Recursion is the process of defining something in terms of itself.

In Python, we know that a function can call other functions. It is even possible for the function to call itself. These types of construct are termed as recursive functions

```
def recurse():  
    ...  
    recurse()  
    ...  
    recurse()
```



The diagram illustrates recursive calls within a function definition. It shows a Python function `def recurse():` with three lines of code: `...`, `recurse()`, and `...`, followed by another `recurse()`. A bracket on the right side of the code block, labeled "recursive call", points to the `recurse()` calls. A line from the top of the bracket points to the first `recurse()` call, and another line from the bottom of the bracket points to the second `recurse()` call.

Higher Order Function

- A higher-order function is a function that can receive other functions as arguments or return them.
- For example, we can pass a lambda function to the built-in `sorted()` function

```
d = {'Oil' : 230, 'Clip' : 150, 'Stud' : 175, 'Nut' : 35}  
# lambda takes a dictionary item and returns a value  
d1 = sorted(d.items( ), key = lambda kv : kv[1])  
print(d1)
```

- To facilitate functional programming, Python provides three higher-order functions—`map()`, `filter()`, and `reduce()`.

Lab - 2

Iterators, Generators, Decorators and Higher Order Functions in Python

What is a Library?

A library is a collection of pre-combined codes that can be used iteratively to reduce the time required to code. They are particularly useful for accessing the pre-written frequently used codes, instead of writing them from scratch every single time. Similar to the physical libraries, these are a collection of reusable resources, which means every library has a root source. This is the foundation behind the numerous open-source libraries available in Python.

Some of the Popular Python Libraries;

1. Numpy
2. Pandas
3. Seaborn
4. Keras
5. Matplotlib
6. Scikit Learn
7. Tensorflow



Library Installation

Install the required package using PiP

```
1 pip install numpy
```

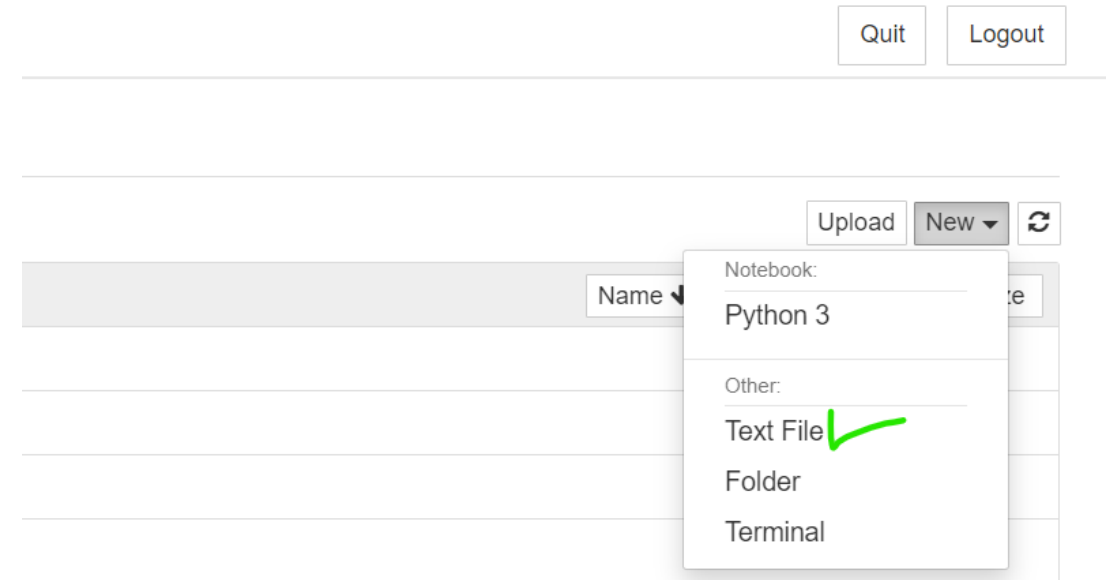
Requirement already satisfied: numpy in c:\users\prana\anaconda3\lib\site-packages (1.20.1)
Note: you may need to restart the kernel to use updated packages.

What is Module in Python?

