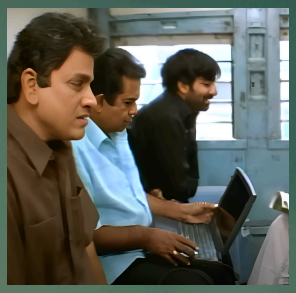
EE.iNfo

MemeCODE - EBook

- coding with memes and fun -



Before

What you get :

1. Python in 21 days
2. DSA Sheet Salary wise
3. Free Resume Projects

Hard Work beats Talent
when talent does not work hard

IMP NOTE

1. Mundhuga E-book ni nammi ekkada varaku vachina varaki heartly welcome.

2. E book lo rasina prathi line 5 years experience tho rasindhi.

3. So ekkadunna prathi word me interview ki help aithene memu include chesam no extra bluff even 1 word also. so deentlo a okka page ni meru skip cheyyakandi.

4. College placements on campus/off campus prathi interview oka coding round oka technical round untundhi. Edhi clear, Evi clear cheyyalante meku edhoka language lo coding problems solve cheyyagalagali. Danikosame ee ebook.

5. Chala mandhi coding priority echi projects ni ignore chestaru.Kani you should balance both. Endhukante projects anevi nv real timelo work cheyygalavu ane proof. So ekkadunna projects meru use cheskondi resume lo pettukondi. Try to add extra feautres to existing code if possible using aitools like cursor, windsurf.

6. chalamandhiki unde brama entante python is not acceptable in interviews, Actually it is not true, interviews code shortcuts not accepted evi c++, java lo kuda untai.
Python ne endhuku ante? AI, Automation, Backend lo use chese major language python assalu AI antene python. Eppudu a software company chuskunna kachitanga ai related job roles unnai like ai engineer, data science.

7. As a fresher ne code lo language kanna ne logic chustaru. Endhukante language edaina logic okkate and language is just a medium like telugu, hindi.

8. Ebook lo ichina prathi task, question ni solve cheyyandi ave mee interview ni clear cheyyadaniki direct or indirect ga help chestai.

9. 2025 lo emaina aisafe jobs unnaya ante, avi python thone link ayyi unnai endhuko meeku telsu AI = Python.

10. Meeku a doubt unna Email ki send cheyyandi “eefriends1729@gmail.com”. We are always ready to help you and guide you comrade.

ENJOY – LEARN – GROW

INDEX

Note: Click on respective content to navigate

EE.iNfo

PYTHON in 21 Days

-Crack any coding interview-

DAY 01 : EE.INFO

Run your first PYTHON Program

50% Learning -50% Practice.

First Program in Python:

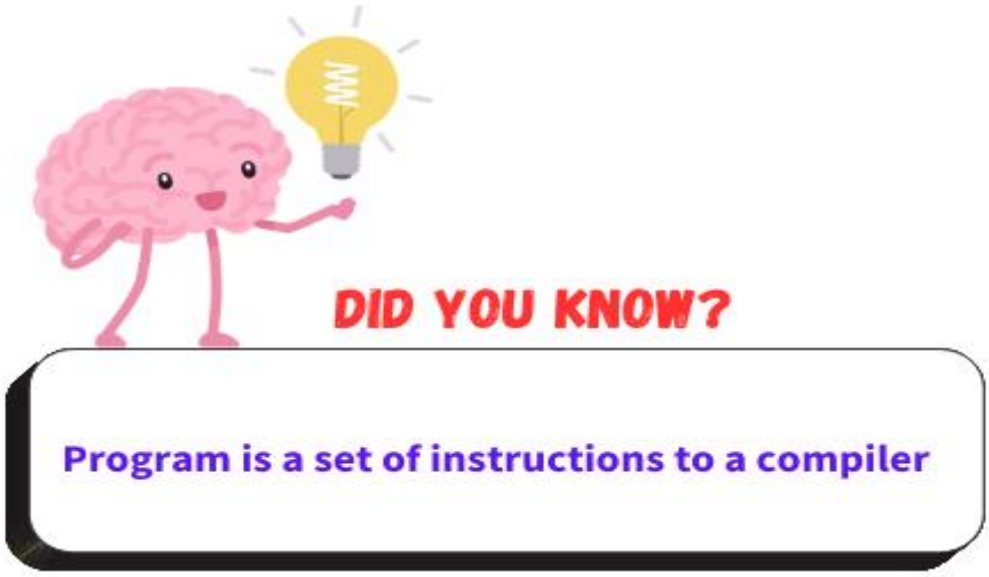

```
print("Hello World")
```

- **OUTPUT: Hello World**

Second Program in Python:

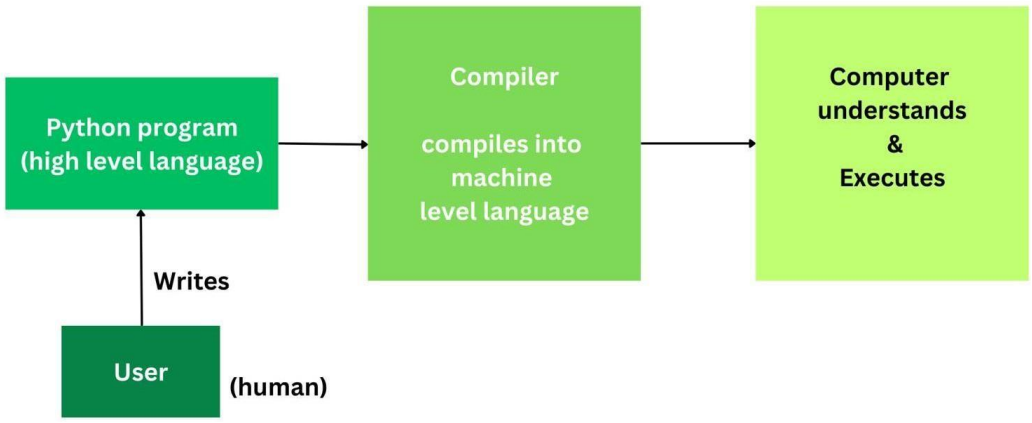
```
a=10
b=20
print(a+b)
```

- **OUTPUT: 30**



Q. Where, How to run first Python Program?

1. VS Code: https://youtu.be/yBopFbgeGaY?si=cP0s6Abb2exHFhU
2. Python IDE: https://youtu.be/riL_xn6BKD8?si=5hCafOMa0lOHtdLr



Day 01 Summary:

- Executing first python program
- మీ first python program ని execute చేసారా?
  లేకపోతే ఇక్కడ click చెయ్యండి: https://www.programiz.com/python-programming/online-compiler/

QUIZ:

- How we print “Hello World” in python?

  a)print(“Hello World”);

  b)print(“Hello World”)

  c) print(‘Hello World’)

  d)Both b & c

- Where you can run python program on our computer?

  a) VS Code

  b) Python IDLE

  c) Online Compiler

  d) All of the above

- What compiler does_____.

Do you Know?

```
a=10
print(a)
    b=20
print(b)
```

Will this execute?

```
ERROR!
Traceback (most recent call last):
  File "<main.py>", line 5
    b=20
IndentationError: unexpected indent
```

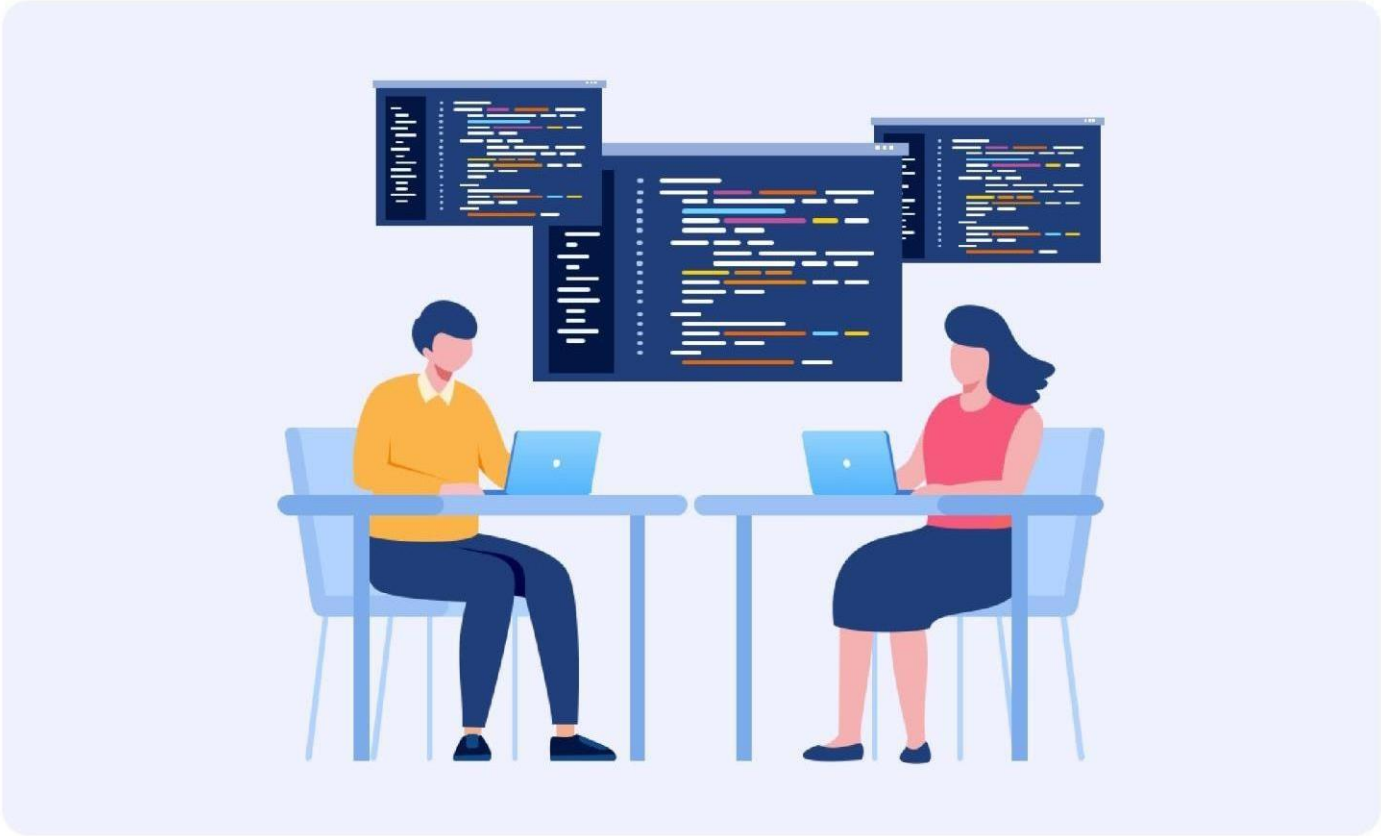
NO because in line 3 there is an unwanted space before b

EE.iNfo

PYTHON in 21 Days

-Crack any coding interview-

DAY 02: EE.INFO

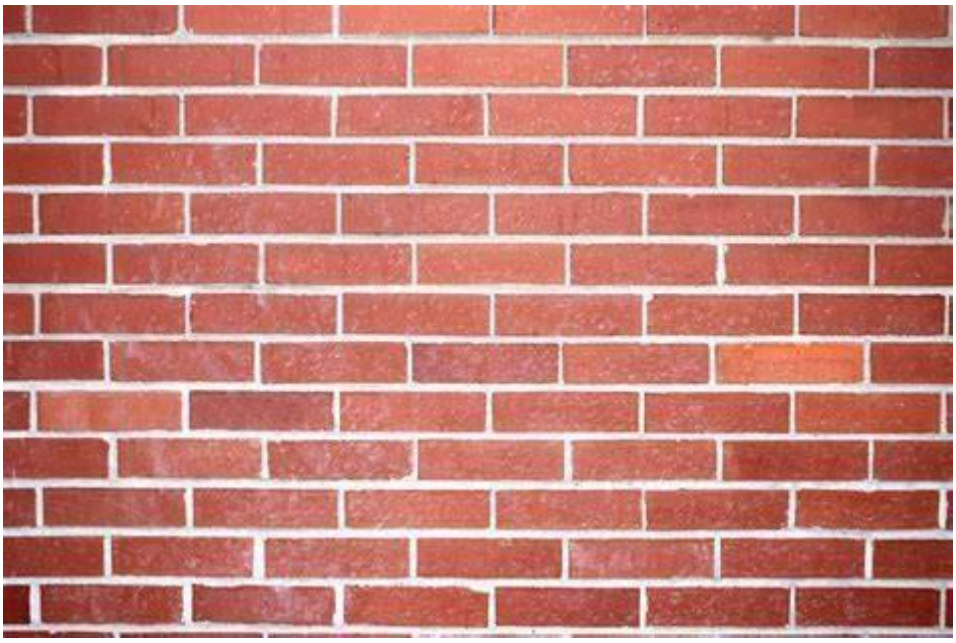


If you know fundamentals, then 80% work is done

Bricks of Python – Fundamentals:

- Brick-01,02,03,04

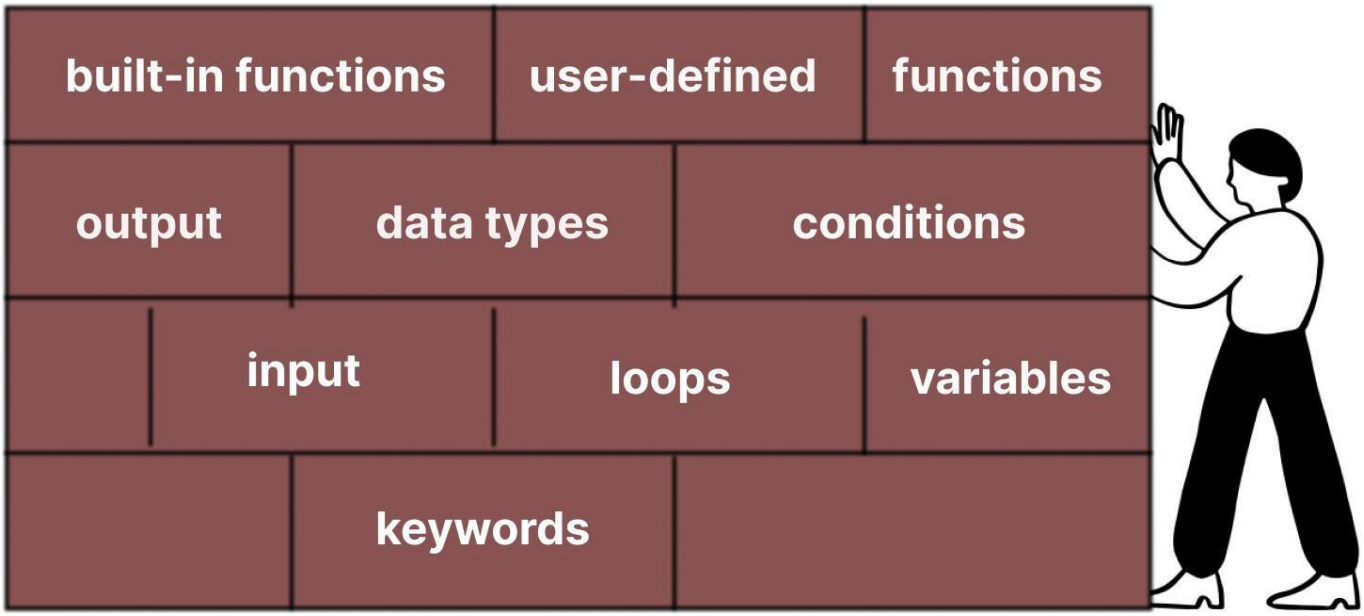
Python Bricks:



Just Imagine,

python ఒక wall అనుకోండి , wall ని build చేయడానికి basic unit

"brick" అలానే python కి bricks ఏంటో చూద్దాం ! ఈ Day 2,3,4 లో .



- Keywords
- Variables
- Data types
- Print,input,output
- Loops
- Conditions
- Built-in functions
- Functions

Brick-01:

2.1 Keywords:

Reserved words in a program, don't panic. ఇవి ఏంటో next chapter లో మీకే అర్థమవుతుంది. Read once.

If , else, elif, for, while, def, return, import, from, as, in, and, or, not, try, except, finally with class lamda.

Ex-1:

```
x=float(1)          # output: 1.0
y=float(29)         # output: 2.9
z=float("3")        # output: 3.0
w=float("8.6")      # output: 4.2
```

Ex-2:

```
x=str("s1")         # output: 's1'
y=str(2)            # output: '2'
z=str(3.0)          # output: '3.0'
```

Comments:

Programలో comments అనేవి execute అవ్వవు, మరి దాని use ఏంటంటే programని అర్థం చేస్కొడానికి use అవుతాయి.

Single line comments:

```
print(10+20)  #sum of two numbers
```

Output:

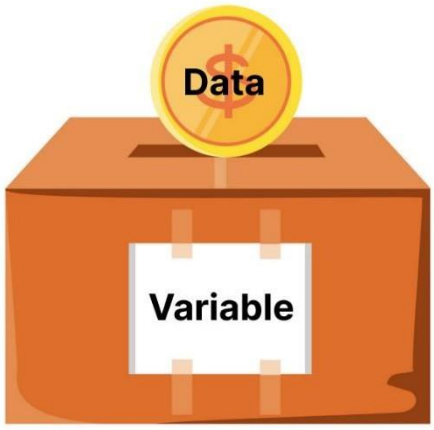
30

The above program will prints the summation of two numbers Own.So click here => [link]

Brick-02:

2.2 Variables:

Suppose ఒక Boxలో Coin వేసారు.....



- **Coin** అనేది **Data.**
- **Box** అనేది **Variable.** Suppose ఆ box కి

  “box1”అనే name ఇచ్చారు, అది

  variable name.

Rules for naming Variables:

1. Must start with a letter or an underscore.
   Ex: name, _name
2. Cannot start with a number.
   Ex: 23name, 7name (wrong)

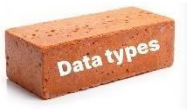
3.Can only contain letters(a-z,A-Z); numbers(0-9), underscores(_).
   Ex:_name12,name21,(name) (wrong),**@attitudeboy** (wrong)

4.Case sensitive.
   Ex: **age!=Age**

Examples:

1. name=”Aggipetti Macha”
2. age=31
3. is_student=false

Brick-03:



2.3 Data types:

Data Type	Example
Int	age=25
Float	height=5.9
str	name=”Mahesh babu”
bool	is_student=False(or)True
list	colors=[“red”,”blue”,”green”]
tuple	coordinates=(10,20)
dict	person={“name”:”john”,”age”:25}
set	studentsid={512,513,514}
nonetype	x=none

Type Casting:

Coding problem solve చేస్తున్నప్పుడు, ఒక datatypeని ఇంకొక datatypeకి మార్చాల్సివస్తుంది.

Ex-1:

```
x=int(1)
print(x)
```

Output:

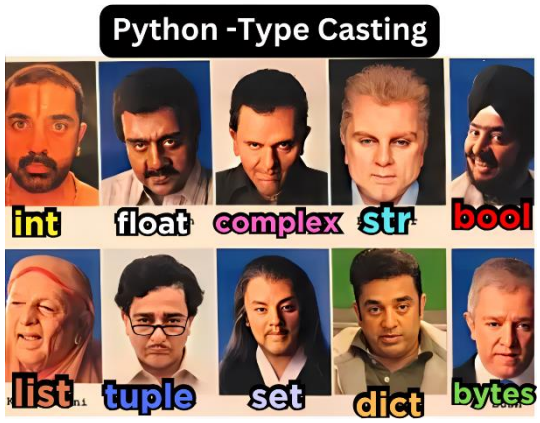
30

Ex-2:

```
y=int(2.8)
print(y)
```

Output:

2



Ex-3:

```
z=int("3")
print(z)
```

Output:

3

Brick-04:

2.4 Input & Outputs in Python:

INPUT FORMAT

1. Pythonలో ఒక string(sentence 1 word)ని ఎలా input తీస్కోవాలి?

Program-1:

```
s1 = input("enter any sentence:")
print("s1 is:", s1)
```

```
Output
enter any sentence:okasari mavayya anamma
s1 is: okasari mavayya anamma
```

2. Pythonలో ఒక integerని ఎలా input తీస్కోవాలి?

Program-2:

```
n = int(input("enter any number:"))
print("number is:",n)
```

```
Output
enter any number:23
number is: 23
```

Similarly,

```
n = float(input("enter any float number:"))
print(n)
```

```
Output
enter any float number:8.234
8.234
```

```
Output
enter any float number:56
56.0
```

మీకు నిజంగా రావాలంటే practice చెయ్యాలి click the below link to practice

[link]

OUTPUT FORMAT

1. Pythonలో ఒక stringని ఎలా print చెయ్యాలి?

Program-1:

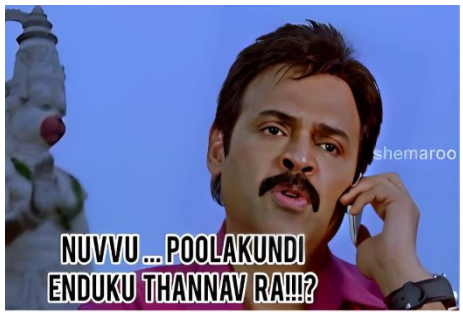
```
s1="Nimmakaya rasam"
print(s1)
  (or)
print("Balayya said"+s1)
  (or)
print("Balayya said"+s1+"on phone")
  (or)
 print(F "Balayya said {s1}")
```

```
Output
Nimmakaya rasam
Balayya saidNimmakaya rasam
Balayya saidNimmakaya rasamon phone
Balayya said {'Nimmakaya rasam'}
```

Program-2:

```
s1="Kundi"
s2="Enduku Tannav Ra"
print(s1+" "+s2)
```

```
Output
Kundi Enduku Tannav Ra
```



Program-3:

```
name="Raja"
age=23
print(f"My name is {name} and iam {age} years old")
```

Output

```
My name is Raja and iam 23 years old
```

మీ styleలో వీటిని try చెయ్యండి [link]

1. Pythonలో numbersని ఎలా print చెయ్యాలి?

Program-1:

```
a,b,c=10,20,30
print("numbers are:" a,b,c)
```

Output-1: numbers are:10,20,30

(or)

```
print(a,b,c)
```

Output-1: 10,20,30

(or)

```
print(f"first number: {a} , second number:{b} , third number:{c}")
```

Output-1: first number:10 , second number:20 ,third number:30

Program-2:

```
x=123.456789
print(f"{x:.2f}")
```

```
x=123.456789
print(f"{x:.3f}")
```

Output-1:123.46

Output:123.46

Output:123. 457

Numbersని pythonలో print చెయ్యండి [link]

Day-02 Summary:

What we have learnt on day-02:

we have seen about fundamental bricks of python. They are:

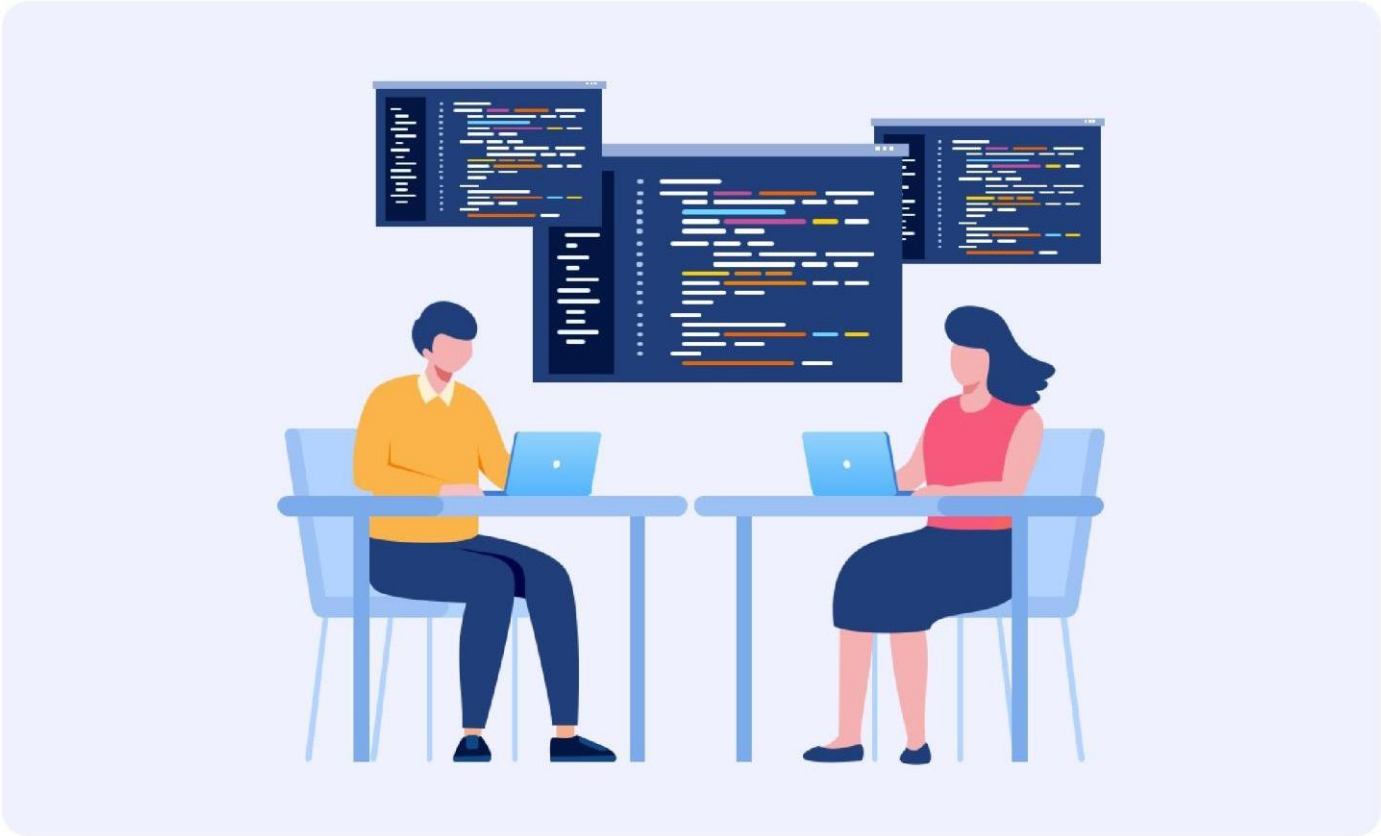
1. What are keywords in python
2. What are variables, naming variables
3. What are data types, type conversions, comments
4. How to take inputs, display outputs in different formats

