KPMG

Day 3 — 09-08-2023

LOOPS

For — finite

for(i=0; i<=10; i++)
{
}

i → loop variable

(1) data type of i for each Iteration → int

Python

Iterations → len(Iterableobject)

for loop_var in Iterobject:

- ✓ str — Iterables
- ✓ list — many ways
- ✓ tuple — 2D obj...
- dict — find obj
- set

range(b)

a = 0
b = b = 5
c = 1

0 : 5 : 1
↓
0
1
2
3
4

for i in range(5):
    print(i)

range(a, b, c)

a = 3
b = 10
c = 3

for i in range(3, 10, 3):
    print(i)

⇒ 3, 6, 9

Prob 1: W.A.P to calculate the factorial of 0-20 values & Handle the odd Inputs.

for / While ?

$\lfloor 5 \rightarrow 1 \times 2 \times 3 \times 4 \times 5$ } 120

$5! \rightarrow 5 \times 4 \times 3 \times 2 \times 1$

* $1! \rightarrow 1$

$0! \rightarrow 1$

* +|- → Any value 0, 1, 3

*/÷ → Except 0 any other value

∞

While cond :

True

a<b

a>b

a<=b

a>=b

a!=b

a==b

a

After use of variable

i++ → Post ↑] i = i+1 or i += 1

++i → Pre ↑

before use of the variable.

i-- Post ↓ } i -= 1 or i = i-1

--i Pre ↓

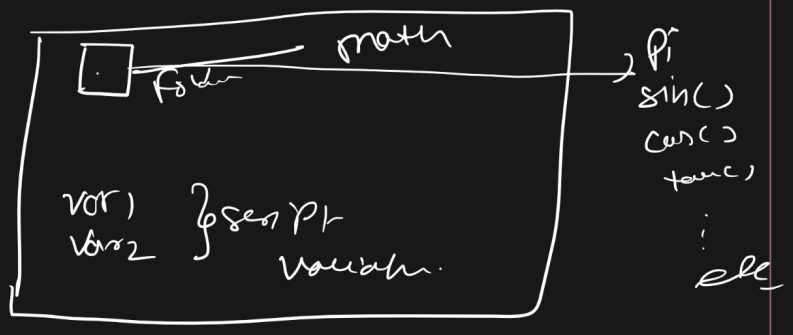
The `break` statement is used to terminate the loop immediately when it is encountered.

The `continue` statement is used to skip the current iteration of the loop and the control flow of the program goes to the next iteration.

1 Import

(a) Import ModuleName

Import math



(1) Module Import as a directory (folder) with its python file with its Module Name

(2) Need Refrence to call them.

math.PI
math.sin()

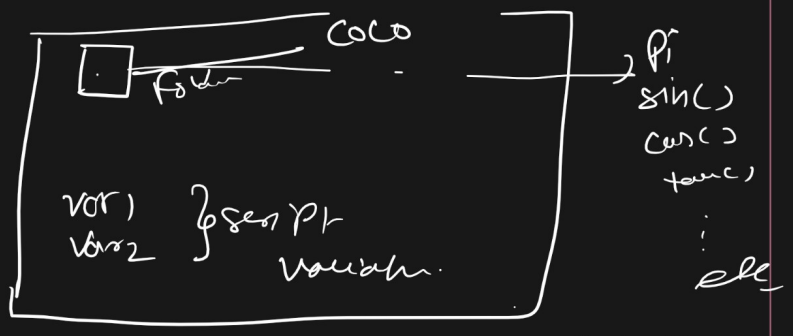
(3) Many Worked.

1 Import #ShortName ✓

Import #rename ✓

(b) Import ModuleName as Shortnam

Import math as coco



(1) Module Import as a directory (folder) with its python file with its Module Name

(2) Need Refrence to call them.

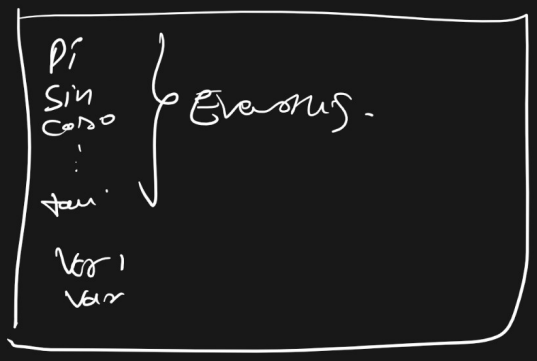
coco.PI
coco.sin()

(3) Many Worked.

