Extra

PYTHON

- high level programming language
- designed by Guido Van Rossum
- portable (works on linux / windows / mac)

Module - file containing code written by somebody else which can be imported and used in our programs.

PIP - package manager for python

types of modules: 1) built in modules - pre installed
2) external modules - need to install

Comments - used to write something which the programmer doesn't want to execute
1. single line comments → written using (#)
2. multi line comments → written using """ comment """

VARIABLES -
used to store values to a memory location in program.

KEYWORDS -
reserved words in python

IDENTIFIRS -
class / function / variable name

DATA Types
1. Integers
2. floating point numbers
3. strings
4. boolean
5. none (null / have no value)

Date - 27/02/23

Code
```
print("Hello world")
```

Output
```
Hello world
```

Reserved keywords - These cannot and should not be used as Identifiers/ variables Names.
The moment you will type a reserved word. It will turn green.
This will be an indication that do not use it for naming a variable.
Eg AND, DEL, For , Is ,FROM

eg Code
```
pass = 8
```

Output
```
pass = 8
     ^
SyntaxError: invalid syntax
```

eg Code
```
print = 8
while = 8
```
green in Colour

Output
```
while = 8
      ^
SyntaxError: invalid syntax
```

Above examples are can not part
should not part
Python is Case sensitive
Everything while is Native to python or intrinsic to it will always be in lower case. All the reserved keywords are by default in lower case
If we change the case of these reserved keywords, the system will allow us to use them as identifiers and variables name. BUT BUT BUT 'NEVER EVER USE THEM'.

eg Code
```
While = 6
Print = 7
Pass = While + Print
print(Pass)
```

Output
```
13
```

How to name the identifiers
What is an identifier
Identifiers is a name give to a variable, function or any object.
We use identifiers to refer or identify objects in the python program.
This means that there are some rules to name an identifiers.

They should not be anything from the Reserved keywords.
Ex – and, print, or, del.

1.) In case of reserved keywords, the case is very important
Ex – Del will not work whereas del will work but again stay away from these words.

2) Your identifier should not be starting from a numerical
Ex. 1raghav, 56Coolboy.

a) What is then allowed? You can name them as Raghav11 instead, Coolboy56.

3) We can not use any special char in our identifiers. Underscore is an exception.
Ex– !@#$%^&*().

4) Try to avoid same names for 2 variables in the same code. If very urgent then use will use.

5) Don't take shortcuts while naming your identifiers/ variables) Because the code will be used by many people and they should not be able to comprehend the code easily.

6) Identifiers can be of any length (79 was earlier). But now NO LIMIT.

Eg
ⓐ Variable for Regression analysis outputted by the Algorithm is represented here = 67.9865
OR
ⓑ X = 67.9865
ⓒ Depends on the work you are doing and with whom call the shots accordingly.

7. WHITESPACES are not allowed means you cannot put space in between while naming your identifier.

8. All the identifiers are CASE SENSITIVE
9. You are not allowed to use all the digits while naming naming your identifier.
   Ex - 7869
10. SPECIAL CASE ALERT - Underscore can precede the variable name.
   Ex _Variable = 789 , Again not advised to use
11. Please try to make sense out of your variable name or identifier.

Date - 28/02/23

Variables are used to store the values of any data type.
Variables need to be declared and then we can assign an value to them

eg Code

```
Radius = 5          # Variable 1
Pi = 3.14           # Variable 2
area = Radius * Radius * pi   # Variable 3
print(area)
```

RN behaving like a constant constant

Output 78.5

#eg2

Code

```
F_Name = "Raghav"
L_Name = "Goel"
print("Hello", F_Name, L_Name)
```

Output Hello Raghav Goel

#eg3 # Suppose we have 3 values - Raghav, Goel, 70

Code

```
F_Name = "Raghhhhav"
L_Name = "Goeeeeel"
Weight = 70
print(F_Name, L_Name, Weight)
```

Output Raghhhhav Goeeeeel 70

#eg 4

Code

```
a,b,c = 56,67,78
print(a,b,c)
```

Output 56 67 78

#eg5

Code

```
f,g,h = 5,6
```

Output

```
----1 f,g,h = 5,6
```

ValueError: not enough values to unpack (expected 3, got 2)

Data Types in Python

1. TEXT TYPE - TEXT, ALPHANUMERIC, STRINGS (STR)
2. NUMERIC TYPE - THESE HAVE TO DEAL WITH ONLY AND ONLY NUMBERS
   A. INTEGER (INT)
   B. FLOAT (FLOAT)
   C. COMPLEX (COMPLEX NUMBERS)
3. BOOLEAN TYPE - 0 OR 1 / TRUE OR FALSE (BOOL) / YES OR NO
4. DERIVED/SEQUENCE DATA TYPE :-
   A. LISTS
   B. TUPLES
   C. RANGE
5. SETS AND DICTIONARY

Qn. WAP - Write a program to describe yours in 5 line. Each line has different variable.