EE.iNfo

PYTHON in 21 Days

-Crack any coding interview-

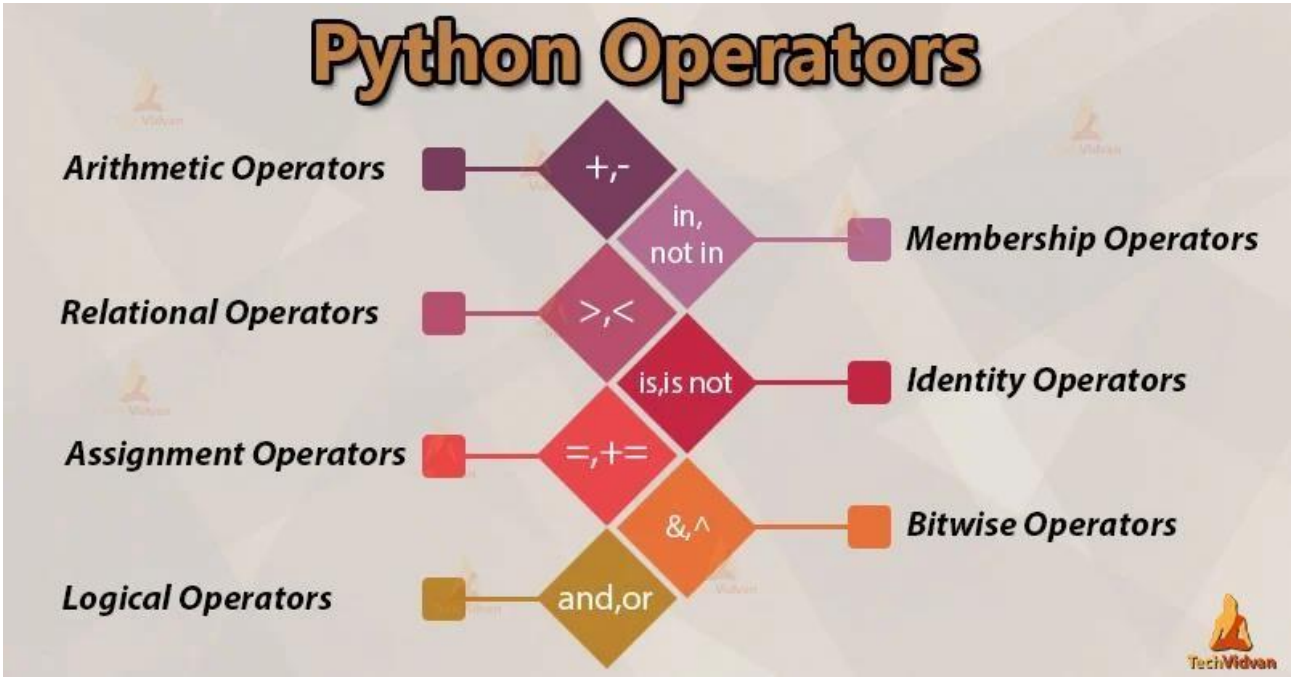
DAY 03: EE.INFO



If you know fundamentals, then 80% work is done

Bricks of Python – Fundamentals:

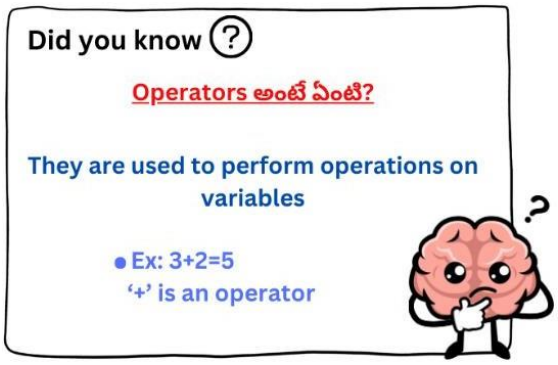
Brick-05: Operators



Let's see the operators used in python programs one by one:

1. Arithmetic Operators: Mathematical operations

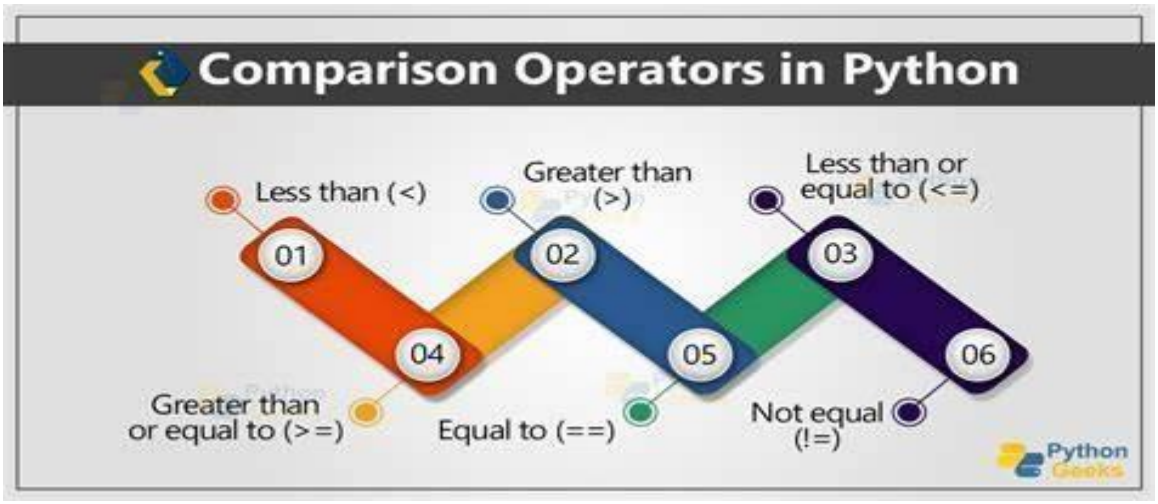
Operator	Description	Example
+	Add two operands	x+y = 26
-	Subtract one operand from another	x-y = 14
*	Multiply one operand with another	x*y = 120
/	Divide one operand with another	x/y = 3.333
%	Remainder of division between two operands	x%y = 2
**	x to the power of y	x**y = 64000000
//	Floor division. Removes decimal part after division.	x//y = 3



2. Assignment Operators:

Operator	Example	Equivalent Expression (m=15)	Result
=	y = a+b	y = 10 + 20	30
+=	m +=10	m = m+10	25
-=	m -=10	m = m-10	5
*=	m *=10	m = m*10	150
/=	m /=10	m = m/10	1.5
%=	m %=10	m = m%10	5
**=	m**=2	m = m**2 or $m = m^2$	225
//=	m//=10	m = m//10	1

3. Comparison Operators: returns either “True”/ “False”



1. Equal operator (“==”):

Ex:

```
x=10
y=20
print(x==y)
```

```
Output
False
```

```
x=10
y=10
print(x==y)
```

```
Output
True
```

2. Not Equal operator (“!=”):

Ex:

```
x=10
y=20
print(x!=y)
```

```
Output
True
```

```
x=10
y=10
print(x!=y)
```

```
Output
False
```

3. Greater than (“>”):

Ex:

```
x=100
y=30
print(x>y)
```

```
Output
True
```

4. Greater than (or) Equal to (“>=”):

Ex:

```
x=40
y=10
print(x>=y)
```

```
Output
True
```

5. Less than (“<”):

Ex:

```
x=30
y=250
print(x<y)
```

```
Output
True
```

6. Less than (or) Equal to(“<=”):

write a program for this

Execution చాలా easy పై programని impliment చేయండి [link]

4. Logical Operators: returns either “True”/ “False”

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then then condition becomes true.	(a and b) is true.
or	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	(a or b) is true.
not	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	not(a and b) is false.

1.“and”:

Ex:

```
a,b,c=40,20,10
if(a>b) and (a>c):
    print("a is the greater number")
```

```
Output
a is the greater number
```

2.“or”:

Ex:

```
if(5>3) or (5>14):
    print("5 is the middle number")
```

```
Output
5 is the middle number
```

3.“not”:

Ex:

```
print(not(2<5) and (2<10))
```

```
Output
False
```

5. Membership Operators:

Operator	Meaning	Example
In	True if value/variable found in the sequence	5 in x
Not in	True if value/variable is not found in sequence	5 not in x

1. **"in"**: ఒక example అర్థం చేస్కొండి.

Ex:

```
fruits=["apple","banana"] #list
print("banana" in fruits)
```

Output

```
True
```

2. **"not in"**: objectలో specific value లేకపోతే true ని return ఈ python operator.

Ex:

```
fruits=["apple","banana"]
print("pine apple" not in fruits)
```

Output

```
True
```

చదివితే రావడానికి ఇది **social** కాదు **python** ఇక్కడ implement చేయండి [link]

Day-03 Summary:

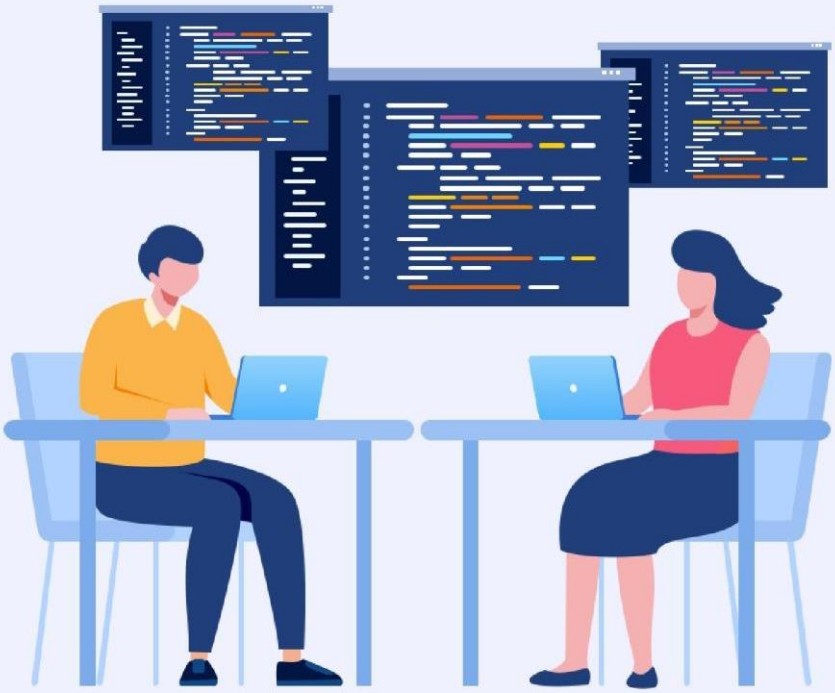
We explored and implemented 5 operators in python.

EE.iNfo

PYTHON in 21 Days

-Crack any coding interview-

DAY 04: EE.INFO



If you know fundamentals, then 80% work is done

Bricks of Python – Fundamentals:

Brick-06: Conditional Statements &
Brick-07: Loops

Conditional Statements

Conditional statements are used for decision making.
Simpleగా అర్థం చేస్కుందాం!

Suppose, ramesh అనే వ్యక్తి ఒక junctionలో ఉన్నాడు. Leftకి వెళ్తే

Hyderabadకి, rightకి వెళ్తే Vijayawada వస్తుంది.

Case-1: ఇప్పుడు Ramesh Hyderabad వెళ్ళాలి so,

దీనినే Decision Making అంటాం! ఈ conceptని pythonలో Conditional Statementsతో implement చేస్తాం!

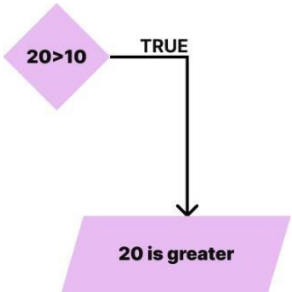
Statement-01: "IF"

Syntax: `if(condition):`
`statement(s)`

Ex:

```
if 20>10:
    print("20 is greater")
```

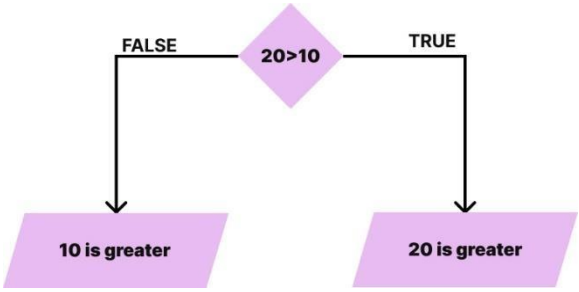
```
Output
20 is greater
```



Statement-02: "IF-ELSE"

Syntax:

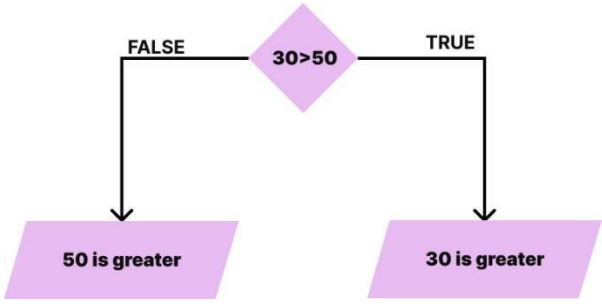
```
if(condition):
    statement(s)
else:
    statement(s)
```



Ex:

```
if 20>10:
    print("20 is greater")
else:
    print("10 is greater")
```

```
Output
20 is greater
```



Ex:

```
a=30
b=50
if a>b:
    print("30 is greater")
else:
    print("50 is greater")
```

```
Output
50 is greater
```

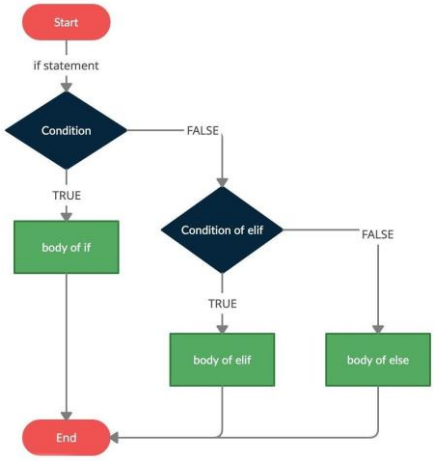
Practice చేస్తేనే వస్తుంది. so click here ➔ [link]

Task: write a program to check a give number is even or not (take a number as input)

Statement-03: "IF-ELIF"

Syntax:

```
if(condition):
    statement(s)
elif(condition):
    statement(s)
else:
    statement(s)
```



Ex:

```
a=10
b=30
c=20
if a>b and a>c:
    print("a is larger")
elif b>c:
    print("b is larger")
else:
    print("c is larger")
```

```
Output
b is larger
```

Ex:

```
a=250
b=30
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")
```

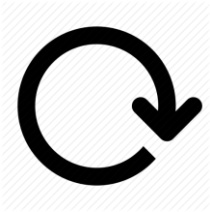
```
Output
a is greater than b
```

Python ,maths రెండు practice చేస్తేనే వస్తుంది ➔ [link]

Loops

ఎప్పుడైనా,**Fundamentals** నుంచి వెళ్ళాలి - **Elon musk**

So, Loop అంటే ఏంటి?



- Doing same thing(one task) for multiplied times.

Ex: Suppose, rameshకి pythonలో Loop అనే concept తెలియదు.

So నేను అతనికి 1 to 100 numbersని pythonలో print చేయమని చెప్పాను.

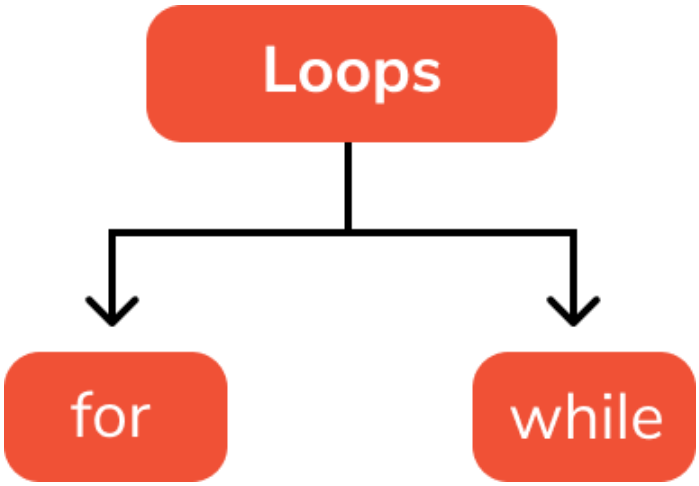
అప్పుడు code ఇలా ఉంటుంది:

```
print(1)
print(2)
print(3)
print(4)
print(5)
.
.
.
print(100)
```

ఇప్పుడు 1-10,000 ని print చేయమని చెప్పాను. ఈ

caseలో ప్రతీసారీ print రాయకుండా Loopని వాడచ్చు.

Pythonలో Two Loop Statements ఉన్నాయి:



Loop-01: “FOR-LOOP”

Syntax: for variable in range(start, end+1, increment/decrement):
#statement(s)

Ex-1:

```
for i in range(0,11):
    print(i)
```

Output

```
0
1
2
3
4
5
6
7
8
9
10
```

ఇప్పుడు rameshని single line 1-10 numbersని print చేయమని చెప్పాను!

అప్పుడు

Code:

```
for i in range(0,11):
    print(i,end=' ')
```

```
Output
0 1 2 3 4 5 6 7 8 9 10
```

ఇప్పుడు మీరూ చెయ్యండి [link](మీరు 1-100 ని print చెయ్యండి!).

Ex-2:

```
for i in range(0,11,2):
    print(i)
```

```
Output
0
2
4
6
8
10
```

Ex-3:

```
for i in range(10,0,-1):
    print(i)
```

```
Output
10
9
8
7
6
5
4
3
2
1
```

Loop-02: “WHILE-LOOP”

Interviewsలో చేసే codingలో for loop కంటే **while loop**ని use చేస్తాం!

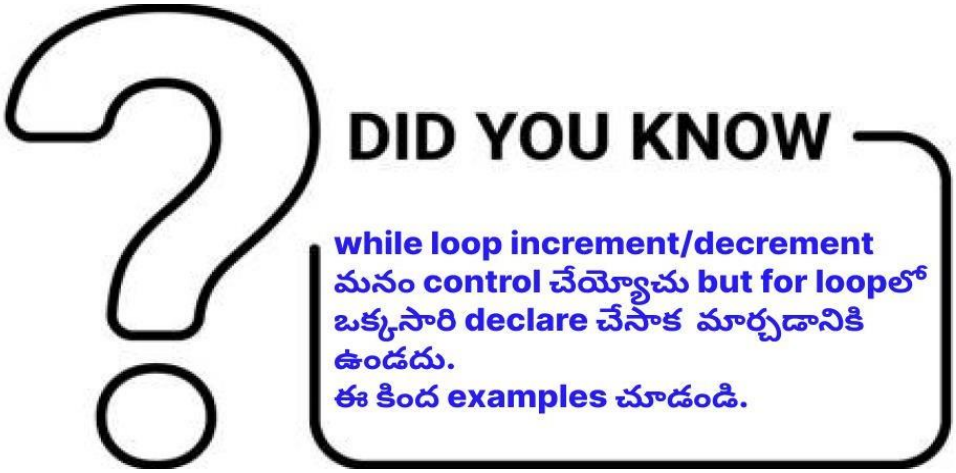
Syntax:

```
variable-initialization
while(condition):
    #statement(s)
    Update variable(increment/decrement)
```

Ex:

```
i=0
while i<11:
    print(i)
    i=i+1
```

```
Output
0
1
2
3
4
5
6
7
8
9
10
```



Ex-1:

```
for i in range(0,10,):
    print(i)
    i=i+5
```

```
Output
0
1
2
3
4
5
6
7
8
9
```

Ex-1:

```
i=0
while(i<11):
    print(i)
    if(i<5):
        i=i+1
    else:
        i=i+2
```

```
Output
0
1
2
3
4
5
7
9
```

కొన్ని example programs చూద్దాం!

Programme-01: print 1-20 numbers using while loop.