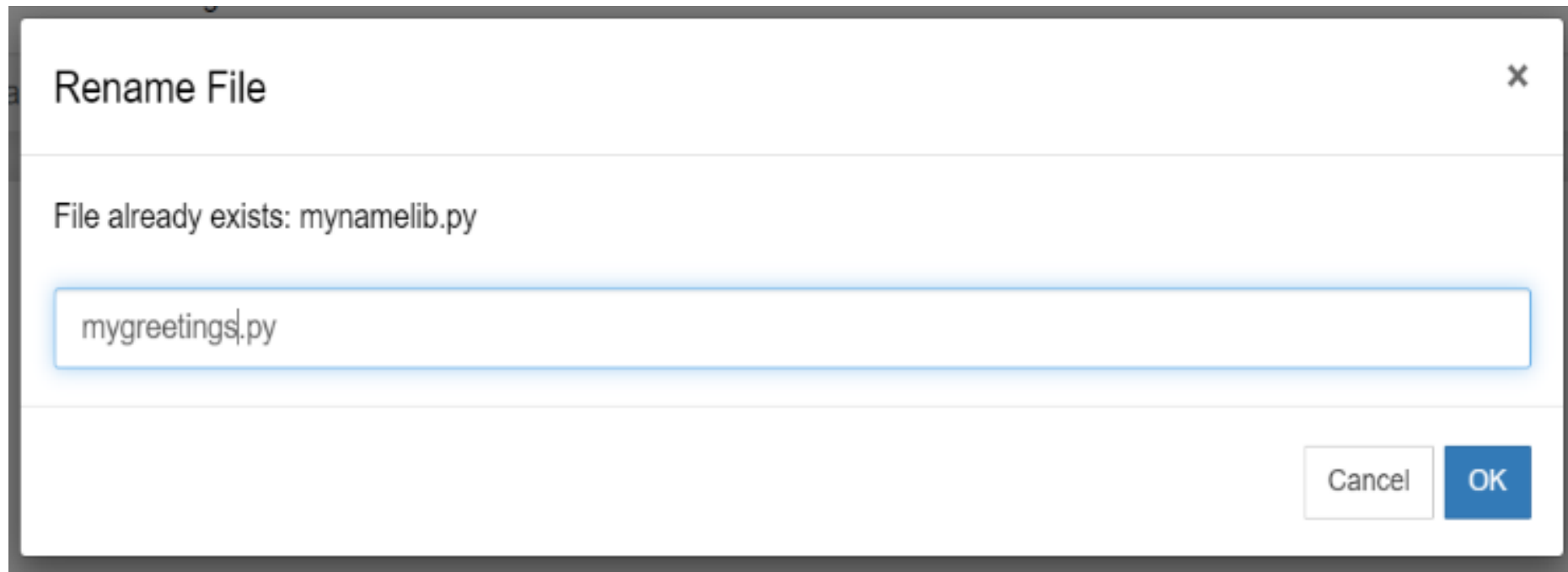
We can Consider a module just same as a code library. A file containing a set of functions you want to include in your application. There are various built-in modules in python like: Math, time, random etc. Apart from that we can also create our own module.

So now we will look how to create our own module or own library.

**Step1: Into your jupyter
Notebook New >> Text file**



Step2 : Naming/Renaming the module by .py extension



Step3: Creating A Function Name Greeting In Module And Write The Same Function Or You Can Give Any