2. from

(a) from modulename import * — Everything.

from math import *

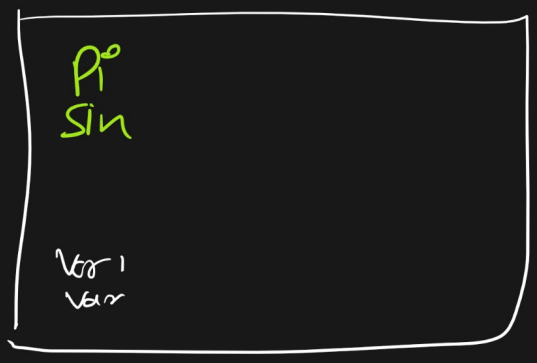


* No directly created.
* No Retrieve required.
* Memory still wasted.

2. from

(b) from modulename import X1, X2

from math import pi, sin



* No directly created.
* No Retrieve required.
* No Memory wasted.

random

randint(a,b)

$a <= N <= b$ → Integer Number

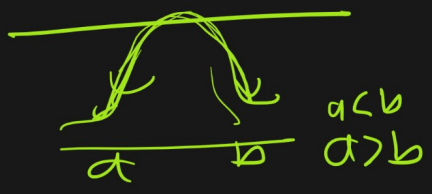
↓ Random Integer value

random( )

$0.0 <= N <= 1.0$

↳ Random float value

Uniform (a,b)



$a <= N <= b$

$a >= N >= b$

any value → integer / float → -ve / +ve

Random float Value

randrange (a,b,c)

a = 10

b = 22

c = 3

10, 13, 16, 19, ~~22~~

diff Import & from syntax

import	from
* Import Everything	* We can manage what we need to import (memory efficient)
* Perform slower	Perform high
* module imported as directory	* No directory created
* Need refer of that directory[.] to call them	* We can call them directly. No need refer.

List

* list object can store different kind of data types.
* list object Iterable & we can apply Indexing | slicing
* list object is ordered (seq)

* list object is mutable C U R D ✓ ✓ ✓ ✓

* [ ] ← empty   ['Test'] ← one value
* [1, 1, 2, 'Test']   multiple values list

List

C

* []
* [1.2]
* [1,2,3, 4.5, 'Tom']

U

* a[2] = 'Newvalue'
* a.append(obj)
* a.insert(Index, Obj)
* a.extend(Iter)

R

* Indexing a[3]
* Slicing a[3:]
* Loop
  for i in a:
  print(a)

D

* del a[Index]
* a.remove(value)
* a.pop()
* a.pop(Index)

`a.append(object)`

object → int, float, Iterable obje → str, list, tuple, dict, set, zip, file, etc

a = [1,2,3] — (3)

a.append(45)

[1,2,3,45] — (4)

a.append([12,13,14])

[1,2,3,[12,13,14]] — (4)

`a.insert(index, object)`

object → Same

a = [1,2,3] — (3)
(0 1 2)

a.insert(1, 45)

→ [1,45,2,3] — (4)
(0 1 2 3)

a.insert(1, [45,36])

→ [1,[45,36],2,3] — (4)
(0 1 2 3)

`a.extend(iterable)`

Iterable object → str, list, tuple, dict, set, zip, etc

a = [1,2,3] — (3)

a.extend([45])

[1,2,3,45] — (4)

a.extend([45,85,90])

[1,2,3,45,85,90] — (6)

```
a=["Test",1,"Test",1.2,"Test",(1,2),"Test",[89,67],"Test","Test",56.6,"Test"]
print(a)
print(id(a),type(a),len(a))
print("==========delete=======")
#create a list of Value "Test" index numbers . ?
#[0,2,4....]
#Delete all the "Test" Values from given list ?
```

append

Input → int
float
any Iterable object

as a last element

[1,2,3]

[1,2,3,obj] +1

extend

only Iterable objet

→ [1,2,3] → 3
old

(1,2,3,4)

→ [1,2,3,1,2,3,4]
New → 7