Code:

```
i=1
while i<=20:
    print(i,end=' ')
    i=i+1
```

```
Output
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

Programme-02: print 1-20 numbers in reverse order.

Code:

```
i=20
while i>=1:      #or i>0
    print(i,end=' ')
    i=i-1
```

```
Output
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
```

Programme-03: print even numbers (0,2,4,..) upto 10.

Code:

```
i=0
while i<=10:
    print(i,end=' ')
    i=i+2
```

```
Output
0 2 4 6 8 10
```

Programme-04: print numbers which are divisible by 3 from 1-20.

Code:

```
i = 1
while i <= 20:
    if i % 3 == 0:
        print(i,end=' ')
    i += 1
```

```
Output
3 6 9 12 15 18
```

Task on loops: task ఏంటి అంటే పై programmeని for loopలో implement

చెయ్యండి=>[link]

Day-04 Summary:

We learnt,

1. What are condition statements?
2. What are loops in python?

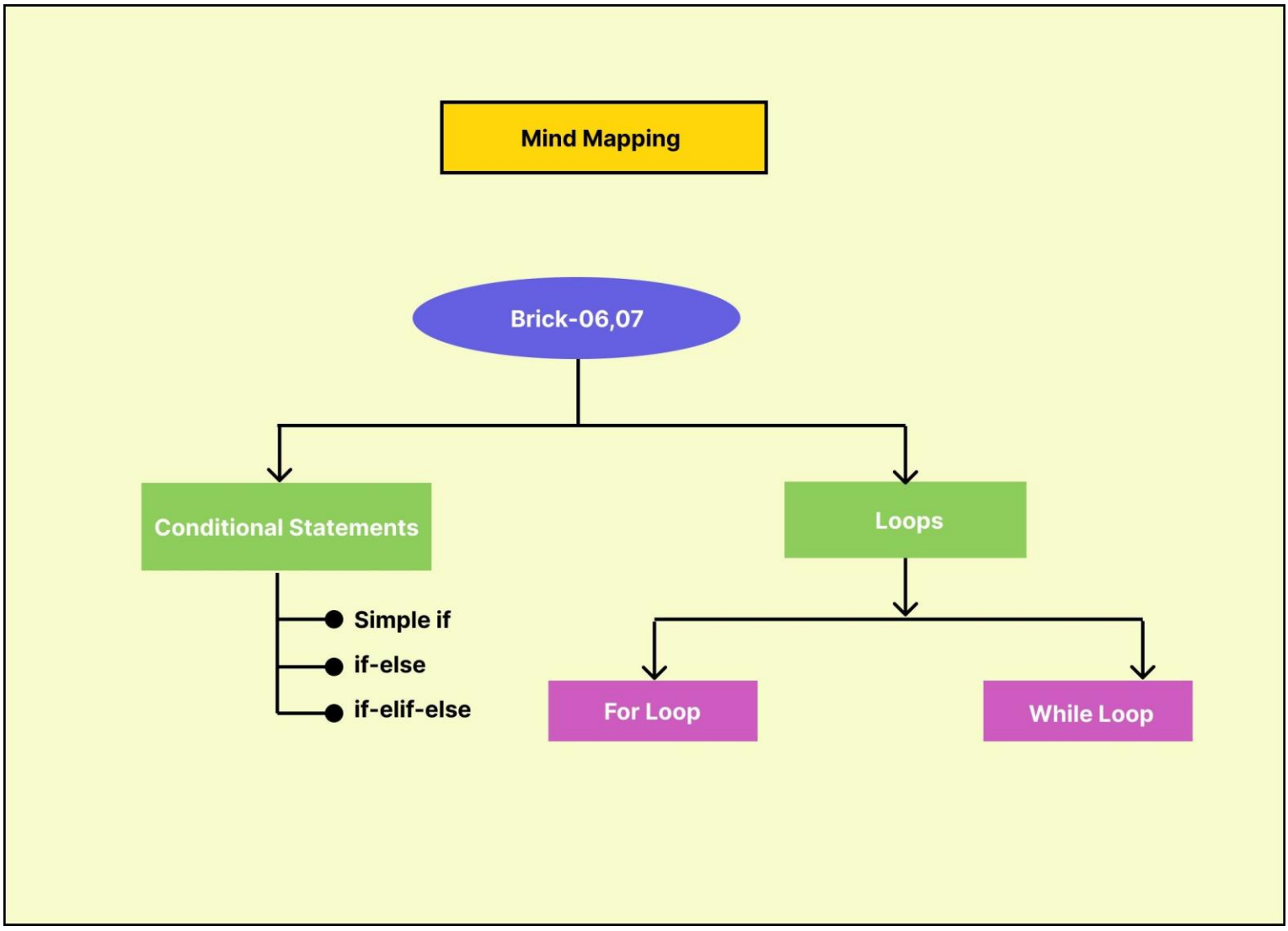
Cheat Sheet:

1. **if condition:**
   #statement(s)

2. **if condition:**
   #statement(s)
   else:
   #statement(s)

3. if condition:
#statement(s)
elif condition:
#statement(s)
else:
#statement(s)

4.write syntax of for &while loop


Mind Mapping
Brick-06,07
Conditional Statements
Simple if
if-else
if-elif-else
Loops
For Loop
While Loop

EE.iNfo

PYTHON in 21 Days

-Crack any coding interview-

Day-05 Coding Practice Sheet

Note: Use Python IDE or VS Code, if not possible go with online compiler **[**link**]** to implement below code answers and don't forget to save the code in your device

1.Conditional statements

- **Number Check:**
  Write a Python program to check if a given number is positive, negative, or zero.
  Note: "Take a number as input from user in the output"
- **Leap Year:**
  Write a program to check if a given year is a leap year.
- **Largest Number:**
  Take three numbers as input and find the largest among them using if-elif-else.
- **Discount Calculator:**
  If a customer's purchase amount exceeds ₹5000, apply a 10% discount. Otherwise, no discount.
- **HackerRank:**
  Given an integer, perform the following conditional actions:

[LINK]

2. Loops:

6. **Print Multiples:**
   Print all multiples of 5 from 1 to 50 using a for loop.
7. **Factorial Calculation:**
   Write a program to calculate the factorial of a number using a while loop.
8.. **Sum of Digits:**
   Take a number as input and calculate the sum of its digits using a while loop.
9. **Reverse a Number:**
   Reverse a given number using a loop (e.g., 123 → 321).
10. **Prime Number Check:**
    Write a program to check if a number is prime or not using a loop.

11. **HackerRank:** Given an integer, print its first 10 multiples **[** link **]**
12. **HackerRank:** Print the square of each number on a separate line from 0 to n-1. **[**link**]**

13. **Draw the below pattern in python using loops:**

Pattern1:

```
*
* *
* * *
* * * *
```

Pattern2:

```
1
1 2
1 2 3
1 2 3 4
```

EE.iNfo

PYTHON in 21 Days

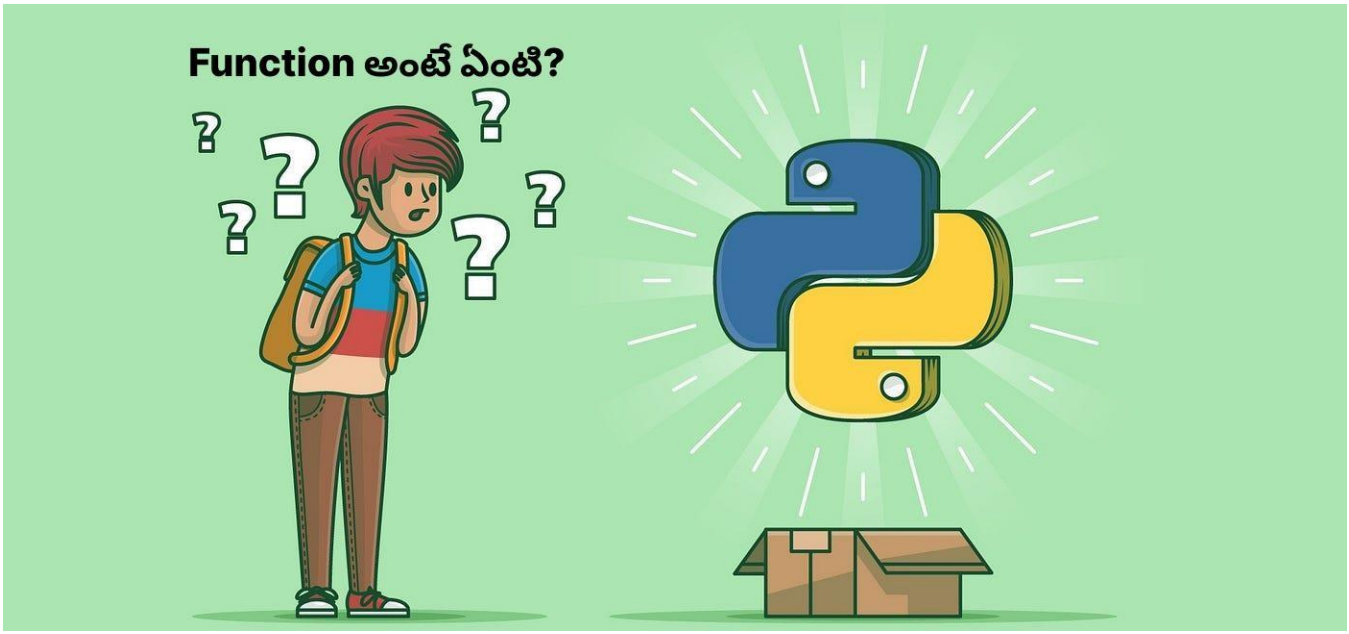
-Crack any coding interview-

DAY 06 : EE.INFO

If you know fundamentals, then 80% work is done

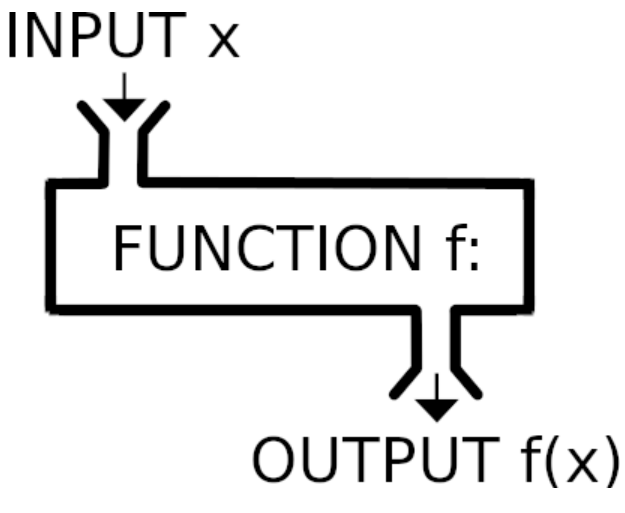
Bricks of Python – Fundamentals:

Brick-08: Functions in Python



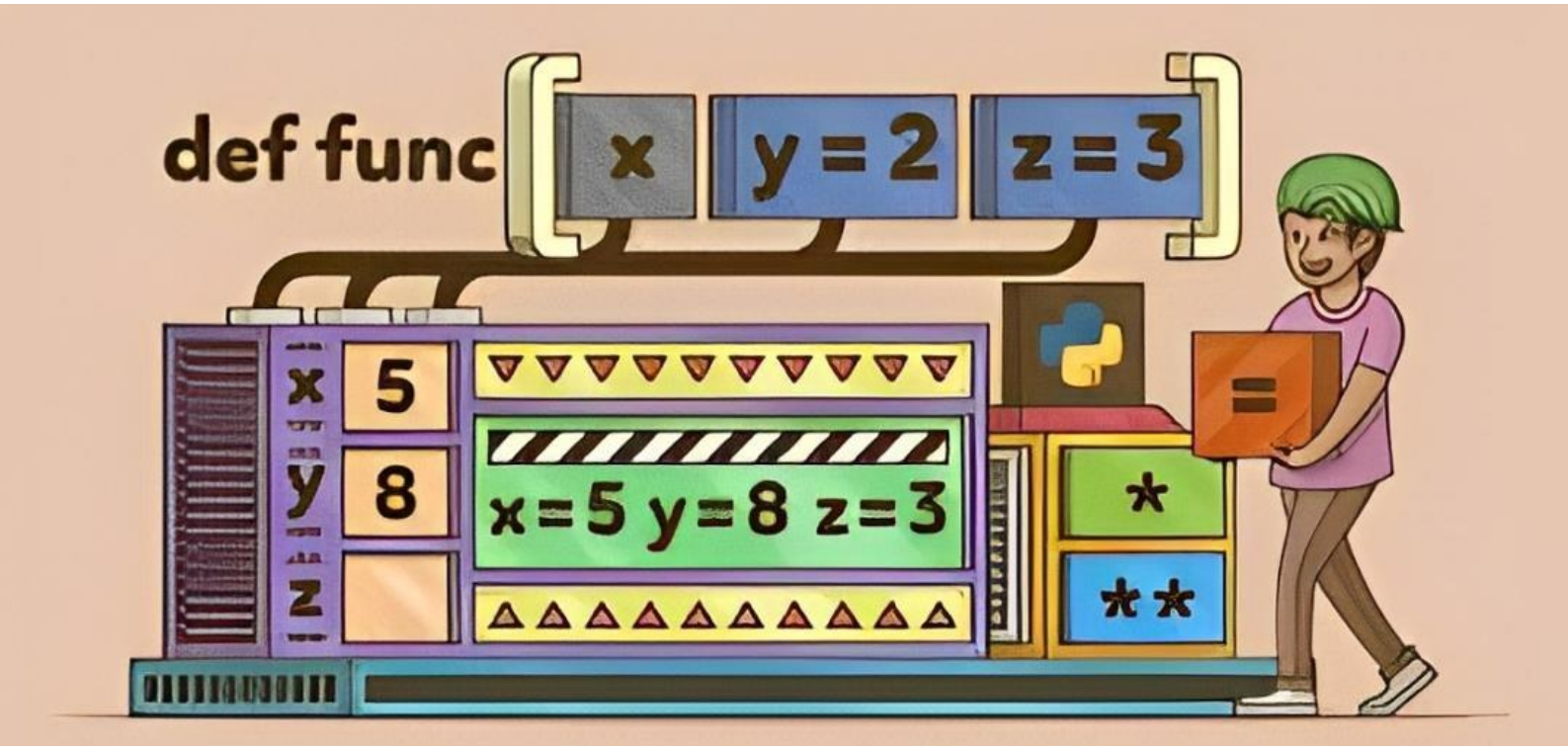
Function అంటే ఏంటో we'll see later. Pythonలో 2 types of Functions ఉంటాయి అవి ఏంటంటే:

1. **User-defined Function**
2. **Built-in Function**



6.1 USER DEFINED FUNCTIONS:

Very easy, as usually let us start from fundamentals.



అసలు Functions Programmeలో ఎందుకు use అవుతాయి?

Definition: A function is a block of code that performs a specific task. A function helps to make your code more organised and reduce repetition of logic (code reusability).

ఒక Simple exampleతో అర్థం చేస్కుందాం.

Finding 3 rectangles area in python programme:

Ramesh దానికి code ఇలా రాశాడు:

```
l1=10
b1=20
area=l1*b1
print("Area of rectangle 1:",area)

l2=12
b2=2
area=l2*b2
print("Area of rectangle 2:",area)

l3=5
b3=4
area=l3*b3
print("Area of rectangle 3:",area)
```

```
Output
Area of rectangle 1: 200
```

```
Output
Area of rectangle 2: 24
```

```
Output
Area of rectangle 3: 20
```

So, ఇక్కడ Ramesh రాసింది correctవి, కానీ programmeలో ఈ approach follow అవ్వకూడదు. ఎందుకంటే,It wastes time, space and efforts. So, అప్పుడే FUNCTION అనే concept తీస్కొచ్చారు.

See same example in function:

Finding 3 rectangles area using functions:

```
def area(l,b):
    return l*b
print(area(10,20))
print(area(12,2))
print(area(5,4))
```

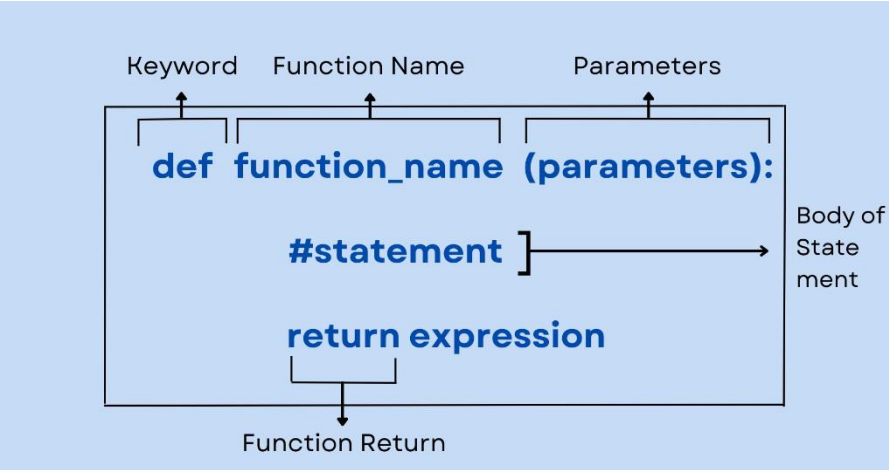
Output

```
200
24
20
```

How Functions Works?

Step-1: Declare Function (function declaration)

Syntax:



Note:
Function may (or) may not have return statements.

But in 90% use cases we have return statements.

Ex:

```
def sum(a,b):
    c=a+b
    print("sum is:",c)
sum(20,2)
```

Output

```
sum is: 22
```

Step-2: Call that function (function call)

```
1. def areaCalc(l,b):
2.     area=l*b
3.     return area
4. #Now Step-2: call the function "areaCalc" to calculate area
5. #function call
6. c=areaCalc(2,10)
7. print("Area of rectangle:",c)
```

Output

```
Area of rectangle: 20
```

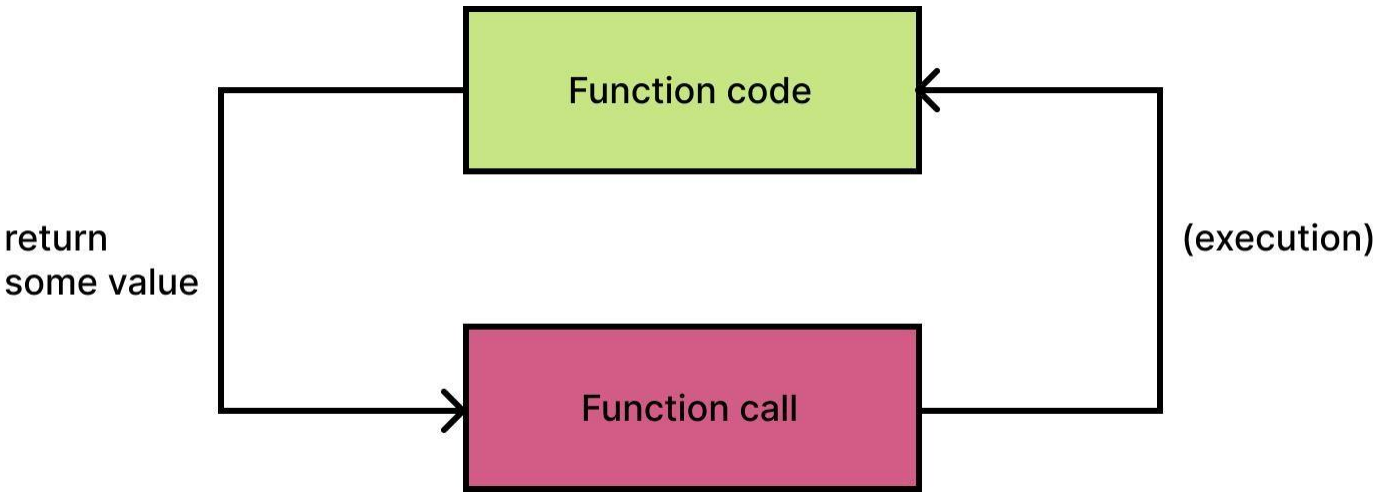
Note: when line 6 is executed on compiler, it goes to function declaration to execute whole function code with the input 2,10 (l,b)

- Calculates area=2*10=20 [line 2]
- Returns 20 [line3]
- Now this 20 is stored in c [line 6]
- In [line 7] we use this c to print rectangle area of (2,10) [line 7]

Ex:

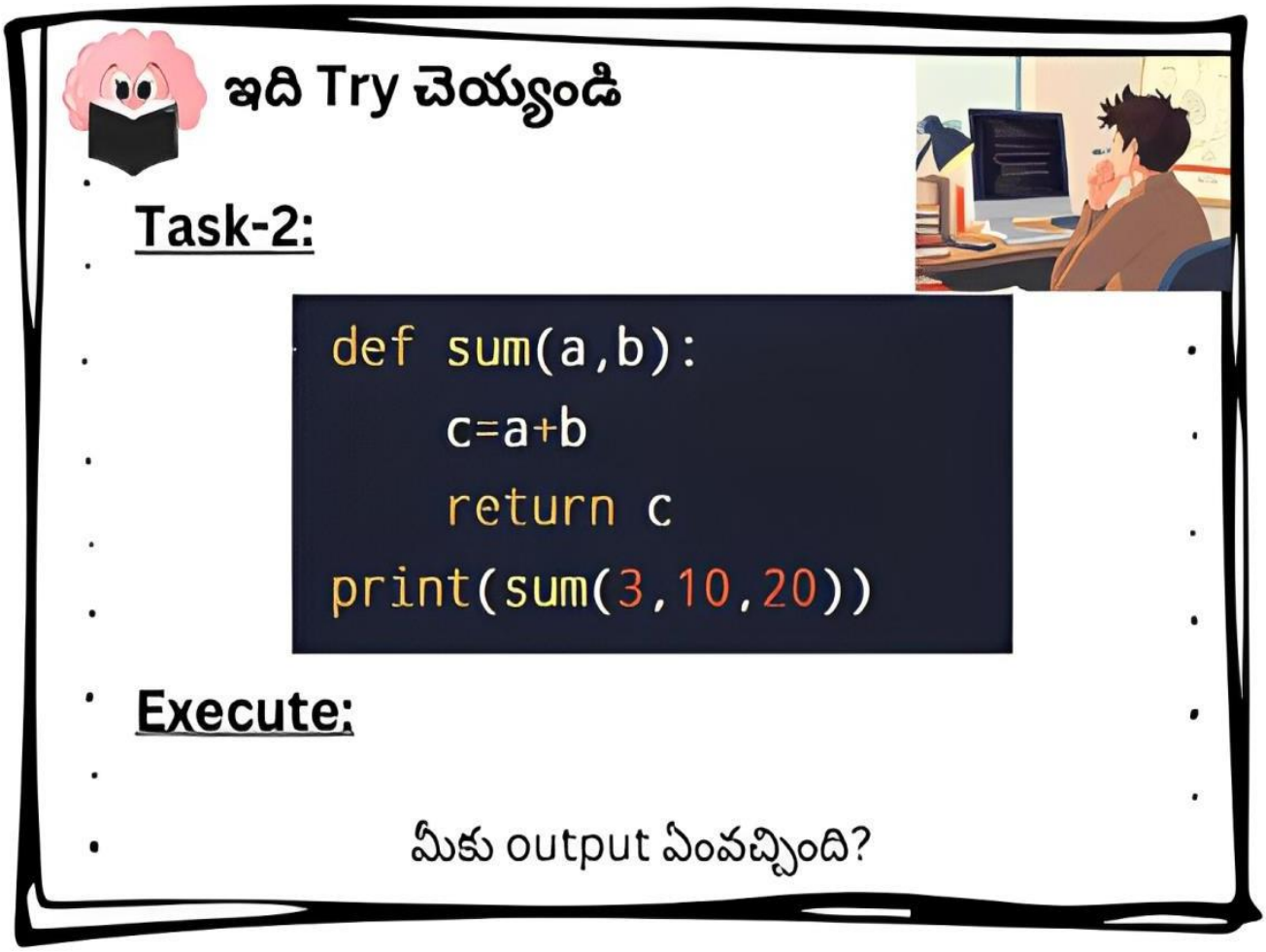
one woman asks sugar from other home by calling the other woman and then she returns it(sugar).

Function Mechanism



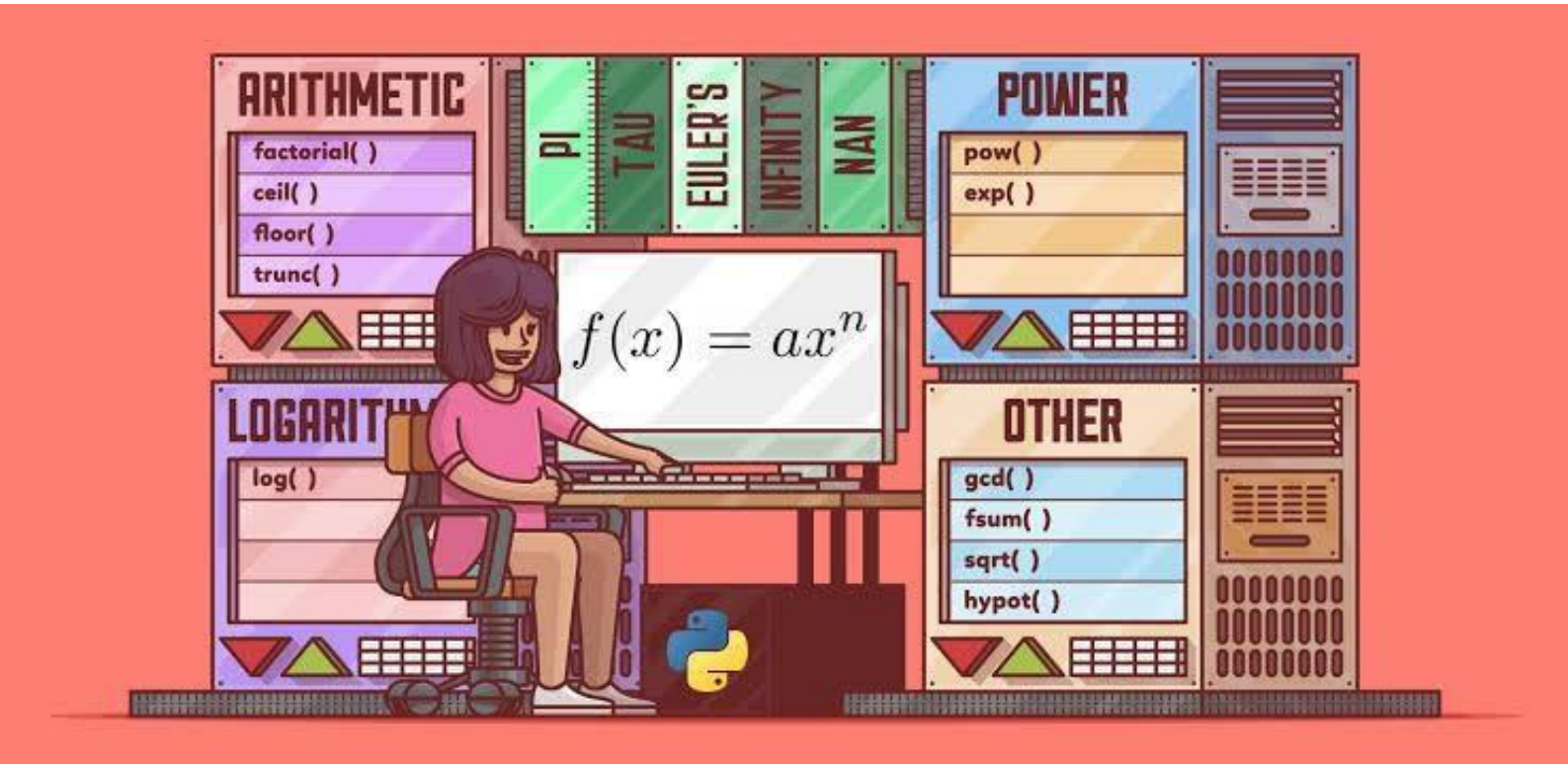
Task 1: Now write a function that returns area of circle here ➔ [link]

Now, Ramesh has understood the concept of functions in python. Do you?



6.2 PYTHON BUILT-IN FUNCTIONS:

ఒక చిన్న Example తో అర్థం చేస్కుందాం!



Rameshకి నేను ఒక string (word / sentence) python use చేసి దాని length find out చేయమని చెప్పాను. అప్పుడు

Ramesh ఇలా code రాశాడు:

Code:

```
s1=input("Enter some string:")
count=0
for char in s1:
    count=count+1    #you try print(char) also
print("String length:",count)
```

```
Output
Enter some string:Hrudaya Kaleyam
String length: 15
```

So, Ramesh ఇంత కష్టపడకూడదని, pythonలో కొన్ని shortcuts ఉన్నాయ్ అవే

“Built-in functions”.

Code:

```
s1=input("Enter some string:")
s1length=len(s1)  #built-in function to find sting length
print("String length:",s1length)
```

Output: Execute చేసి చూడండి. [link]

So, Built-in Functions are nothing but shortcuts / built-in code to make operations in python.

Some most used Built-in Functions in python:

1.Mathematical:

Ex:

```
a=-10
b=3.768
c=[1,2,3,4,5]  #it is a list we'll see it in Day-08
print(abs(a))  #prints absolute value
print(round(b)) #rounds to nearest integer
print(round(b,2))
print(pow(2,3)) #Exponentiation 2*2*2
print(max(c)) #maximum value in list"c"
print(min(c)) #minimum value in list"c"
print(sum(c)) #sum of all in list"c"
```

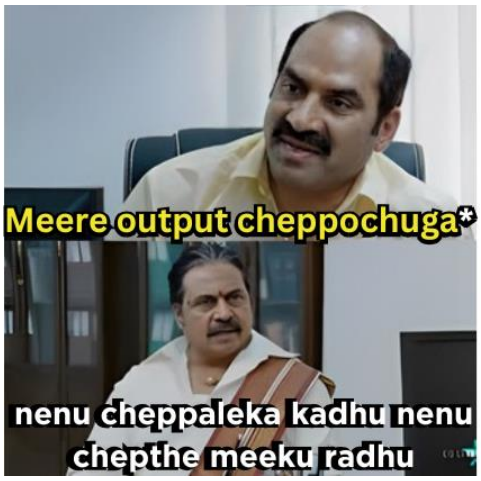
```
Output
10
4
3.77
8
5
1
15
```

Ramesh 1-10 numbersలో random numberని print చెయ్యడానికి code ఇలా రాశాడు built-in functions use చేసి:

Code:

```
import random
n=random.randint(1,10) #including 1,100
print(n)
```

Output: ఏంటి?

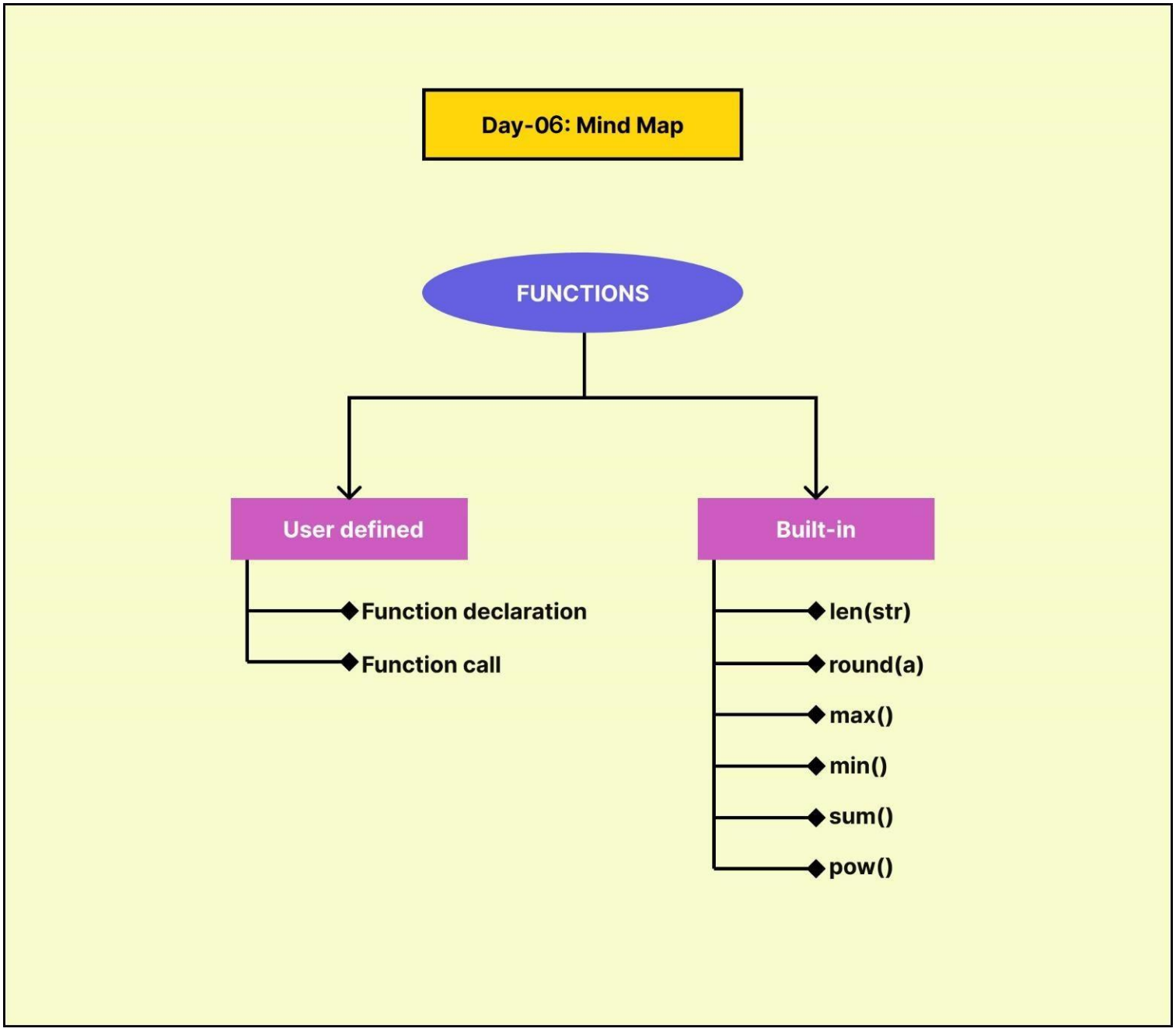


[link]

Day-06 Summary:

What we have learnt on Day-06?

1. What are functions? How to declare and call functions?
2. What are built-in functions ? How they differ from user defined functions
3. Some important built-in functions.



EE.iNfo

PYTHON in 21 Days

-Crack any coding interview-

Day-07 Coding Practice Sheet

User defined functions: (Hackerrank)

1.Print the list of integers from through as a string, without spaces: [LINK]

write your code here

2.Leap year or not: [LINK]

write your code here

User-Defined Functions:

⇒ **Circle Area Function:**
Define a function that takes a radius as input and returns the area of a circle.

⇒ **Simple Calculator:** Implement a function that performs addition, subtraction, multiplication, and division based on user input.

⇒ **Prime Number Check:**
Write a function that takes a number and returns True if it's prime, otherwise False.

⇒ **Reverse a String:**
Create a function that takes a string as input and returns its reversed version.

Built-in functions:

⇒ **Find Maximum & Minimum:**
Take a list of numbers [12, 34, 40, 25, 30, 10] and use built-in functions max() and min() to find the highest and lowest numbers.

⇒ **Random Number Generator:**
Use the random module to generate and print a random number between 1 and 100.

⇒ **Length of a String:**
Take a sentence as input and use len() to count the number of characters.

⇒ **Sort a List:**

Take a list of numbers and use sorted() to arrange them in ascending order like [12, 34, 40, 25, 30, 10].

⇒ **Power Calculation:**

Use pow() to calculate a^b where a and b are user inputs.

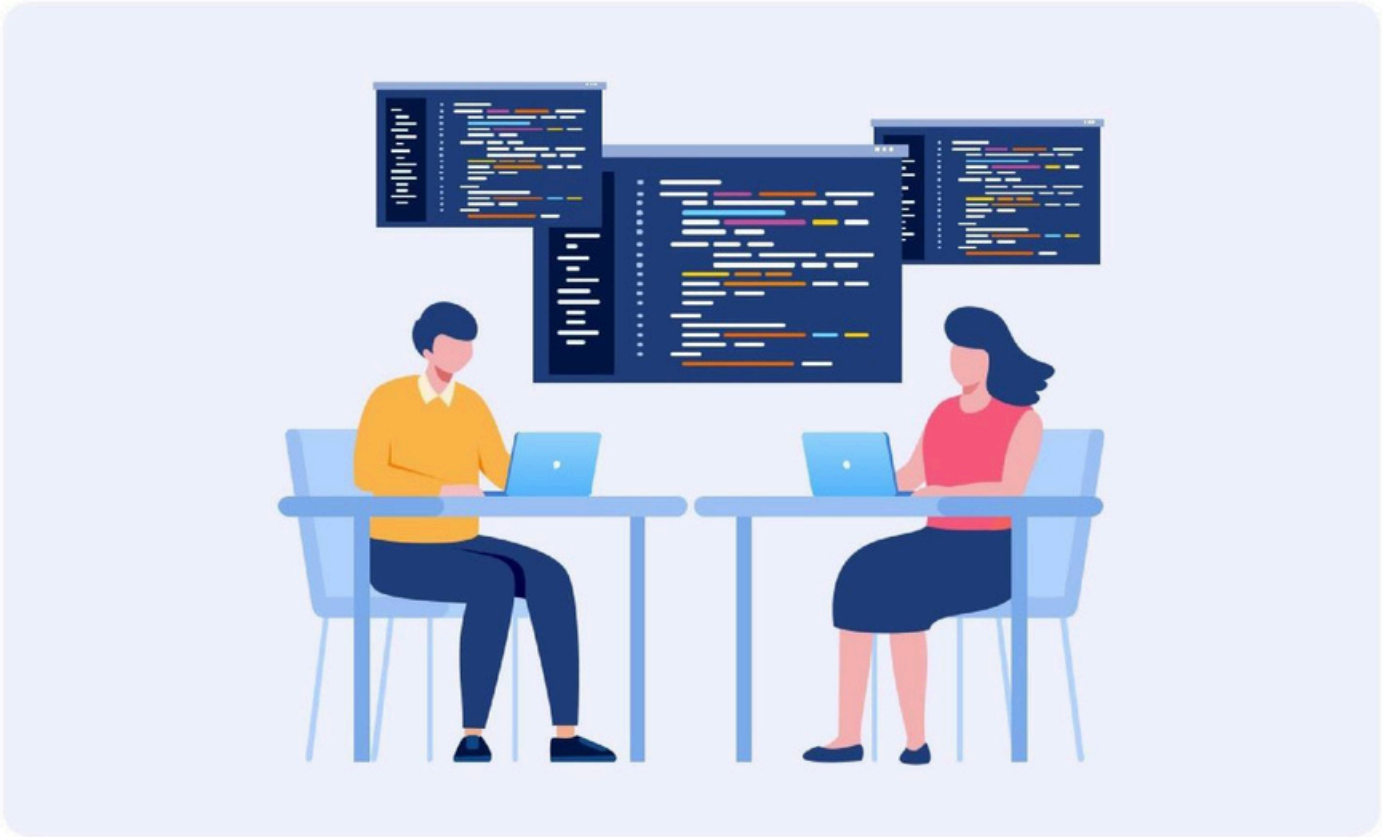
write your code here [LINK]

EE.iNfo

PYTHON in 21 Days

-Crack any coding interview-

DAY 08: EE.INFO



“Every Software/application works on Data”

“CLASSES and OBJECTS”

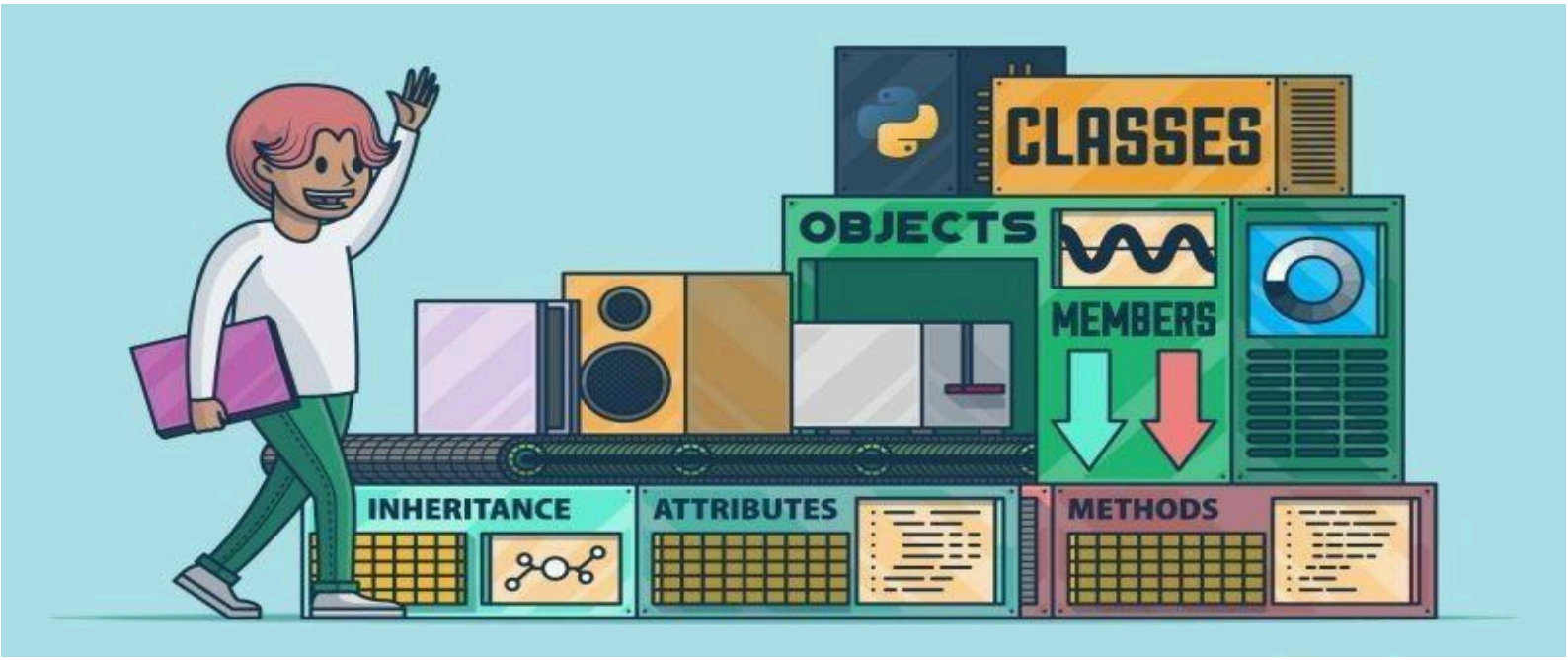
“in real time applications, the code is written in Classes and Objects even in coding rounds also”

8.1 CLASSES

Lecturer: Ramesh! Classes అంటే ఏంటీ?

Ramesh: తరగతి గది!

ఇప్పుడు Rameshకి అర్థమయ్యేలా Classes అంటే ఏంటో చూద్దం !



Class Definition: Collection of Variables, functions(methods), as one unit is called a Class.

(or)

In otherwords Class is a "Blueprint" for creating an object.

Object Definition: Instance of the class.

Case-1:

Ex:

```python
class Student:
  def __init__(self,name,age,grade):    #constructor
      self.name=name
      self.age=age
      self.grade=grade
  def display_info(self):
    print(f"Name: {self.name}, Age:{self.age}, Grade:{self.grade}")
#creating an object
Student1=Student("Alice",15,"10th")    #invokes constructor
Student2=Student("Bob",14,"9th")       #invokes constructor
Student1.display_info()                #calls display_info() functions
Student2.display_info()                #calls display_info() functions
```

Output:

```
Name: Alice, Age:15, Grade:10th
Name: Bob, Age:14, Grade:9th
```

Task:

Include your details as "student3" object in the above class example programme and check the output.

Case-2: Codingలో ఈ formatలోనే code రాస్తాం So highly important.

Ex:

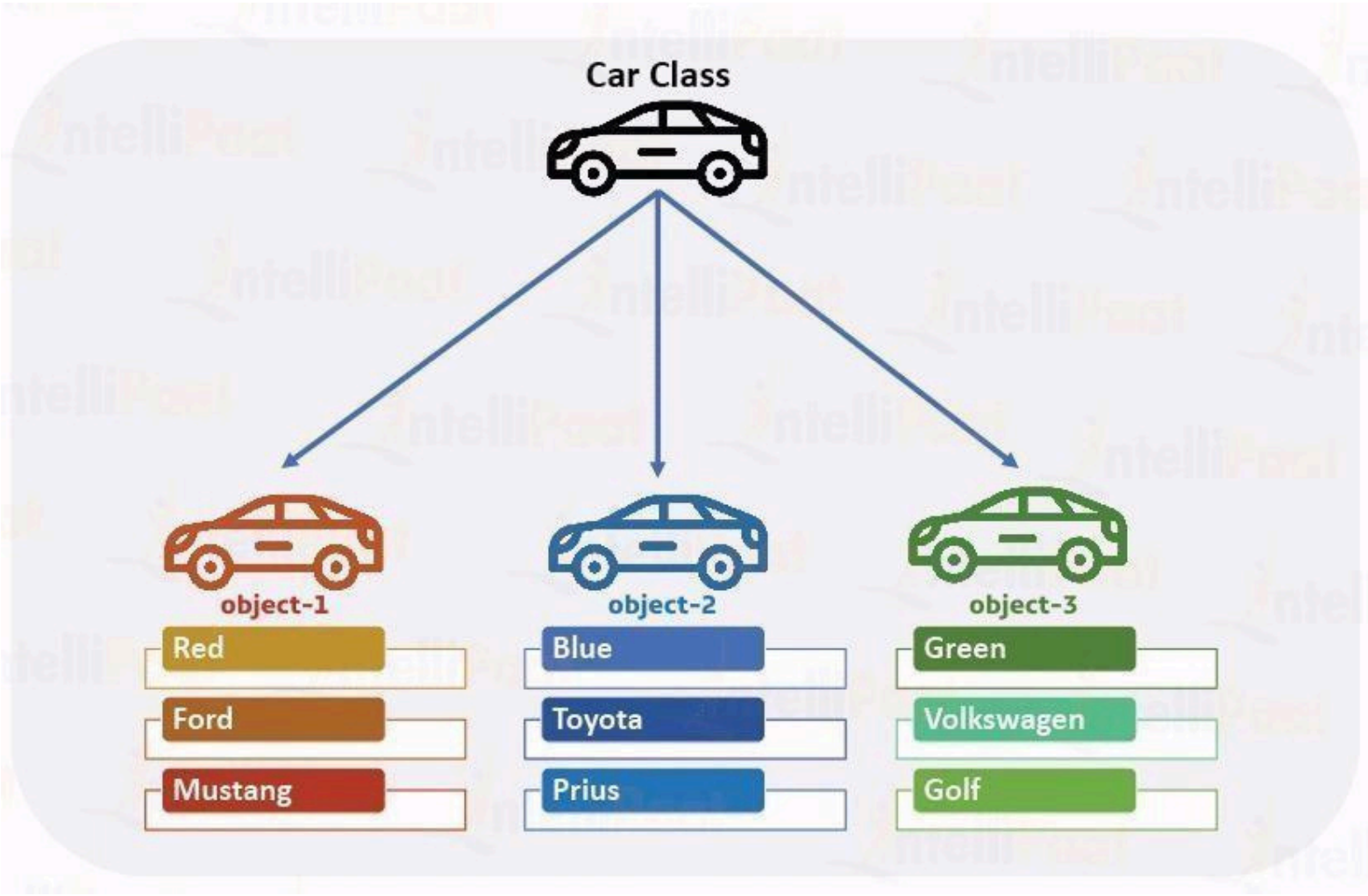
```
class MathOperations:
  def add(self,num1,num2):
    return num1+num2
  def multiply(self,num1,num2):
    return num1*num2

#Creating an object
ob1=MathOperations()

#Calling methods with arguments and object
sum_result=ob1.add(10,5)
product_result=ob1.multiply(10,5)

print("Sum: ",sum_result)
print("Product: ",product_result)
```

```
Output:

Sum:    15
Product:    50
```

Did you observe?

"What is the difference you observed from the above two example cases of classes and objects"

Simple Illustration for Classes:



CONCLUSION: ఇప్పుడు Rameshకి Classes & Objects అంటే

ఏంటో ఒక Basic idea వచ్చింది

Coding & Maths practice చేస్తేనే you'll get a complete idea.

Because **"Practice makes man perfect".**

Day-09 Coding Practice:

1. Create a Simple Class

[Compiler Link]

✅ **Task:** Create a class Car with brand and model. Add a method show() to print the details.

📌 **Example Output:** **Brand: Toyota, Model: Corolla**

2. Using _init_() Constructor

✅ **Task:** Create a Person class with name and age. Use _init_() to set values.

📌 **Example Output:** **Name: Alice, Age: 25**

3. Method for Calculation

✅ **Task**: Create a Rectangle class with length and width. Add a method area().

📌 **Example Output:** **Area: 50**

4. Method with Parameters

✅ **Task:** Modify Rectangle so area() takes length and width as arguments.

📌 **Example Output:** **Area: 30**

5. Change Attribute Value

✅ **Task:** Create a Student class with name and grade. Update grade after creating the object.

📌 **Example Output:** **Old Grade: 9**

New Grade: 10

6. Default Values in Class

✅ **Task:** Create a BankAccount class where balance = 0 by default. Add deposit() and withdraw() methods.

📌 **Example Output:** **Deposited: $500**

Balance: $500

Withdrawn: $200

Balance: $300

7. Class Attribute vs Instance Attribute

✅ **Task:** Create a Dog class with species = "Mammal" (class attribute) and name (instance attribute).

📌 **Example Output:** **Dog 1: Mammal, Name: Max**

Dog 2: Mammal, Name: Bella

8. Returning Values Instead of Printing

✅ **Task:** Modify Car class so show() returns car details instead of printing them.

📌 **Example Output:** **Car Details: Toyota Corolla**

9. Using Multiple Methods

✅ **Task:** Create a Circle class with radius. Add area() and circumference().

📌 **Example Output:** **Area: 78.5**

Circumference: 31.4

10. Creating Multiple Objects

✅ **Task:** Create a Book class with title and author. Create three book objects.

📌 **Example Output:** **Title: Harry Potter, Author: J.K. Rowling**

Title: The Hobbit, Author: J.R.R. Tolkien

Title: The Alchemist, Author: Paulo Coelho

EE.iNfo

PYTHON in 21 Days

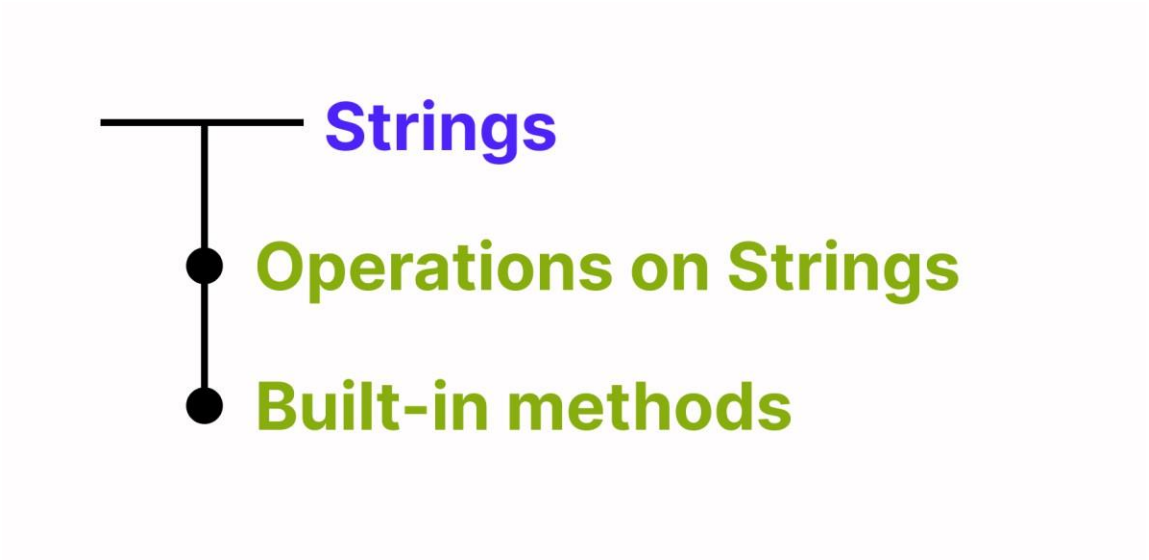
-Crack any coding interview-

DAY 10: EE.INFO

“Every Software/application works on Data”

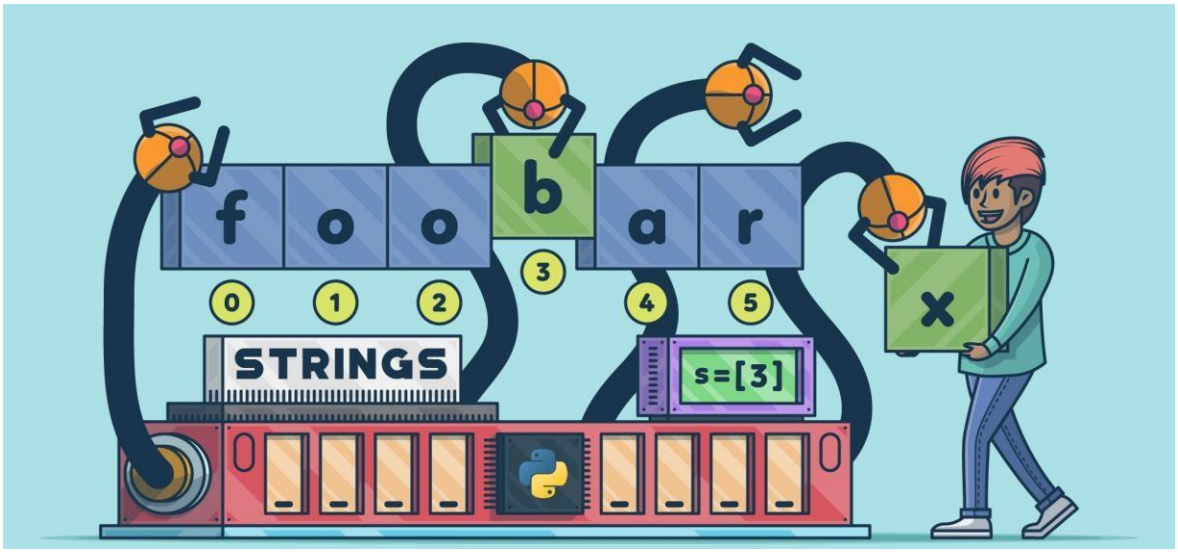
“Strings in Python”

Note: very important for coding rounds



10. STRINGS

ముందుగా String అంటే ఏంటి?



Definition: Strings are nothing but a word, a sentence or a number, surrounded by either single quotation marks(‘ ‘) / double quotation marks(“ “) in python.

- **‘hello’** is a string
- **“hello”** is also a string

10.1 String Creation, Printing:

```
s1="Hello World!"
print(s1)
```

```
Output
Hello World!
```

10.2 Operations on Strings:

1.ACCESSING CHARECTERS IN STRINGS:

Try This:

```
a="ramesh"
print(a[2])
print(a[-2])
```

What is the output? [link]

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

Output=? [Execute, See the output]

2.STRING SLICING:

Slicing అంటే ➔

String slicing means extracting a part (or) a portion of a string.

Syntax: string [start : end : step]

- start-> starting index (included)
- end -> ending index (excluded)
- skip-> steps to jump (optional)

Python Indexing:

Ex:

L	o	v	e		P	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10

```
s="Love Python"
print(s[2:8])
```

Output

```
ve Pyt
```

```
s="Love Python"
print(s[2:8:2])
```

Output

```
v y
```

```
s="Love Python"
print(s[0:10:2])
```

Output=?