 jupyter mynamelib.py ✓ 16 minutes ago

File

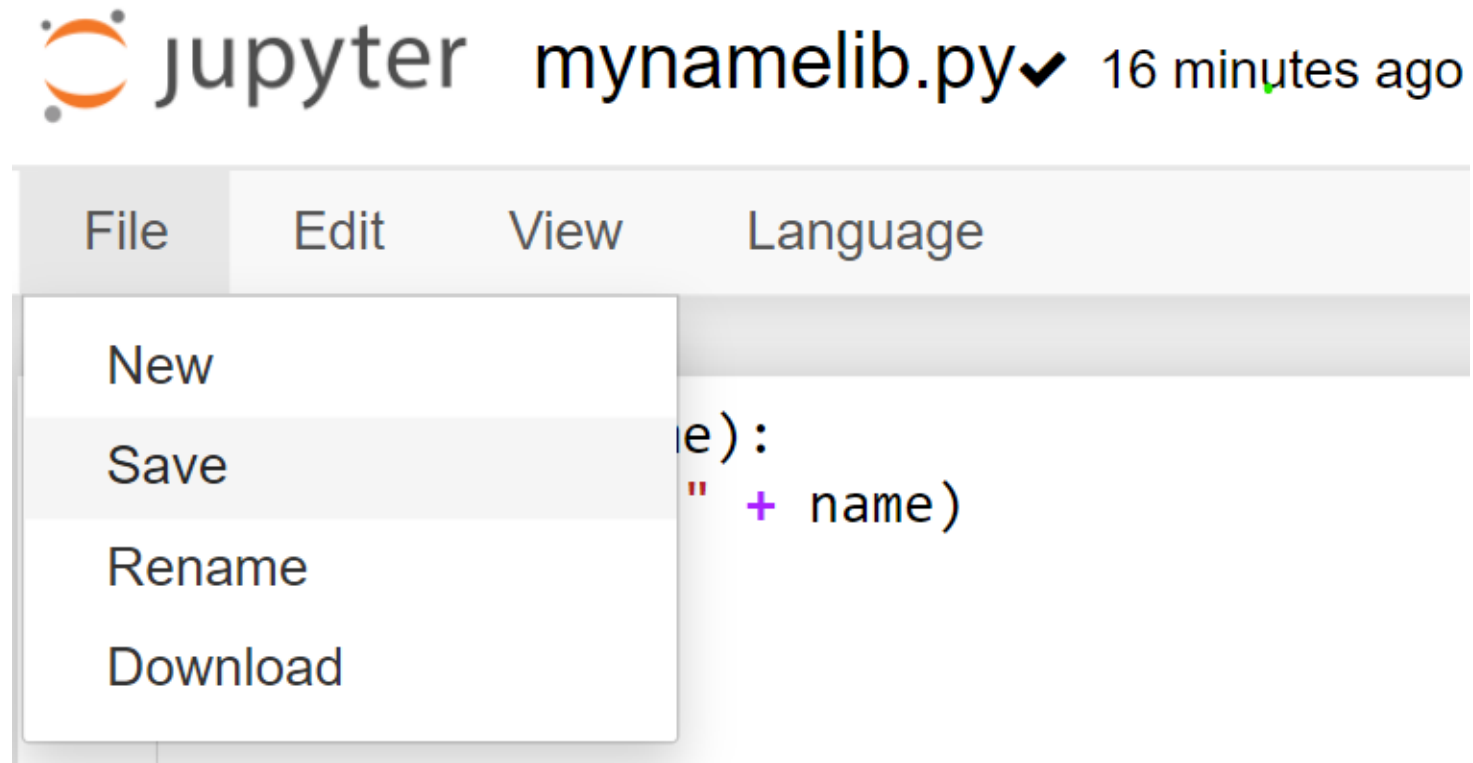
Edit

View

Language

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

Step4: Saving the file by giving the extension txt to .py extension.



Note: Our module is created now we are ready to use our own module. So let's use our module.

Now How to Use the Module

Use a Module: import the module and call the greeting function under the module and see the output.

Note: When using a function from a module, use the

syntax: *module_name.function_name*.

```
1 import mynamelib
2 mynamelib.greeting("pranay")
```

Hello, pranay

Variables in Module

The module can contain functions, as already described, but also variables of all types (arrays, dictionaries, objects etc). Now let's take another example in which we will create various variables in module name mymodule.py. Save this code in the file mymodule.py

Step1: Create a module

Here we are creating three variable name, age and country.

```
person1 = {  
    "name": "Prince",  
    "age": 17,  
    "country": "India"  
}
```

Step2: Use the module: Import the module named mymodule, and access the person1 "age" dictionary:

```
import mymodule  
a = mymodule.person1["age"]  
print(a)
```

ouput:

17

How to use any Built-in Modules

There are several built-in modules/library in Python like: platform, math, datetime, random, pandas etc. which you can import whenever you like.

Example Import and use the **platform** module:

```
1 import platform
2
3 x = platform.system()
4 print(x)|
```

Windows

Here! we can check the OS of our system

Using the dir() Function

We can see easily list all the functions available in our built-in module or created by ourself, through dir() function, let's look what are functions are there in math module.

```
1 import math
2
3 x = dir(math)
4 print(x)
5
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
```

Lab - 3

Function , Module and Creation

Lab - 4

Conditional execution with scenario

Conclusion

- Control flow can make your program take different paths based on certain conditions, enabling you to create complex logic and decision-making processes.
- Implementing functions allow you to encapsulate code into reusable blocks, making your programs more modular and easier to read, maintain, and debug.
- Understanding Installing libraries allows you to leverage a vast ecosystem of pre-built tools and functionalities
- Modules and packages allow you to build complex systems by breaking down your code into manageable components.





Let's Start

Quiz

1. Which of the following is NOT a valid Python conditional statement?

- a) if
- b) else
- c) elseif
- d) elif



Answer: C
elseif

Quiz

2. What will be the output of the following code snippet?

```
for i in range(3):  
    print(i)
```

- a) 0 1 2
- b) 1 2 3
- c) 0 1 2 3
- d) 1 2

Answer: A

0 1 2



Quiz

3. What does the continue statement do in a loop?

- a) It exits the loop completely
- b) It skips the rest of the code inside the loop for the current iteration only
- c) It pauses the loop until the user provides input
- d) It restarts the loop from the beginning



Answer: B

It skips the rest of the code inside the loop for the current iteration only

Quiz

4. What is the correct way to define a function in Python?

- a) `function myFunction():`
- b) `def myFunction():`
- c) `func myFunction():`
- d) `myFunction():`

Answer: B

`def myFunction():`



Quiz

5. Which command is used to install a library in Python?

- a) install library_name
- b) pip install library_name
- c) library install library_name
- d) python -install library_name



Answer: B

pip install library_name

References

- https://www.tutorialspoint.com/python/python_variable_types.htm
- <https://www.edureka.co/blog/variables-and-data-types-in-python/>
- <https://data-flair.training/blogs/python-variables-and-data-types/>
- <https://www.programiz.com/python-programming/variables-datatypes>
- <https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python>
- <https://www.python.org/doc/essays/blurb/>
- https://www.w3schools.com/python/python_conditions.asp
- <https://www.toppr.com/guides/python-guide/tutorials/python-flow-control/if-elif-else/python-if-if-else-if-elif-else-and-nested-if-statement/>

Thank You...!