```
s="Love Python"
print(s[:4])
```

Output=?

```
s="Love Python"
print(s[2:])
```

Output=?

```
s="Love Python"
print(s[::-1])
```

Output=? [hint: reversing a string]

```
s="Love Python"
print(s[-5:-2])
```

```
Output
yth
```

```
s="Love Python"
print(s[-2:-5:-1])
```

```
Output
oht
```

Don't see the below code, Try your own code first:

You have given a string s1="Kundi enduku tannav ra" to reverse it and print it.

Code:

```
s1="Kundi enduku tannav ra"
rs1=s1[::-1]
print(rs1)
```

(or)

```
print(s1[::-1])
```

Output:

```
Output
ar vannat ukudne idnuK
```

DID YOU KNOW ?

String‌లో characters పెద్దోడు లాంటివి

అవి ఎవరికోసం మారవు (immutable).

Ex:

```
s="pedhoda"
s[2]="k"
print(s)
```

```
Output
ERROR!
Traceback (most recent call last):
  File "<main.py>", line 4, in <module>
TypeError: 'str' object does not support item assignment
```

3.BASIC OPERATIONS:

1.Concatenation (+):

Code:

```
s1="pedhodu"
s2="chinnodu"
print(s1+s2)
```

Output=?

2. Repetition (*):

Code:

```
s1="Ha"
print(s1*3)
```

Output

```
HaHaHa
```

3. Membership (in):

Code:

```
s="python"
print("y" in s)
```

Output

```
True
```

4. Length (len()):

Code:

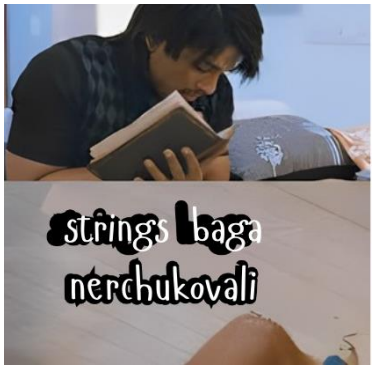
```
s1="chinnodu"
print(len(s1))
```

Output

```
8
```

4. STRING BUILT-IN FUNCTIONS / METHODS:

Built-in methods are highly important and useful in coding.



1. Case changing Methods:

```
s="hello world"
print(s.upper())
#"HELLO WORLD" (converts to uppercase)

print(s.lower())
#"hello world" (converts to lowercase)

print(s.title())
#"Hello World" (capitalizes 1st letter in each word)

print(s.capitalize())
#"Hello world" (capitalizes 1st letter in the sentence)

print(s.swapcase())
#"HELLO WORLD"-->"hello world"
```

2.Trimming & Padding:

Trimming:

```
s="python"
print(s.center(10,"-"))
#"--python--" (center align with padding)

print(s.ljust(10,"-"))
#"python----  " (left align with padding)

print(s.rjust(10,"-"))
#"----python" (right align with padding)
```

Padding:

```
s="  hello  "
print(s.strip())
#"hello" (removes spaces from both sides)

print(s.lstrip())
#"hello  " (removes spaces from left side)

print(s.rstrip())
#"  hello" (removes spaces from right side)
```

Can you try this ?

Code:

```
s="cheppara"
print(s.center(11,"*"))

print(s.ljust(11,"*"))

print(s.rjust(11,"*"))
```

Can you Guess the output without executing ?

3. Search & Replace Methods:

```
s="hello world, welcome to the world"
print(s.find("world"))
# 6 (Finds First occurence of "world")

print(s.rfind("world"))
# 28 (Finds last occurence of "world")

print(s.index("world"))
# 6 (same as find, but raises an error if not found)

print(s.replace("world","python"))
# "hello world, welcome to the python" (replaces all occurences of
                                                    "world")
```

4. Splitting & Join Methods:

1. Splitting: Converts string into list with a given character.

Ex-1:

```
s="manushulu antene manchollu"
print(s.split(" "))
```

```
Output
['manushulu', 'antene', 'manchollu']
```

Ex-2:

```
s="apple,banana,grape"
print(s.split(","))
```

```
Output
['apple', 'banana', 'grape']
```

1. Joining Methods:

Ex-2:

```
words=["Hello","world"]
print(" ".join(words))
```

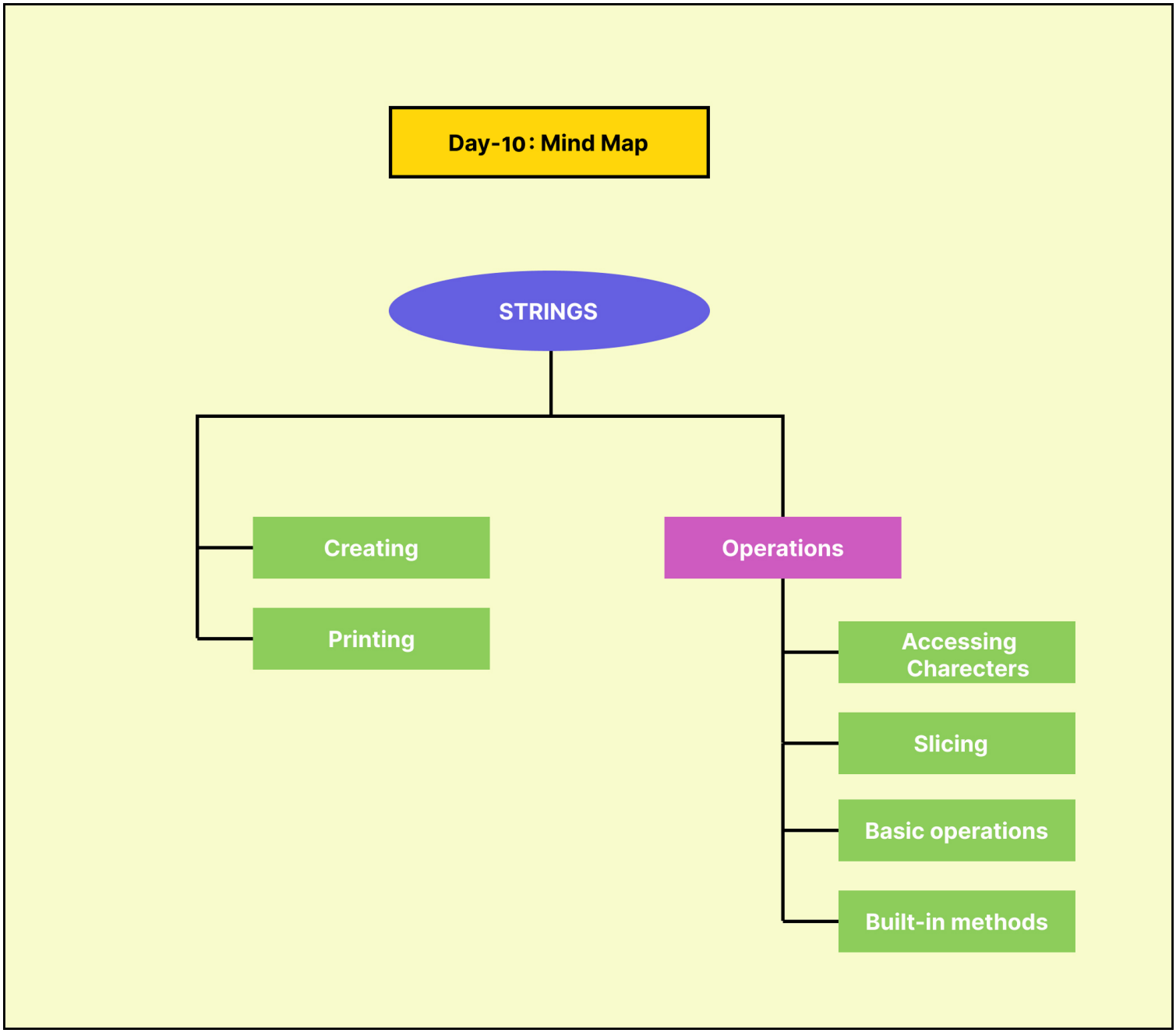
```
Output
Hello world
```

Task: Give your own example for each above string operations

Day-10 Summary:

What we have learnt on Day-10?

1. What are Strings and it's operations.
2. Accessing strings, String slicing, basic operations like [concatenation(+),repetition(*),membership(in),len()].
3. Most used string built-in methods like **s.replace()**.



EE.iNfo

PYTHON in 21 Days

-Crack any coding interview-

Day-11 Coding Practice Sheet

Here are **10 coding questions** covering all major topics from **Day 10 (Strings in Python):**

1. String Creation & Printing

Question: Create a string variable containing your full name and print it.

2. Accessing Characters in a String

Question: Given `s = "Python"`, print the first and last character using **positive** and **negative indexing**.

3. String Slicing

Question: Given `s = "HelloWorld"`, extract and print:
- First 5 characters
- Last 5 characters
- The string in reverse order

4. Concatenation (+) and Repetition (*)

Question: Given `s1 = "Hello"` and `s2 = "World"`, perform and print:
- `s1 + s2`
- `s1 * 3`

5. Membership Operator (in)

Question: Given `s = "Coding is fun"`, check if the word `"fun"` is present in the string.

6. Finding Length of a String

Question: Write a Python program that takes a user-inputted string and prints its length using `len()`.

7. String Built-in Methods (Case Changing)

Question: Given `s = "pYtHoN Is AwEsOmE"`, print the string in:
- **Uppercase**
- **Lowercase**
- **Title case**

8. Search & Replace Methods

Question: Given `s = "I love Python"`:
- Replace `"Python"` with `"JavaScript"` and print the new string.

9. Splitting & Joining Methods

Question: Given `s = "apple,banana,grape"`, split the string into a **list** and then join it back using `"-"` as a separator.

10. Reverse a String without Slicing

Question: Given `s = "reverse"`, reverse the string using a **loop** and print it.

write your code here [LINK]

LeetCode

1️⃣ Merge Strings Alternately (Two-Pointer)

🔗 **Merge Strings Alternately – LeetCode 1768**
📝 **Problem:** Given two strings `word1` and `word2`, merge them alternately. If one string is longer, append the rest at the end.
💼 **Asked in:** **Google, Amazon, Microsoft** [LINK]

write your code here

2️⃣ Defanging an IP Address (String Replacement)

🔗 **Defanging an IP Address – LeetCode 1108**
📝 **Problem:** Given a valid IP address, replace every `"."` with `"[.]"`.
💼 **Asked in:** **Google, Amazon, Apple** [LINK]

write your code here

3️⃣ Goal Parser Interpretation (String Manipulation)

🔗 **Goal Parser Interpretation – LeetCode 1678**
📝 **Problem:** Given a command string containing `"G"`, `"()"` (interpreted as `"o"`), and `"(al)"` (interpreted as `"al"`), return the interpreted string.
💼 **Asked in:** **Facebook, Amazon, Google** [LINK]

write your code here

4️⃣ Check If Two Strings are Equivalent (String Join)

🔗 **Check If Two String Arrays are Equivalent – LeetCode 1662**
📝 **Problem:** Given two string arrays `word1` and `word2`, return `True` if they form the same string when concatenated.
💼 **Asked in:** **Apple, Google, Microsoft** [LINK]

write your code here

5️⃣ Count the Number of Vowel Strings in a Range (String Counting)

🔗 **Count the Number of Vowel Strings in a Range – LeetCode 2586**
📝 **Problem:** Given a list of words and a range `[left, right]`, count words that **start and end with a vowel** (`a, e, i, o, u`).
💼 **Asked in:** **Amazon, Meta, Adobe** [LINK]

write your code here

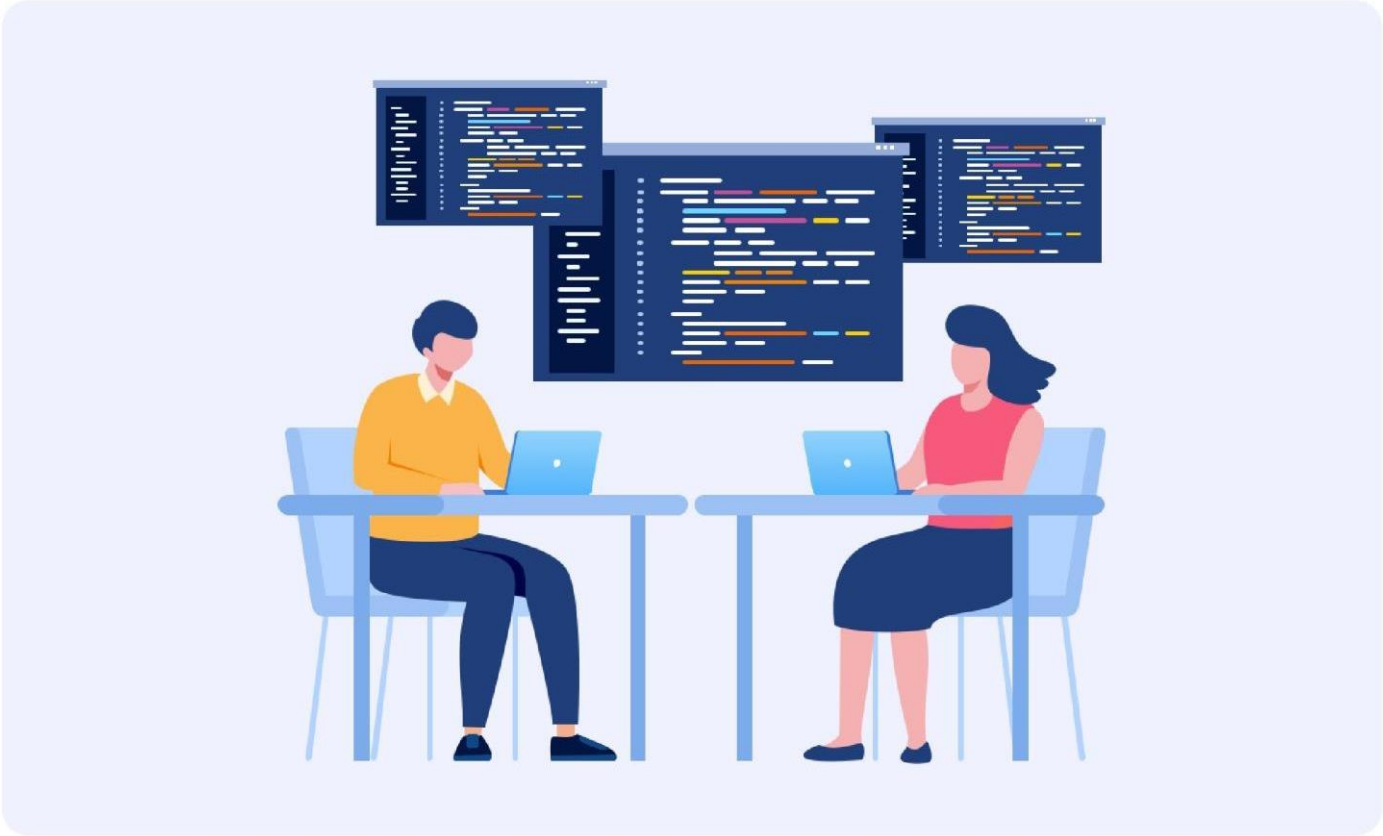
🔥 **These **easy string-based** LeetCode problems are great for beginners and frequently asked in interviews!** 🚀

EE.iNfo

PYTHON in 21 Days

-Crack any coding interview-

DAY 12: EE.INFO



“Everything is possible” – Swamy Vivekananda

“LISTS IN PYTHON”

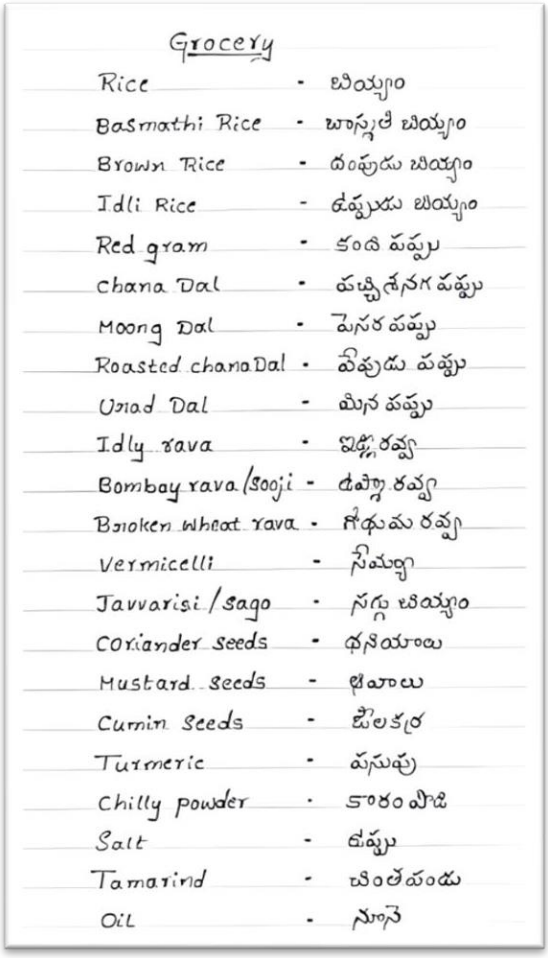
“80% outputని 20%inputతో వస్తుంది

ఆ 20%లో “list” concept ఉంటుంది.” - Pareto Principle

12. LISTS

Suresh: Ramesh! list అంటే ఏంటి?

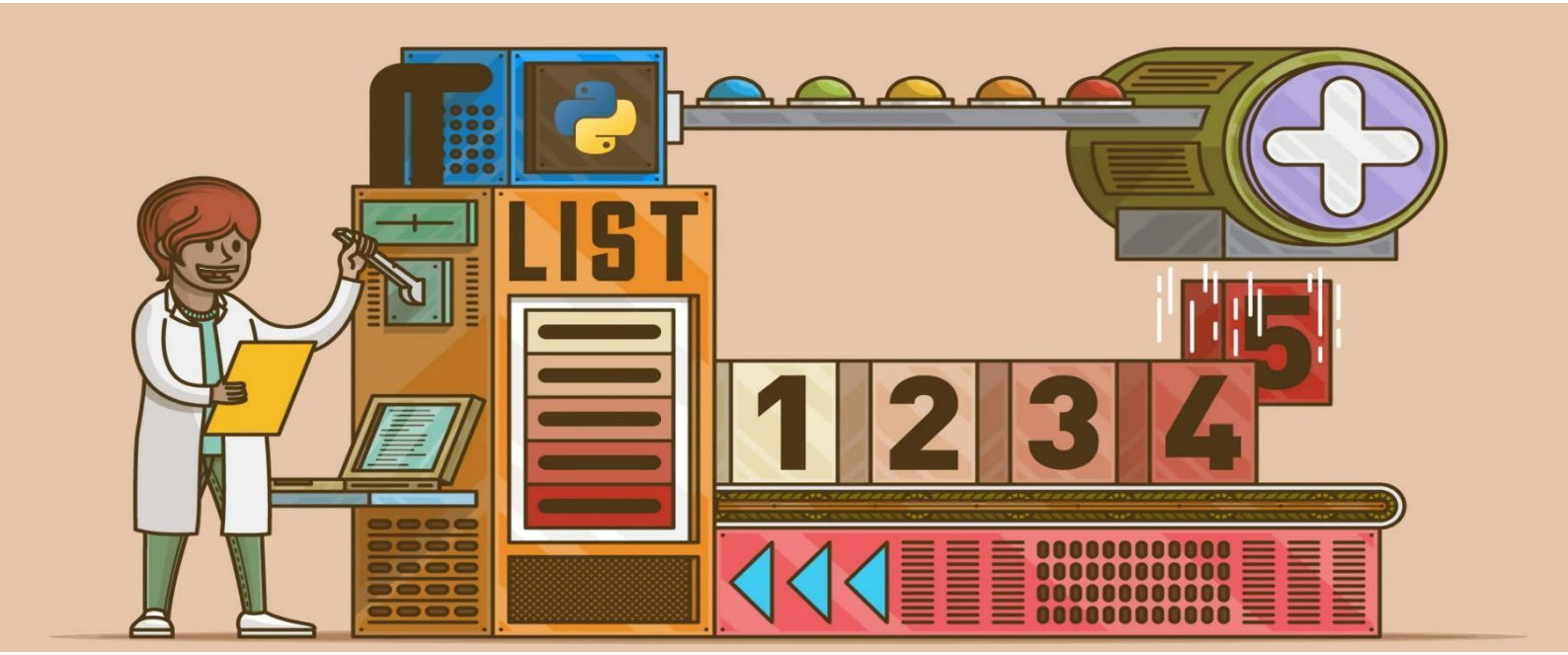
Ramesh: మనుం కిరాణా సామాన్లు రాస్తాం కదా! అదే list.


Grocery

Rice	బియ్యం
Basmathi Rice	బాస్మతి బియ్యం
Brown Rice	దంపుడు బియ్యం
Idli Rice	ఇడ్లి బియ్యం
Red gram	కంది పప్పు
Chana Dal	పచ్చిశనగ పప్పు
Moong Dal	పెసర పప్పు
Roasted chana Dal	వేపుడు పప్పు
Urad Dal	మిన పప్పు
Idly rava	ఇడ్లి రవ్వ
Bombay rava (Sooji)	ఉప్మా రవ్వ
Broken Wheat rava	గోధుమ రవ్వ
Vermicelli	సేమ్యా
Javvarisi / Sago	సగ్గు బియ్యం
Coriander seeds	ధనియాలు
Mustard seeds	ఆవాలు
Cumin seeds	జీలకర్ర
Turmeric	పసుపు
Chilly powder	కారం పొడి
Salt	ఉప్పు
Tamarind	చింతపండు
Oil	నూనె

Actualగా, ఇక్కడ ramesh చెప్పింది almost correct. ఒక Paperలో ఎలా అయితే multiple items రాస్తామో

-variableలో multiple itemsని store చేస్తాం. దానినే Pythonలో **list** అని అంటాం.



List Definition:

A list in python is used to store multiple values in a single variable of same (or) different data types.

Ex:

```
L1=["Ramesh",20,1,2,"Suresh",3.2]
L2=["Ramesh","Suresh","Python"]
L3=[1,2,3,4,5,10,12]
print(L1)
print(L2)
print(L3)
```

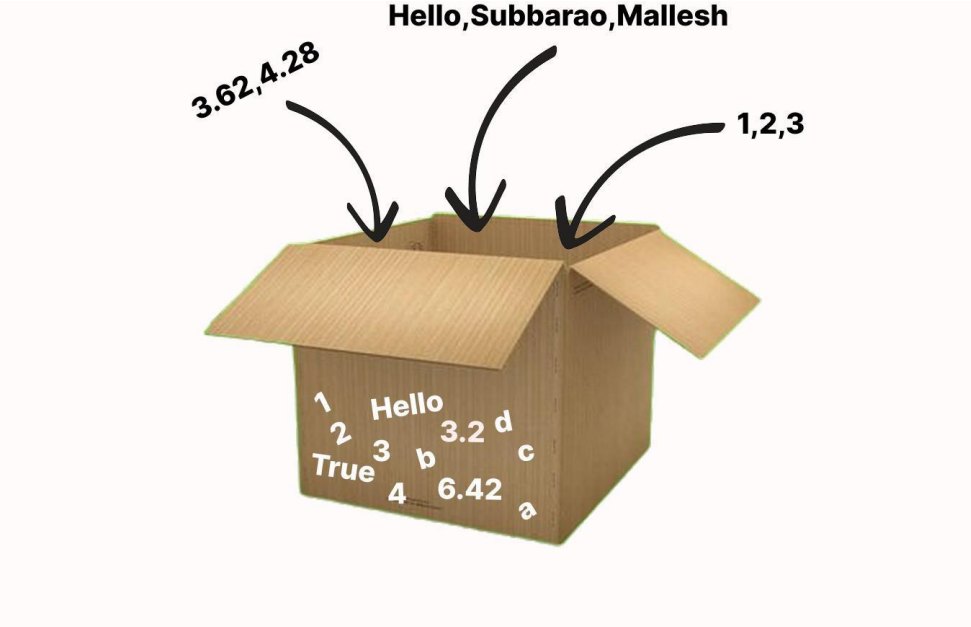
Output:

```
['Ramesh', 20, 1, 2, 'Suresh', 3.2]
['Ramesh', 'Suresh', 'Python']
[1, 2, 3, 4, 5, 10, 12]
```

12.1 CREATING A LIST:

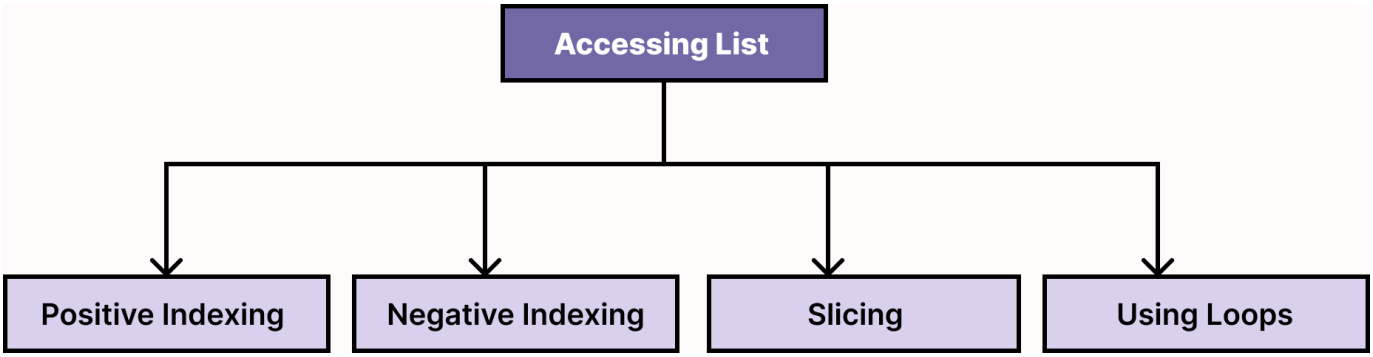
A list is created using square brackets [ ] , with elements seperated by commas as shown as below:

```
# Creating different types of lists
# Strings list
fruits=["apple","banana","cherries"]

# Integer list
numbers=[1,2,3,4,5]

# Mixed list
mixed=["hello",42,3.5,True]

# Empty list
L1=[]
```



[List imagination]

12.2 ACCESSING LIST:



1.Positive Indexing:(→)

Starts from 0.

Index

EX:

```
fruits=["apple","banana","cherry"]
print(fruits[0])
print(fruits[1])
print(fruits[2])
```

Output:

```
apple
banana
cherry
```

2.Positive Indexing:(←)

Starts from -1, from the end.

Apple	Banana	Cherry
-3	-2	-1

Code:

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

print(fruits[-1])

print(fruits[-2])
```

Output=?

3.Slicing:

Syntax: List_name [start:end+1]

Code:

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

print(fruits[0:2])

print(fruits[:2])

print(fruits[1:])

print(fruits[-2:])
```

```
Output:

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

4. Using Loops:

Approach-1:

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

for i in range(0,len(fruits)):

    print(fruits[i])
```

```
Output:

apple
banana
cherry
```

Approach-2:

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

for i in fruits:

    print(i)
```

```
Output:

apple
banana
cherry
```

Which approach do you prefer from the above?

Approach-3:

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

for index,fruits in enumerate(fruits):

    print(index,fruits)
```

```
Output:

0 apple
1 banana
2 cherry
```

Did you know?

"i" is just a variable name, you can replace any letter/word. But from beginning coders have been using "i" as the variable name
"అలవాటైపోయింది".

Task: Create a list and access using above 4 methods.

12.2.3 OPERATIONS ON LIST:

1. Modifying a list:

Lists are mutable, means their elements can be changed.

Code:

```
L1=[10,20,30,40,50]

L1[1]=25

print(L1)
```

Output:

```
[10, 25, 30, 40, 50]
```

2. Adding elements to a list:

1. append( ): Adds an element to the end.

Code:

```
L1.append(60)
print(L1)
```

```
Output:
[10, 25, 30, 40, 50, 60]
```

2. insert(index,value): Adds an element at index position.

Code:

```
L1.insert(2,15)
print(L1)
```

```
Output:
[10, 25, 15, 30, 40, 50, 60]
```

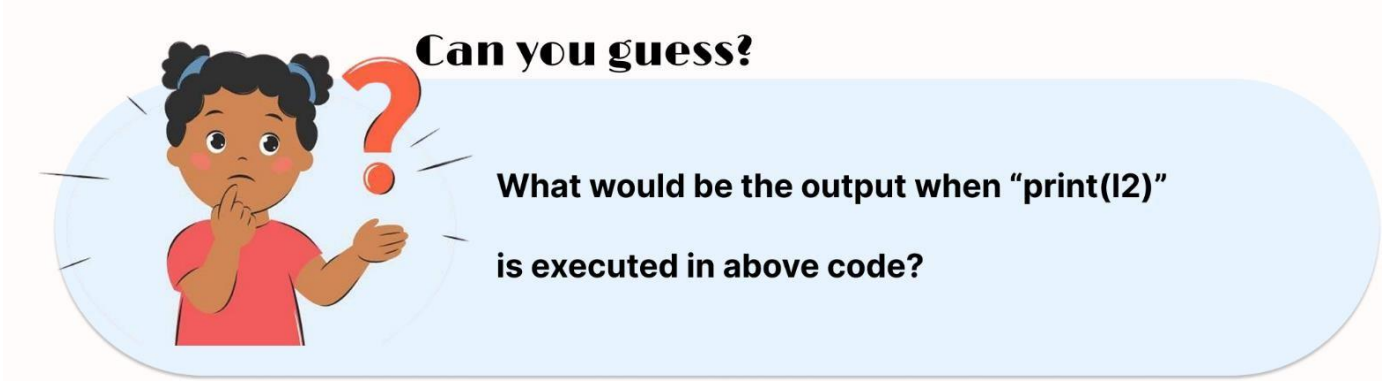
3. extend( ): Merges two lists.

Syntax: list1.extend (list2)

Code:

```
L2=[70,80,90]
L1.extend(L2)
print(L1)
```

```
Output:
[10, 25, 15, 30, 40, 50, 60, 70, 80, 90]
```



3. Removing elements from a list:

1. remove(value): Removes by value (first occurrence only).

Code:

```
L1.remove(30)
print(L1)
```

```
Output:
[10, 25, 15, 40, 50, 60, 70, 80, 90]
```

2. **pop(index):** Removes by index position (default:last element).

Code:

```
L1.pop(2)  # removes 15
L1.pop()   # removes last element i.e 90
print(L1)
```

Output:

```
[10, 25, 40, 50, 60, 70, 80]
```

3. **del:** Deletes an element or entire list.

Code:

```
del L1[2]
print(L1)
```

Output:

```
[10, 25, 50, 60, 70, 80]
```

Code:

```
del L2
print(L2)
```

Output:

```
Traceback (most recent call last):
  File "/main.py", line 25, in <module>
    print(L2)
          ^^
NameError: name 'L2' is not defined. Did you mean: 'L1'?
```

4. clear(): Empties the list.

Code:

```
L1.clear()

print(L1)
```

```
Output:

[]
```

4. Checking if an element exists:("in")

Code:

```
L1=[10,20,30,40]

if 30 in L1:
  print("30 is in list")
```


```
Output:

30 is in list
```

5. Lists -Built-in Functions:

- len(),max(),min()

1. len():

```
L1=[10,20,30,40]

print(len(L1))
```

```
Output:

4
```

2.max():

```
L1=[10,20,30,40]
print(max(L1))
```

```
Output:
40
```

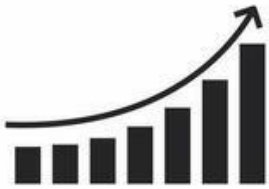
3.min():

```
L1=[10,20,30,40]
print(min(L1))
```

```
Output:
10
```

6.Sorting list:

1.sort(): Ascending order



Code:

```
L1=[10,30,20,40,50]
L1.sort()
print(L1)
```

```
Output:
[10, 20, 30, 40, 50]
```

2.<u>sort(reverse=True)</u>: Descending order

Code:

```
L1=[10,20,30,40,50]
L1.sort(reverse=True)
print(L1)
```

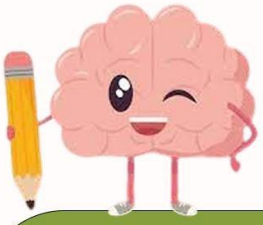
```
Output:

[50, 40, 30, 20, 10]
```

3.<u>sorted()</u>: Returns a new sorted list (Original remains unchanged)

Code:

```
L1=[10,60,30]
L2=sorted(L1)

print(L2)
print(L1)
```

```
Output:

[10, 30, 60]
[10, 60, 30]
```



Did you Observe?

Very important difference between soorted() and sort()

Ex: sorted(L1) : L1.sort()

```
L1=[10,60,30]
L2=sorted(L1)
print(L2)
print(L1)
L1.sort()
print(L1)
```

OUTPUT: Execute and analyze the output

7. Nested list: "List inside list"

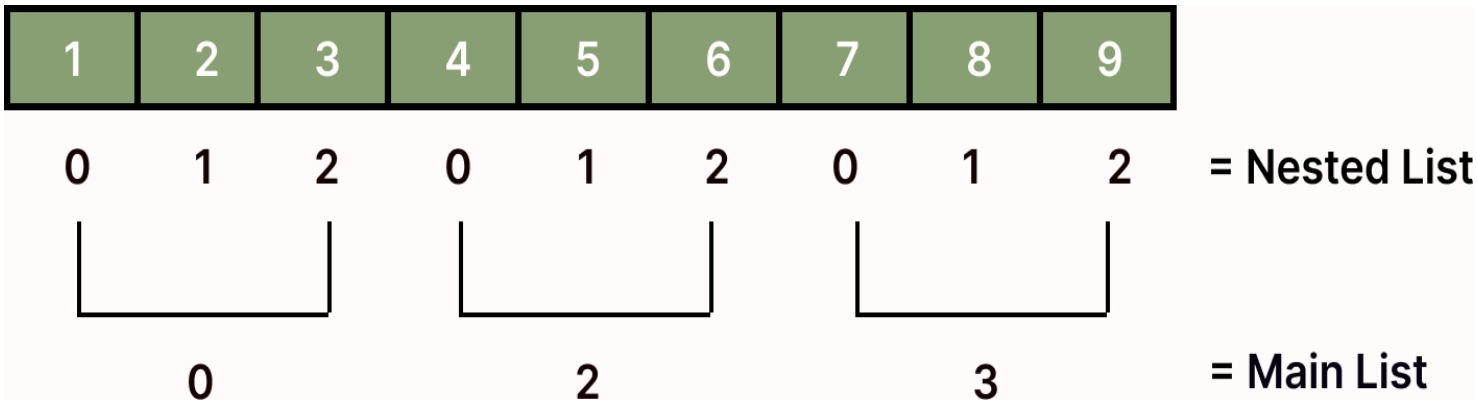
Code:

```
m1=[[1,2,3],[4,5,6],[7,8,9]]
print(m1[1])
print(m1[1][2])
```

Output:

```
[4, 5, 6]
6
```

ILLUSTRATION:

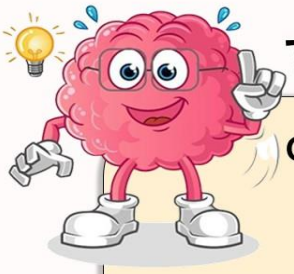


8. Count: Returns count of a value from the list.

Code:

```
L1=[1,1,20,30,40,1]

print(L1.count(1))
```

```
Output:

3
```



Try this:

Guess the output ?

```
t=[10,20,30]

print(t.count(50))
```

output = ?

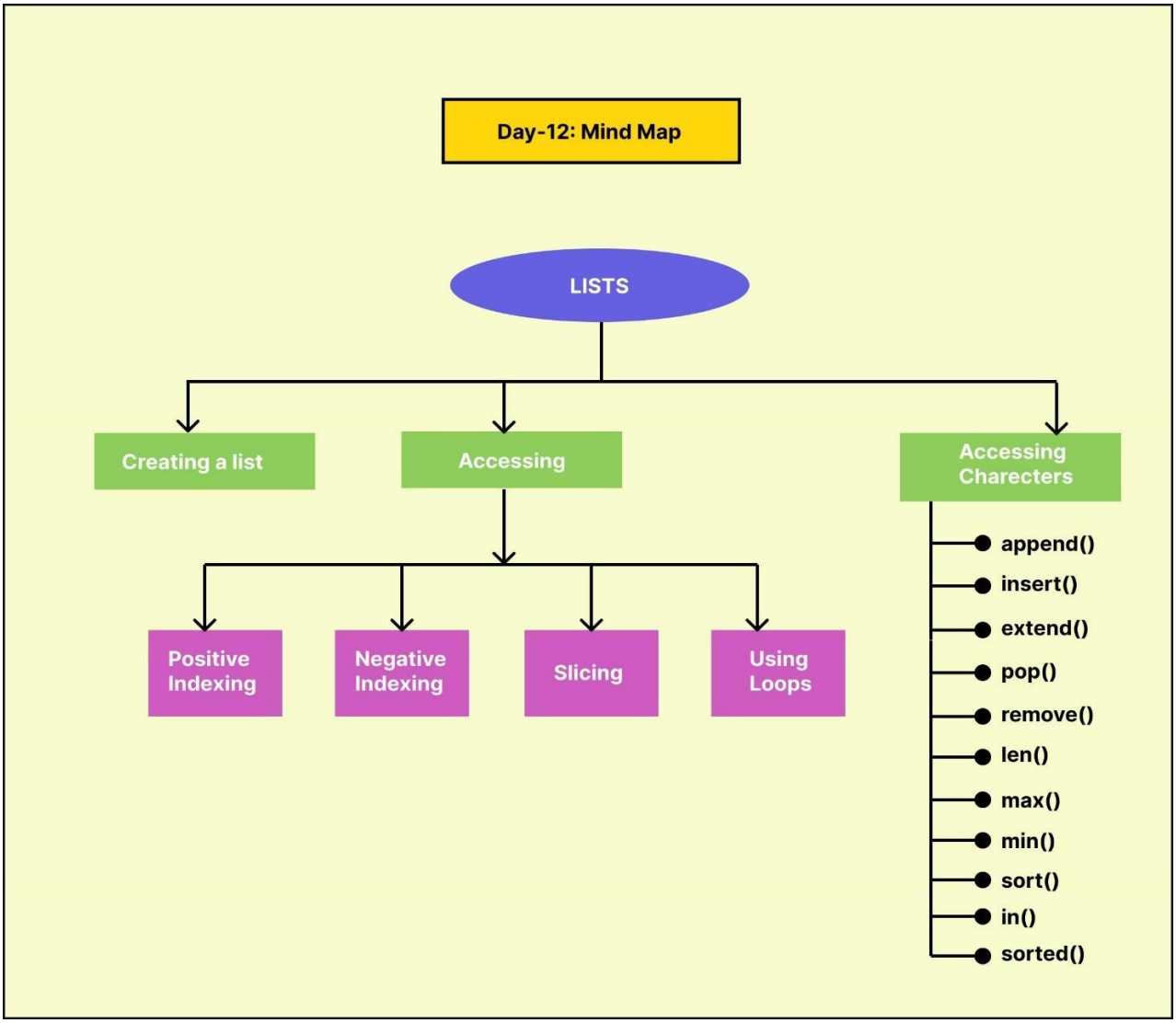
List Methods – Summary Table:

Method	Description
append(x)	Adds x to the end of the list
insert(i,x)	Adds x at index i
list1.extend(list2)	Merges two lists
remove(x)	Removes first occurrence of x
pop(i)	Returns & removes the item at index i
pop()	Removes last item from the list
clear()	Removes all items from the list
in	Checks whether the item is existing in the list or not
len(l1)	Returns length of l1
max(l1)	Returns the maximum value of list l1
min(l1)	Returns the minimum value of list l1
Index(r)	Returns index of r
.sort()	Sorts the list in ascending order
reverse()	Reverses the list
sorted()	Returns sorted list but original list is unchanged

ఇప్పుడుRameshకి sureshకి lists అంటే ఒక clarity వచ్చింది. మరి మీకు?

What we have learnt on day 12:

1. Lists are mutable, ordered and allow duplicates.
2. Lists uese zero-based indexing.
3. Lists can store different datatypes (integer, float, boolean).
4. Lists use methods like append(), insert(), remove() to modify lists.



EE.iNfo

PYTHON in 21 Days

-Crack any coding interview-

Day-13 Coding Practice Sheet

Here are some easy coding questions based on the "Lists in Python" topic:

1. Creating a List

Question: Create a list of your five favorite movies and print it.
[Try it here] [LINK]

2. Accessing Elements in a List

Question: Given a list `numbers = [10, 20, 30, 40, 50]`, print the first and last elements using both positive and negative indexing.
[Try it here] [LINK]

3. Slicing a List

Question: Given `fruits = ["apple", "banana", "cherry", "mango", "orange"]`, print only the first three elements using slicing.
[Try it here] [LINK]

4. Using Loops with Lists

Question: Write a Python program to iterate through a list `colors = ["red", "green", "blue", "yellow"]` and print each color.
[Try it here] [LINK]

5. Modifying a List

Question: Given `marks = [80, 90, 70, 85]`, change the second element to `95` and print the modified list.
[Try it here] [LINK]

6. Adding Elements to a List

Question: Start with an empty list and use `append()` to add the numbers `5, 10, 15, 20` to it, then print the list.
[Try it here] [LINK]

7. Removing Elements from a List

Question: Given `animals = ["cat", "dog", "elephant", "tiger"]`, remove `"elephant"` using the `remove()` method and print the updated list.
[Try it here] [LINK]

8. Sorting a List

Question: Given `scores = [40, 10, 90, 70, 30]`, sort the list in ascending order and print it.
[Try it here] [LINK]

9. Checking if an Element Exists in a List

Question: Given `names = ["Alice", "Bob", "Charlie", "David"]`, check if `"Bob"` is in the list and print `True` or `False`.
[Try it here] [LINK]

10. Nested Lists

Question: Create a nested list `matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]` and print the second row.
[Try it here] [LINK]

These questions cover the basics of lists in Python and align with the PDF content. Let me know if you need more! 🚀

LeetCode sample Company questions:

1️⃣ Two Sum (List Indexing & Iteration)

🔗 **Two Sum – LeetCode 1**
📝 **Problem:** Given an array of integers `nums` and an integer `target`, return indices of the two numbers such that they add up to `target`.
💼 **Asked in:** **Amazon, Google, Microsoft** [LINK]

write your code here

2️⃣ Remove Duplicates from Sorted Array (Modifying Lists)

🔗 **Remove Duplicates from Sorted Array – LeetCode 26**
📝 **Problem:** Given a sorted array, remove the duplicates **in-place** so that each element appears only once and return the new length.
💼 **Asked in:** **Facebook, Apple, Google** [LINK]

write your code here

3️⃣ Merge Two Sorted Lists (Merging Lists)

🔗 **Merge Two Sorted Lists – LeetCode 21**
📝 **Problem:** Given two sorted linked lists, merge them into a single sorted list and return it.
[LINK]
💼 **Asked in:** **Amazon, Microsoft, Adobe**

write your code here

4️⃣ Best Time to Buy and Sell Stock (Finding Min/Max in a List)

🔗 **Best Time to Buy and Sell Stock – LeetCode 121**
📝 **Problem:** Given a list of stock prices, find the best time to buy and sell to maximize profit.
💼 **Asked in:** **Google, Meta, Bloomberg** [LINK]

write your code here

5️⃣ Find the Index of the First Occurrence in a String (Searching in a List)

🔗 **Find the Index of the First Occurrence in a String – LeetCode 28**
📝 **Problem:** Given two strings `haystack` and `needle`, return the index where `needle` first appears in `haystack`, or `-1` if it doesn't exist.
💼 **Asked in:** **Amazon, Apple, Google** [LINK]

write your code here

🔥 **These **easy LeetCode problems** are great for beginners and commonly asked in interviews!** 🚀

EE.iNfo

PYTHON in 21 Days

-Crack any coding interview-

DAY 14: EE.INFO

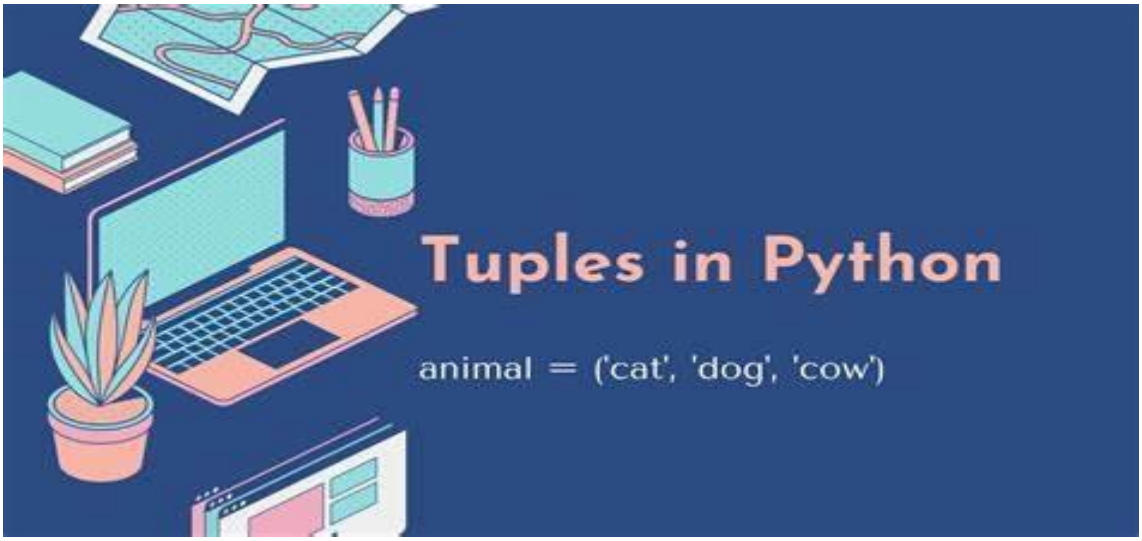
“First step is always hard, 90% will stop there. But if you can cross that one step, it is very easy”. -Author

14. TUPLES

Ramesh: నాకు day 12లో నేర్చుకున్న lists బాగా అర్థమయ్యాయి మరి ఈ tuple ఏంటి?

Suresh: List, Tuple, Dictionary, Set almost ఒక్కటే చిన్నచిన్న differences and usecases **ఉంటాయి అంతే!**

ఇప్పుడు అవి ఏంటో చూద్దాం:



Tuple Definition:

- Similar to list, tuples are used to store multiple items in a single variable.
- But the difference is Tuples are immutable (immutable means unchangeable).
- Tuples are ordered, unchangeable, and allow duplicates.

Ramesh: Unchangeable అంటే?

Suresh: ఒక సరి tuple declare చేసిన తర్వాత, దానిలో elements update చెయ్యడం కాని, కొత్త element add/ remove చెయ్యడం కాని చెయ్యలేం. క్రింద example చూడు:

14.1. CREATING A TUPLE:

Code:

```
cricketers=("virat","rohith","bumrah")
print(cricketers)
```

```
Output:

('virat', 'rohith', 'bumrah')
```

Unchangeable nature:

For ex:

```
cricketers[1]="pandya"
print(cricketers)
```

```
Output:

Traceback (most recent call last):
  File "/main.py", line 1, in <module>
    cricketers[1]="pandya"
    ^^^^^^^^^^
NameError: name 'cricketers' is not defined
```

Similar గా tupleకి కొత్తవి add చెయ్యలేం, ఉన్నవి remove చెయ్యలేం.

14.2. ACCESSING TUPLES:

Using indexing (starts from 0):

```
cricketers=("virat","rohith","bumrah")
print(cricketers[0])
print(cricketers[1])
print(cricketers[2])
```

Output:

```
virat
rohith
bumrah
```

(or)

```
for i in cricketers:
  print(i)
```

14.3. OPERATIONS ON TUPLES:

1.Length of the tuple:

```
cricketers=("virat","rohith","bumrah")
print(len(cricketers))
```

Output:

```
3
```

2.Check if item exists:

```
cricketers=("virat","rohith","bumrah")
if"virat" in cricketers:
    print("virat there")
```

Output: Guess it..

3.Tuple Slicing:

```
cricketers=("virat","rohith","bumrah")
print(cricketers[0:2])
```

Output:

```
('virat', 'rohith')
```

4.Count:

```
cricketers=("virat","rohith","bumrah")
print(cricketers.count("rohith"))
```

Output:

1

5.Index:

```
cricketers=("virat","rohith","bumrah")
print(cricketers.index("bumrah"))
```

Output:

2

Think How?

మనం tuplesలో elementsని change చెయ్యలేం.
కానీ listగా convert చేసి మార్చచ్చు.

Ex:

```
cricketers=("virat","rohith","bumrah")
temp_list=list(cricketers)
temp_list.append("jadeja")
temp_list[1]="pandya"
cricketers=tuple(temp_list)
print(cricketers)
```

Output:

```
('virat', 'pandya', 'bumrah', 'jadeja')
```

EE.iNfo

PYTHON in 21 Days

-Crack any coding interview-

Day-15 Coding Practice Sheet

Here are some easy coding questions based on the "Tuples in Python" topic:

1. Super Power Mix

-- Create a tuple of 3 super power you wish to have
-- Print those super powers using for and while loops
[Try it here] [LINK]

2. Spotify Playlist

-- Create a tuple of your favourite songs
-- check if the song “kodni koy” exists in the tuple or not & print a message.
[Try it here] [LINK]

3. Dream destinations

-- Create a tuple of your dream places
-- Print first two elements using slicing operations
[Try it here] [LINK]

EE.iNfo

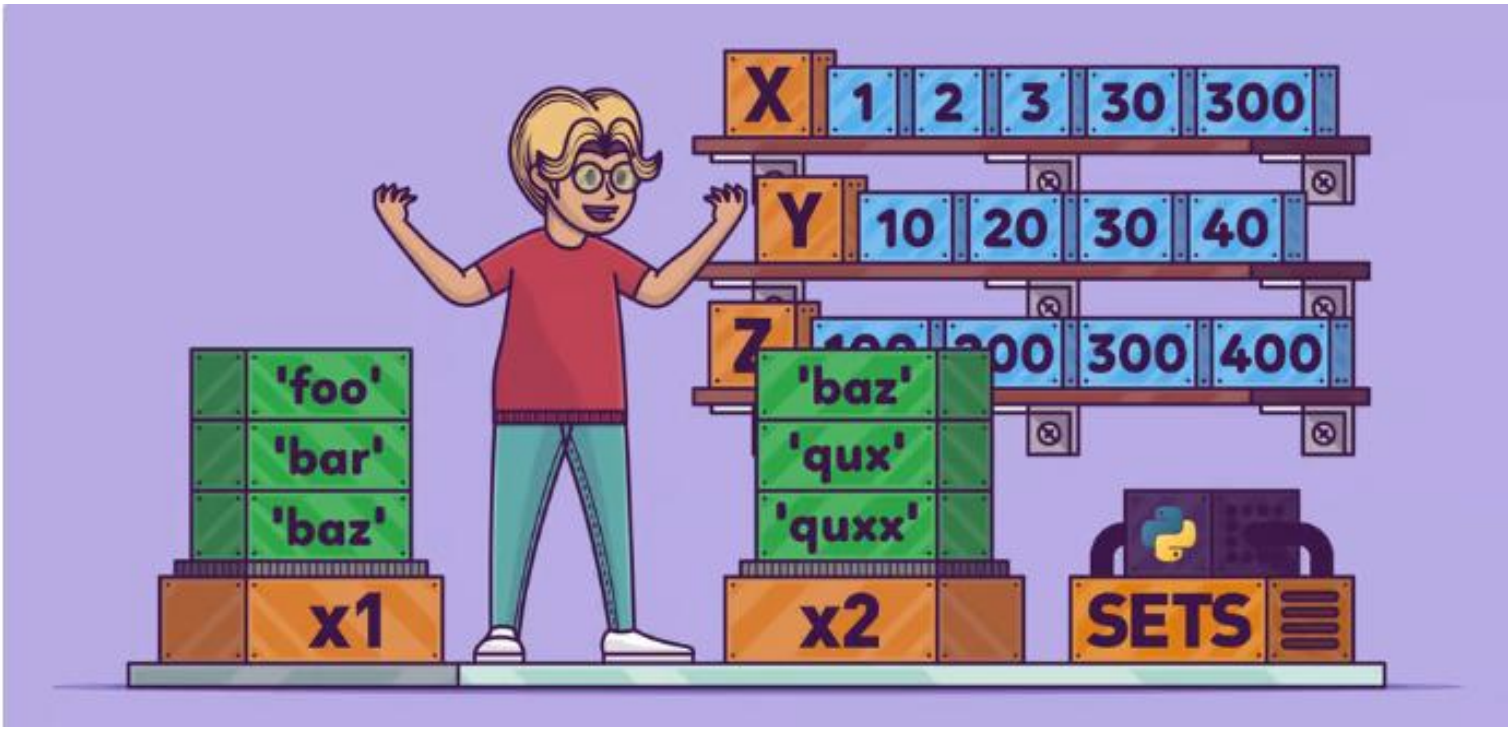
PYTHON in 21 Days

-Crack any coding interview-

DAY 16: EE.INFO

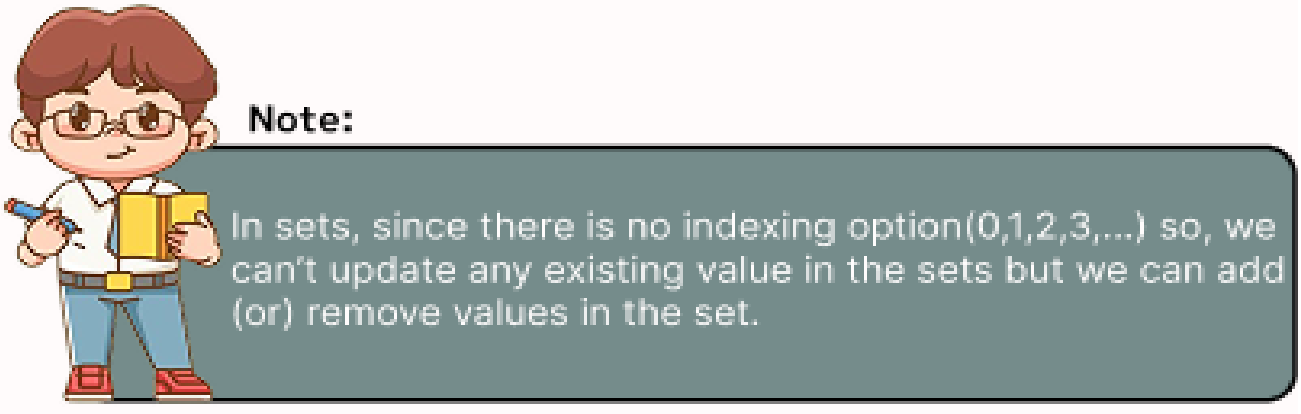
"Sets does not allow duplicates"

16. SETS



Set Definition: A set is similar to a list, but the distance is set is unordered, unindexed (no index values) and mainly does not allow duplicates.

Representation: by "{ }" (or) set()



16.1. CREATING A SETS:

1.Using curly braces:

Code:

```
n1={1,2,3}
print(n1)
```

Output:

```
{1, 2, 3}
```

Note:

The Order is random in sets. So the output changes after each execution

1.Using curly braces:

Code:

```
n1= set ([1,2,3])
print(n1)
```

Output:

```
{1, 2, 3}
```

Think and discuss:

To remember (or), learn any skill you should always think "why" after learning a new point and should connect the dots by experimenting existing Code ond Concepts. only you will master anything, Until you teach yourself, you Can't learn anything. There is no other best teacher.

NO DUPLICATES IN A SET:

If you add duplicates values, sets will automatically remove them.

Code:

```
numbers = {1,2,2,3,4,4}
print(numbers)
```

Output:

```
{1, 2, 3, 4}
```

Duplicates are automatically removed.

16.2.OPERATIONS ON SETS

1. Accessing elements in a set:

మళ్లీ చెప్తున్నా Sets అనేవి unordered. so, index values తో access చెయ్యలేం.

For Ex:

```
numbers= {1,2,3,4}
print(numbers[2])
```

```
Output:

Traceback (most recent call last):
  File "/main.py", line 2, in <module>
    print(numbers[2])
          ~~~~~~~~~^^^
TypeError: 'set' object is not subscriptable
```

LOOP THROUGH A SET:

For Ex:

```
numbers= {1,2,3,4}
for i in numbers:
    print(i)
```

```
Output:

1
2
3
4
```

2. Adding elements to a set: (add())

For Ex:

```
numbers= {1,2,3}
numbers.add(4)
numbers.add((5,6))
print(numbers)
```

Output:

```
{1, 2, 3, 4, (5, 6)}
```

3. Adding two sets:

For Ex:

```
s1={1,2,3}
s2={4,5,6}
s1.update(s2)
print(s1)
```

Output:

```
{1, 2, 3, 4, 5, 6}
```

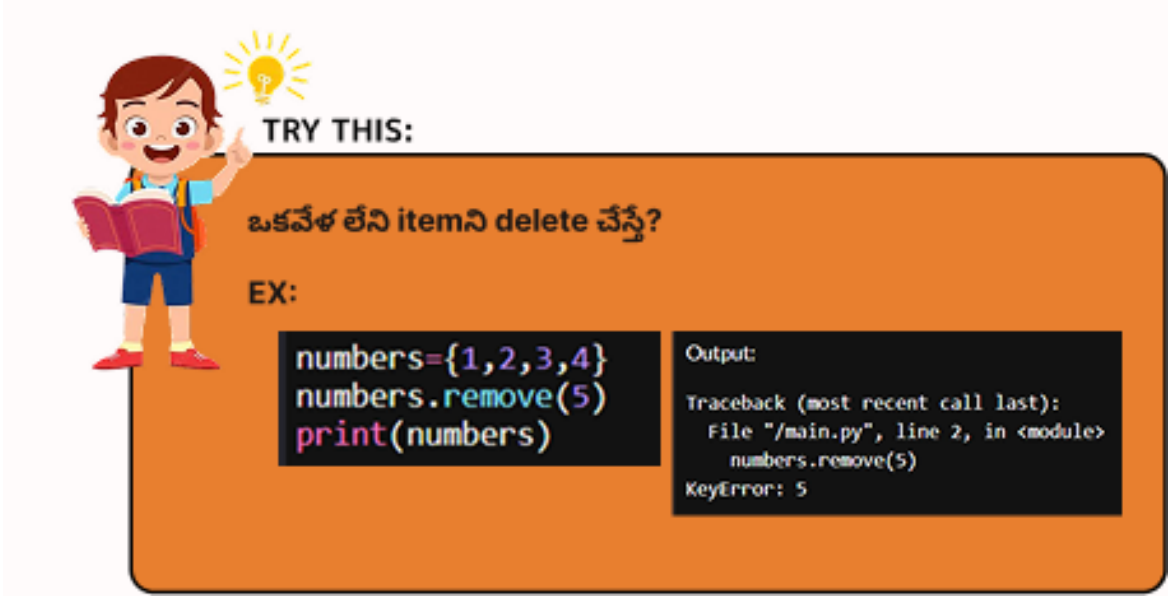
4. Removing elements from a set:

For Ex:

```
numbers={1,2,3,4}
numbers.remove(2)
print(numbers)
```

Output:

```
{1, 3, 4}
```



4.Clear all Items:

For Ex:

```
numbers={1,2,3,4}
numbers.clear()
print(numbers)
```

Output:

```
set()
```

5.Set Operations:

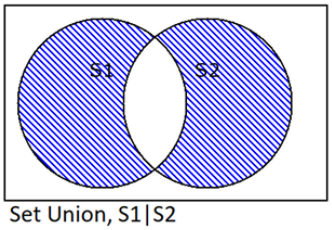
1.UNION (combining two sets)

Code:

```
a={1,2,3}
b={3,4,5}
print(a.union(b))
```

Output:

```
{1, 2, 3, 4, 5}
```



Set Union, S1|S2

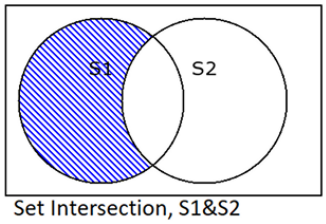
2.INTERSECTION (common items)

Code:

```
a={1,2,3}
b={3,4,5}
print(a.intersection(b))
```

Output:

```
{3}
```



Set Intersection, S1&S2

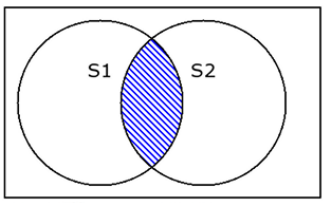
3.DIFFERENCE (items only in S1)

Code:

```
a={1,2,3}
b={3,4,5}
print(a.difference(b))
```

Output:

```
{1, 2}
```



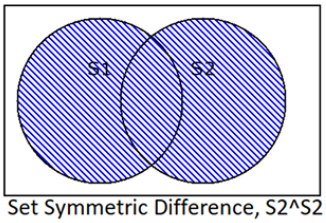
Set Difference, S1-S2

4.SYMMETRIC_DIFFERENCE (non-common in both)

Code:

```
a={1,2,3}
b={3,4,5}
print(a.symmetric_difference(b))
```

```
Output:

{1, 2, 4, 5}
```



Set Symmetric Difference, S2^S2

5.CHECK IF AN ITEM EXISTS:

Code:

```
numbers={10,20,40}
if 30 in numbers:
  print("30 exists")
else:
  print("30 is not there")
```

Output: Execute and find out on your own…

6.SET LENGTH:

Code:

```
numbers={10,20,40}
print(len(numbers))
```

```
Output:

3
```

7.CONVERT SET TO LIST:

Code:

```
numbers={10,20,30}
nlist=list(numbers)
print(nlist)
```

```
Output:

[10, 20, 30]
```

8.IMMUTABLE ELEMENTS:

Set only allow immutable(unchangeable) items like

- ✓ Numbers, Strings,Tuples
- × Lists, Dictionaries, Stes

Wrong Example:

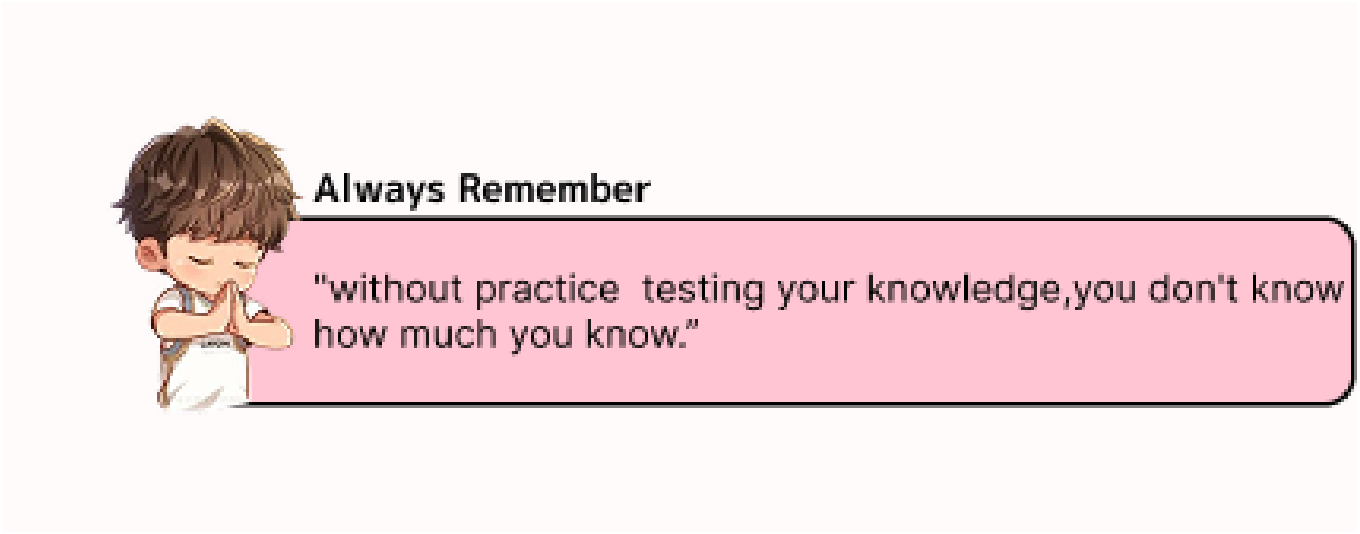
```
myset={1,2,[3,4]}
```

(Error) list not allowed

Correct Example:

```
myset={1,2,(3,4)}
```

tuple allowed



EE.iNfo

PYTHON in 21 Days

-Crack any coding interview-

Day-17 Coding Practice Sheet

Here are some easy coding questions based on the "Sets in Python" topic:

Given two Sets:

-- a={1,2,3,4}
-- b={3,4,5,6}

1.Print **“a”** values and then print **“b”** values.
2.Print **“a”** values using **for loop**.
3.Add **8** to set **“a”**.
4.Remove **4** from set **“b”**.
5.Find the common elements **(hint: Set Operations)**.
6.Combine both sets without duplicates.

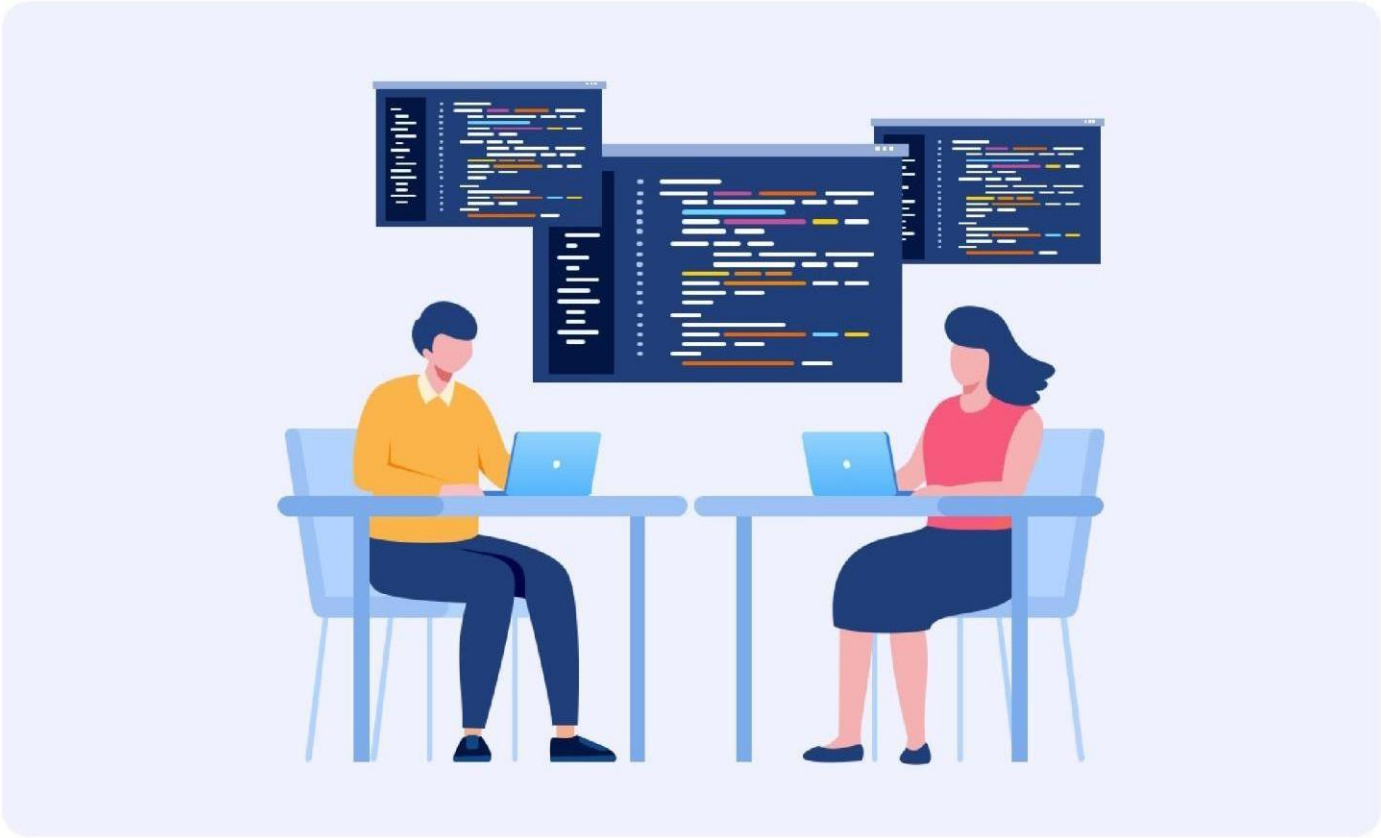
[Try here] [LINK]

EE.iNfo

PYTHON in 21 Days

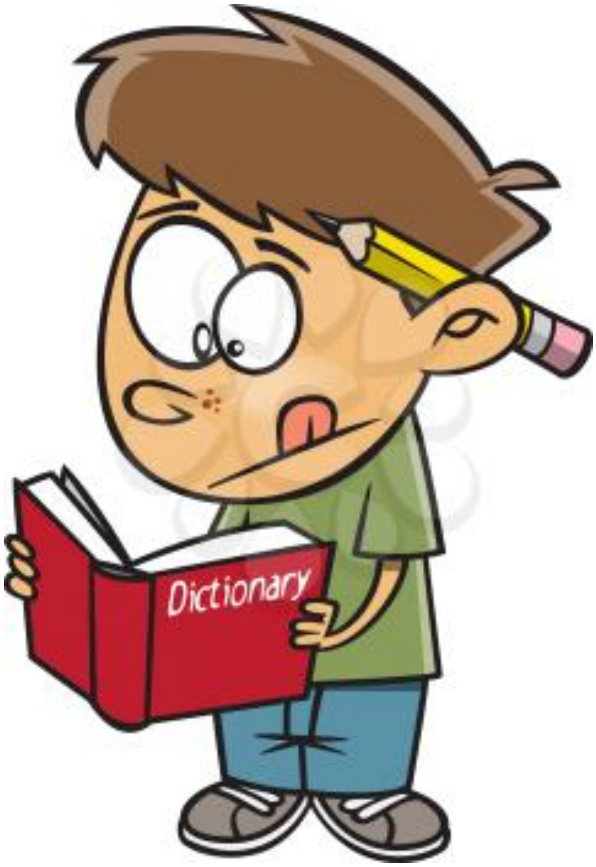
-Crack any coding interview-

DAY 18: EE.INFO



“Take a deep breath and analyze it” – Author

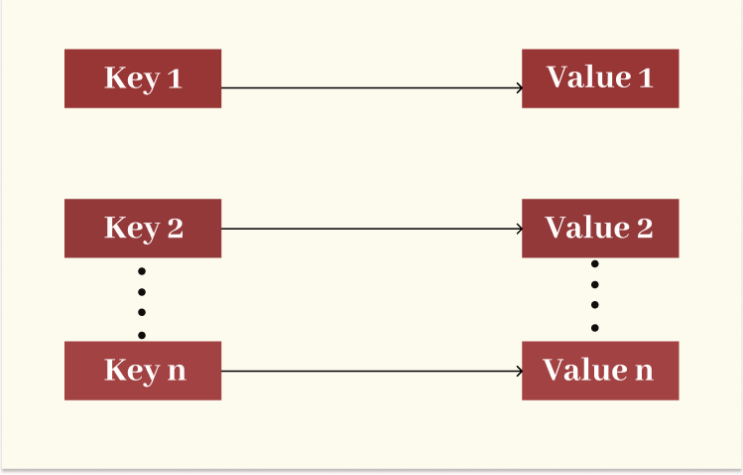
“DICTIONARIES IN PYTHON”



18. DICTIONARIES

ఒక Dictionaryలో contentని page no.తో access చేస్తాం. Similar గా PYTHONలో dictionaries ని వాడుతాం. ఇక్కడ **Values** and **Keys** ఉంటాయి.

--Dictionaries are **mutable**(changeable).



Dictionary Definition: It is an unordered, mutable collection used to store data in **key-value pairs** in "**{ }**".

Ex:

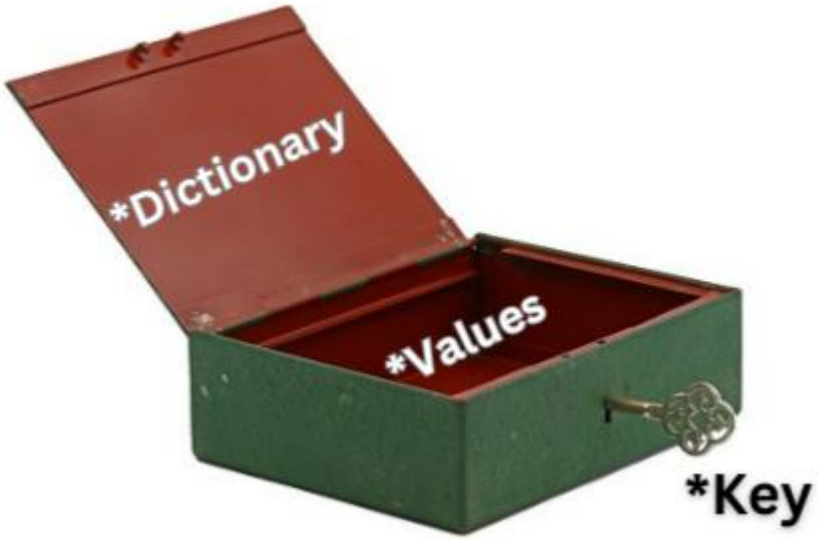
```
student={"name":"babu","age":21,"course":"python"}
print(student["name"])
print(student["age"])
print(student)
```

Output:

```
babu
21
{'name': 'babu', 'age': 21, 'course': 'python'}
```

Note:

-keys: "home, age, Course".
-value: babu, 21, Python
- mutuble.(?).



18.2. ACCESSING DICTIONARIES:

Case-1: Accessing by keys

Ex:

```
dict1={"name":"Stevejobs","company":"apple","age":30}
print(dict1["name"])
print(dict1["company"])
print(dict1["age"])
```

Output:

```
Stevejobs
apple
30
```

Case-2: Accessing only keys

Ex:

```
dict1={"name":"Stevejobs","company":"apple","age":30}
for key in dict1.keys():
  print(key)
```

Output:

```
name
company
age
```

Case-3: Accessing only Values

Ex:

```
dict1={"name":"Stevejobs","company":"apple","age":30}
for values in dict1.values():
  print(values)
```

Output:

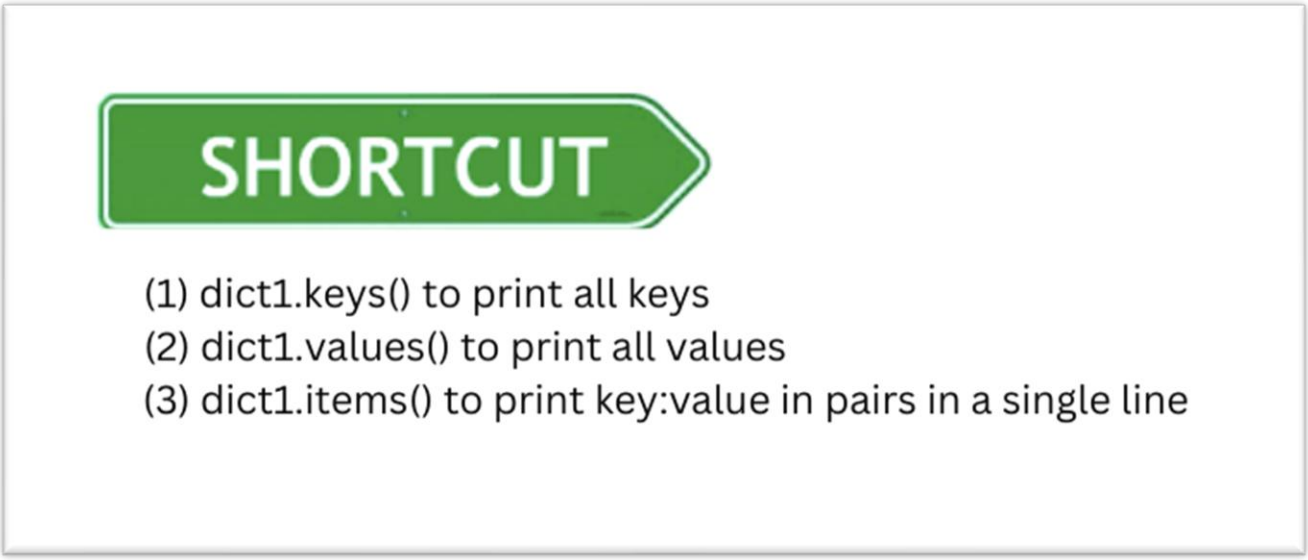
```
Stevejobs
apple
30
```

Case-4: Accessing both keys & values at a time

Ex:

```
dict1={"name":"Stevejobs","company":"apple","age":30}
for key,value in dict1.items():
  print(key,value)
```

Output:?



18.3 OPERATIONS ON DICTIONARIES:

1.MODIFYING VALUES:

EX:

```
dict2={"mobile":"521FG","year":"2021"}
dict2["year"]=2022
print(dict2)
```

Output:

```
{'mobile': '521FG', 'year': 2022}
```

2.ADDING A NEW KEY-VALUE PAIRS:

EX:

```
dict2={"mobile":"521FG","year":"2021"}
dict2["company"]="samsung"
print(dict2)
```

Output:

```
{'mobile': '521FG', 'year': '2021', 'company': 'samsung'}
```

3.REMOVING A KEY-VALUE PAIR:

(1) Using Pop:

EX:

```
dict2={"mobile":"521FG","year":"2021"}
dict2.pop("year")
print(dict2)
```

Output:

```
{'mobile': '521FG'}
```

(2) Using del keyword:

EX:

```
dict2={"mobile":"521FG","year":"2021"}
del dict2["mobile"]
print(dict2)
```

Output:

```
{'year': '2021'}
```

4.CHECKING IF A KEY EXISTS:

EX:

```
dict3={"device":"laptop","year":2020,"cost":40000}
if "cost" in dict3:
  print("cost exists")
```

Output:

```
cost exists
```

5.DICTIONARY LENGTH:

EX:

```
dict3={"device":"laptop","year":2020,"cost":40000}
print(len(dict3))
```

Output:

```
3
```

18.3 OPERATIONS ON DICTIONARIES:(IMP)

Note: No need to remember for coding POV

Method	Description
clear()	Removes all the elements from dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

EE.iNfo

PYTHON in 21 Days

-Crack any coding interview-

Day-19 Coding Practice Sheet

Here are some easy coding questions based on the "Dictionaries in Python" topic:

Let us make it simple yet effective:

Given Dictionaries:

```
car={
    “brand” : ”Toyota”
    “model” : “Corolla”
    “year”: 2020
    }
```

Tasks:

1.Print the car.model.
2.Change the year to 2025.
3.Add a new color:RED.
4.Delete the model key.

Can you do this now ? [Submit in assignment]

[Try here] [LINK]

EE Info-Python Cheat Sheet

Base Types

integer, float, boolean, string, bytes

```
int 783  0 -192  0b010 0o642 0xF3
          zero   binary  octal  hexa
float 9.23 0.0  -1.7e-6
                ×10^-6
bool True False
str "One\nTwo"        Multiline string:
    escaped new line  """X\tY\tZ
    'I\'m'            1\t2\t3"""
    escaped '         escaped tab
bytes b"toto\xfe\775"
      hexadecimal  octal
```

immutables

Container Types

- **ordered sequences**, fast index access, repeatable values

```
list [1,5,9]   ["x",11,8.9]   ["mot"]   []
tuple (1,5,9)  11,"y",7.4     ("mot",)  ()
```

Non modifiable values (immutables) — *expression with only comas →* `tuple`

```
str bytes  (ordered sequences of chars / bytes)   ""  b""
```

- **key containers**, no *a priori* order, fast key access, each key is unique

```
dictionary  dict {"key":"value"}   dict(a=3,b=4,k="v")   {}
(key/value associations)  {1:"one",3:"three",2:"two",3.14:"π"}
collection  set {"key1","key2"}    {1,9,3,0}            set()
```

keys=hashable values (base types, immutables…) `frozenset` *immutable set* *empty*

Identifiers

for variables, functions, modules, classes… names

`a…zA…Z_` followed by `a…zA…Z_0…9`

- diacritics allowed but should be avoided
- language keywords forbidden
- lower/UPPER case discrimination

☺ `a toto x7 y_max BigOne`
☹ ~~`8y and for`~~

Variables assignment

`=`

assignment ⇔ **binding** of a *name* with a *value*

1) *evaluation of right side expression value*
2) *assignment in order with left side names*

```
x=1.2+8+sin(y)
a=b=c=0          assignment to same value
y,z,r=9.2,-7.6,0 multiple assignments
a,b=b,a          values swap
a,*b=seq         unpacking of sequence in
*a,b=seq         item and list
x+=3   increment ⇔ x=x+3      and *= /= %= …
x-=2   decrement ⇔ x=x-2
x=None « undefined » constant value
del x  remove name x
```

Conversions

`type(`*expression*`)`

```
int("15") → 15
int("3f",16) → 63        can specify integer number base in 2nd parameter
int(15.56) → 15          truncate decimal part
float("-11.24e8") → -1124000000.0
round(15.56,1)→15.6      rounding to 1 decimal (0 decimal → integer number)
bool(x)  False for null x, empty container x , None or False x ; True for other x
str(x)→"…"               representation string of x for display (cf. formatting on the back)
chr(64)→'@'  ord('@')→64   code ↔ char
repr(x)→"…"              literal representation string of x
bytes([72,9,64]) → b'H\t@'
list("abc") → ['a','b','c']
dict([(3,"three"),(1,"one")]) → {1:'one',3:'three'}
set(["one","two"]) → {'one','two'}
```

separator `str` and *sequence of* `str` → *assembled* `str`

```
':'.join(['toto','12','pswd']) → 'toto:12:pswd'
```

`str` *splitted on whitespaces* → `list` of `str`

```
"words with  spaces".split() → ['words','with','spaces']
```

`str` *splitted on separator* `str` → `list` *of* `str`

```
"1,4,8,2".split(",") → ['1','4','8','2']
```

sequence of one type → `list` of another type *(via list comprehension)*

```
[int(x) for x in ('1','29','-3')] → [1,29,-3]
```

Sequence Containers Indexing

for lists, tuples, strings, bytes…

```
negative index   -5  -4  -3  -2  -1
positive index    0   1   2   3   4
lst=[10, 20, 30, 40, 50]
positive slice    0   1   2   3   4   5
negative slice   -5  -4  -3  -2  -1
```

Items count

`len(lst)→5`

index from 0
(here from 0 to 4)

Individual access to **items** via `lst[`*index*`]`

```
lst[0]→10   ⇒ first one      lst[1]→20
lst[-1]→50  ⇒ last one       lst[-2]→40
```

On mutable sequences (`list`), remove with `del lst[3]` *and modify with assignment* `lst[4]=25`

Access to **sub-sequences** via `lst[`*start slice*`:`*end slice*`:`*step*`]`

```
lst[:-1]→[10,20,30,40]   lst[::-1]→[50,40,30,20,10]   lst[1:3]→[20,30]   lst[:3]→[10,20,30]
lst[1:-1]→[20,30,40]     lst[::-2]→[50,30,10]         lst[-3:-1]→[30,40]  lst[3:]→[40,50]
lst[::2]→[10,30,50]      lst[:]→[10,20,30,40,50]  shallow copy of sequence
```

Missing slice indication → from start / up to end.
On mutable sequences (`list`), remove with `del lst[3:5]` *and modify with assignment* `lst[1:4]=[15,25]`

Boolean Logic

Comparisons : `< > <= >= == !=` (*boolean results*) — ≤ ≥ = ≠

`a and b` logical and — *both simulta-neously*

`a or b` logical or — *one or other or both*

pitfall : `and` *and* `or` *return* *value* *of* `a` *or of* `b` *(under shortcut evaluation).*
⇒ *ensure that* `a` *and* `b` *are booleans.*

`not a` logical not

`True` `False` — True and False constants

Statements Blocks

parent statement:
  statement block 1…
parent statement:
  statement block2…
next statement after block 1

indentation !

configure editor to insert 4 spaces in place of an indentation tab.

Modules/Names Imports

module `truc` ⇔ *file* `truc.py`

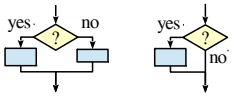
```
from monmod import nom1,nom2 as fct
                →direct access to names, renaming with as
import monmod  →access via monmod.nom1 …
```

modules and packages searched in *python path* (cf `sys.path`)

Conditional Statement

statement block executed only ***if*** *a condition is true*

`if` *logical condition*`:`
  → *statements block*



Can go with several *elif*, *elif*... and only one final *else*. Only the block of first true condition is executed.

```
if age<=18:
    state="Kid"
elif age>65:
    state="Retired"
else:
    state="Active"
```

with a var `x`:
`if bool(x)==True:` ⇔ `if x:`
`if bool(x)==False:` ⇔ `if not x:`

Maths

floating numbers… approximated values

Operators: `+ - * / // % **`
Priority `(…)` — × ÷ ↑ ↑ a^b — integer ÷ ÷ remainder

`@` → matrix × *python3.5+* `numpy`

```
(1+5.3)*2→12.6
abs(-3.2)→3.2
round(3.57,1)→3.6
pow(4,3)→64.0
```

usual order of operations

angles in radians

```
from math import sin,pi…
sin(pi/4)→0.707…
cos(2*pi/3)→-0.4999…
sqrt(81)→9.0      √
log(e**2)→2.0
ceil(12.5)→13
floor(12.5)→12
```

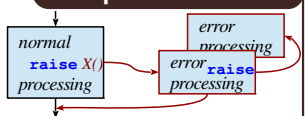
modules `math`, `statistics`, `random`, `decimal`, `fractions`, `numpy`, *etc. (cf. doc)*

Exceptions on Errors

Signaling an error:
`raise` *ExcClass(…)*

Errors processing:

```
try:
    → normal procesising block
except Exception as e:
    → error processing block
```



`finally` *block for final processing in all cases.*

Conditional Loop Statement

*statements block executed **as long as** condition is true*

```
while logical condition:
    statements block
```


```
s = 0          # initializations before the loop
i = 1          # condition with a least one variable value (here i)
while i <= 100:
    s = s + i**2
    i = i + 1      # make condition variable change !
print("sum:",s)
```

Algo: $s = \sum_{i=1}^{i=100} i^2$

beware of infinite loops!

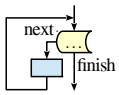
Loop Control

`break` *immediate exit*
`continue` *next iteration*
`else` *block for **normal** loop exit.*

Iterative Loop Statement

*statements block executed **for each** item of a container or iterator*

```
for var in sequence:
    statements block
```



Go over sequence's **values**

```
s = "Some text"   # initializations before the loop
cnt = 0
for c in s:       # loop variable, assignment managed by for statement
    if c == "e":
        cnt = cnt + 1
print("found",cnt,"'e'")
```

Algo: count number of `e` *in the string.*

loop on dict/set ⇔ loop on keys sequences
use *slices* to loop on a subset of a sequence

Go over sequence's **index**

- modify item at index
- access items around index (before / after)

```
lst = [11,18,9,12,23,4,17]
lost = []
for idx in range(len(lst)):
    val = lst[idx]
    if val > 15:
        lost.append(val)
        lst[idx] = 15
print("modif:",lst,"-lost:",lost)
```

Algo: limit values greater than 15, memorizing of lost values.

good habit : don't modify loop variable

Go simultaneously over sequence's **index** and **values**:

```
for idx,val in enumerate(lst):
```

Display

```
print("v=",3,"cm :",x,",",y+4)
```

items to display : literal values, variables, expressions

`print` options:

- `sep=" "` items separator, default space
- `end="\n"` end of print, default new line
- `file=sys.stdout` print to file, default standard output

Input

```
s = input("Instructions:")
```

`input` always returns a **string**, convert it to required type (cf. boxed *Conversions* on the other side).

Generic Operations on Containers

`len(c)` → items count
`min(c)` `max(c)` `sum(c)`
`sorted(c)` → `list` sorted *copy*
`val in c` → boolean, membership operator `in` (absence `not in`)
`enumerate(c)` → *iterator* on (index, value)
`zip(c1,c2…)` → *iterator* on tuples containing c_i items at same index
`all(c)` → `True` if **all** `c` items evaluated to true, else `False`
`any(c)` → `True` if **at least one** item of `c` evaluated true, else `False`

*Note: For dictionaries and sets, these operations use **keys**.*

*Specific to **ordered sequences containers** (lists, tuples, strings, bytes…)*

`reversed(c)` → *inversed iterator* `c*5`→ duplicate `c+c2`→ concatenate
`c.index(val)` → *position* `c.count(val)` → *events count*

```
import copy
copy.copy(c) → shallow copy of container
copy.deepcopy(c) → deep copy of container
```

Operations on Lists

modify original list

`lst.append(val)`	add item at end
`lst.extend(seq)`	add sequence of items at end
`lst.insert(idx,val)`	insert item at index
`lst.remove(val)`	remove first item with value *val*
`lst.pop([idx])` →*value*	remove & return item at index *idx* (default last)
`lst.sort()` `lst.reverse()`	sort / reverse liste *in place*

Operations on Dictionaries

```
d[key]=value        d.clear()
d[key] → value      del d[key]
d.update(d2)   update/add associations
d.keys()
d.values()     →iterable views on keys/values/associations
d.items()
d.pop(key[,default]) → value
d.popitem() → (key,value)
d.get(key[,default]) → value
d.setdefault(key[,default]) →value
```

Operations on Sets

Operators:
`|` → union (vertical bar char)
`&` → intersection
`-` `^` → difference/symmetric diff.
`<` `<=` `>` `>=` → inclusion relations
Operators also exist as methods.

```
s.update(s2)  s.copy()
s.add(key)  s.remove(key)
s.discard(key)  s.clear()
s.pop()
```

Files

storing data on disk, and reading it back

```
f = open("file.txt","w",encoding="utf8")
```

- file **variable** for operations
- **name** of file on disk (+path…) — cf. modules `os`, `os.path` and `pathlib`
- opening **mode**: `'r'` read, `'w'` write, `'a'` append, …`'+'` `'x'` `'b'` `'t'`
- **encoding** of chars for *text files*: `utf8` `ascii` `latin1` …

writing

```
f.write("coucou")
f.writelines(list of lines)
```

reading

read empty string if end of file

```
f.read([n])        → next chars
    if n not specified, read up to end !
f.readlines([n])   → list of next lines
f.readline()       → next line
```

*text mode `t` by default (read/write `str`), possible binary mode `b` (read/write `bytes`). **Convert from/to required type** !*

`f.close()` dont forget to **close the file** after use !

`f.flush()` write cache
`f.truncate([size])` resize

reading/writing progress sequentially in the file, modifiable with:

`f.tell()` →*position*
`f.seek(position[,origin])`

Very common: opening with a guarded block (automatic closing) and reading loop on lines of a text file:

```
with open(…) as f:
    for line in f :
        # processing of line
```

Integer Sequences

`range([start,] end [,step])`

start default 0, *end* not included in sequence, *step* signed, default 1

`range(5)` → 0 1 2 3 4
`range(3,8)` → 3 4 5 6 7
`range(2,12,3)` → 2 5 8 11
`range(20,5,-5)` → 20 15 10
`range(len(seq))` → *sequence of index of values in seq*

range provides an immutable sequence of int constructed as needed

Function Definition

function name (identifier)
named parameters

```
def fct(x,y,z):
    """documentation"""
    # statements block, res computation, etc.
    return res
```

`fct`

result value of the call, if no computed result to return: `return None`

parameters and all variables of this block exist only *in* the block and *during* the function call (think of a "black box")

Advanced: `def fct(x,y,z,*args,a=3,b=5,**kwargs):`
**args variable positional arguments (→`tuple`), default values,*
***kwargs variable named arguments (→`dict`)*

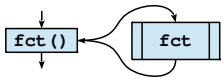
Function Call

```
r = fct(3,i+2,2*i)
```

storage/use of returned value — *one argument per parameter*

this is the use of function name *with parentheses* which does the call

*Advanced: *sequence **dict*



Operations on Strings

```
s.startswith(prefix[,start[,end]])   s.strip([chars])
s.endswith(suffix[,start[,end]])
s.count(sub[,start[,end]])   s.partition(sep) → (before,sep,after)
s.index(sub[,start[,end]])   s.find(sub[,start[,end]])
s.is…()  tests on chars categories (ex. s.isalpha())
s.upper()  s.lower()  s.title()  s.swapcase()
s.casefold()  s.capitalize()  s.center([width,fill])
s.ljust([width,fill])  s.rjust([width,fill])  s.zfill([width])
s.encode(encoding)  s.split([sep])  s.join(seq)
```

Formatting

formating directives — values to format

```
"modele{} {} {}".format(x,y,r) → str
"{selection:formatting!conversion}"
```

- **Selection** :
  `2`
  `nom`
  `0.nom`
  `4[key]`
  `0[2]`

Examples:

```
"{:+2.3f}".format(45.72793)
→'+45.728'
"{1:>10s}".format(8,"toto")
→'      toto'
"{x!r}".format(x="I'm")
→'"I\'m"'
```

- **Formatting** :

fill char *alignment* *sign* *mini width*.*precision~maxwidth* *type*

`< > ^ =` `+ - space` `0` at start for filling with 0

integer: `b` binary, `c` char, `d` decimal (default), `o` octal, `x` or `X` hexa…
float: `e` or `E` exponential, `f` or `F` fixed point, `g` or `G` appropriate (default),
string: `s` … `%` percent

- **Conversion** : `s` (readable text) or `r` (literal representation)

Note that this cheat sheet is to remember or revise the concepts quickly after the whole course completion, you don't understand anything if you directly try to learn from this cheat sheet only. -Author.

1. 3 - 20 LPA Coding Sheet :

[CLICK HERE]

STEPS:

- click above sheet link
- select your target salary
- Try to solve problems on your own for first 20 mins.
- If you unable to solve it is ok then

Go to Youtube search “problem name + Striver or babar”

- Dont just see, understand the approach always think why this? why not that? then only you can solve other problems on your own.

“That’s It you are ready to crack any high paying job”

1.**Most Asked Python Interview Questions :** [CLICK HERE]

2. 71 Fresher Must have Projects. [CLICK HERE]

3. Final Year 700+ Advanced, AI ML Projects. [CLICK HERE]

* Note :

➢ There is no rocket science in copying existing projects.

➢ They are all hosted in github.

➢ Hack : seacrh “project name (optional) + tech stack + github.” on gooogle.

➢ Ex : Beginner AI Projects Python github.

This hack is very useful because different companies comes with different tech stack projects expectations in our resume like some ask ai projects, some ask mern projects, some ask java projects and so on.

➢ Projects categories : Beginner, Intermediate, adanced. If you really passionate solve the problems around you using softwares like files sharing app for you class, these makes you standout for interviews.

"Good artists copy, great artists steal"