Strings -> They are the default data types for alphabets and alpha numeric.

How do we define a Strings ??
Syntax of list
" " - Double Quotes OR
' ' - Single Quotes

eg 1

Code
```
str1 = "Raghav"
type(str1) # SYNTAX for TYPE - type(variable name)
```

Output
```
str
```

Code
```
str1 = "Raghav"
type(str1)
print(str1)
```

Output
```
Raghav
```

Single quotes

eg2

Code
```
str2 = 'Groel'
type(str2)
```

Output
```
str
```

Code
```
str2 = 'Groel'
type(str2)
print(str2)
```

Output
```
Groel
```

LENGTH FUNCTION which is in built function — len()

Code
```
F_Name = "Raghav"
len(F_Name)
```

Output
```
6
```

Code
```
g = "My Name Is Raghav Groel"
len(g)
```

Output
```
22
```

Code
```
h = 'Python Course'
len(h)  # because whitespace is also counted as char
```

Output
```
13
```

INDEX in python and we have some indexable datatypes
STRING is a Indexable data type.
Indexable - data types which have a index which is like a path to # transverse data type.
Inder in python allows you to slice and dice the data type.
Imagine you have a knife with you and you wish to cut or slice the data type.

eg-1

SYNTAX to slice and dice — variablename[index or position]

Code
```
str1 = "My Name"
len(str1)
```

Output
```
7
```

Code

```
str1 = "My Name"
len(str1)
str1[1]
```

Output

IndexError: string index out of range

The above is happening because, in python the index starts from 0 and not 1.
The count starts from zero.

Code

```
str1 = "My Name"
str1[6]
```

Output

'e'

Code

```
str1 = "My Name"
str1[0]
```

Output

'M'

Ex-3

```
str3 = 'Today is Tuesday'
```

Check the Type of str3
Check the Length
Fetch 'T' from Tuesday
Fetch 'y' from Today
Fetch 'is'

Code

```
type(str3)
len(str3)
str3[9]
str3[4]
str3[6:8]
```

Output

```
str
16
'T'
'y'
'is'
```

The above is a concept of colon where the value before the colon
signifies the starting point of the slicing and value post the colon.
# indicates the end point of slicing

Code

```
str3 = 'Today is Tuesday'
str3[6:7]
```

Output

'i'

In above the answer should have been 'is' but the output is 'i'
This happened because with value after colon will be considered as 'value - 1'
SYNTAX becomes :- Variable_name[start : End+1]

Code
```
str3 = 'Today is Tuesday'
str3[6:8]
```

Output
is

or 'ay is T'

Code
```
str3 = 'Today is Tuesday'
str3[3:10]
```

Output 'ay is T'

Joining the strings — Concatenation of string

Code
```
f = "Hello"
g = "World"
f+g
```

Output 'HelloWorld'

Multiplication of strings — Sounds weird right ?!

Code
```
f = 'Hello'
f*3
```

Output 'HelloHelloHello'

Multiplication of strings refers to the repetition of the string.
We also saw \n

Code
```
x = "My\n Name\n is\n Raghav\n Goel"
print(x)
```

Output
My
Name
is
Raghav
Goel

\t — TAB

Code
```
t = "My\t Name\t is\t Raghav\t Goel"
print(t)
```
Output: My Name is Raghav Goel

So for system all Raghav's will be different

```
a1 = "Raghav Goel"
a2 = "RAGHAV Goel"
a3 = "raghav Goel"
a4 = "Rag HaV Goel"
a5 = "RAGHaV Goel"
```

Lets evanise them to lower case — Lower() is a method

```
a1 = a1.lower()
a2 = a2.lower()
a3 = a3.lower()
a4 = a4.lower()
a5 = a5.lower()
```

a1 becomes 'raghav goel' | a2 'raghav goel' | a3 'raghav goel' | a4 'raghav goel' | a5 'raghav goel'

Code
```
a1 = "Raghav Goel"
a2 = "RAGHAV Goel"
a3 = "raghav Goel"
a4 = "Rag HaV Goel"
a5 = "RAGHaV Goel"
a1 = a1.lower()
a2 = a2.lower()
a3 = a3.lower()
a4 = a4.lower()
a5 = a5.lower()
print(a1)
print(a2)
print(a3)
print(a4)
print(a5)
```

Output
```
raghav goel
raghav goel
raghav goel
raghav goel
raghav goel
```

```
a5
a5.upper()
```
O/P GAGHAV GOEL

Code
```
a5 = "RAGHaV Goel"
a5 = a5.upper()
print(a5)
```
Output: RAGHAV GOEL

Date- 6/03/23

Code

```
fn = "Raghav"
ln = "Goel"
print("{0}{1}".format(ln, fn))
```

Output GoelRaghav

We are formatting the print statement
Till now, we have been taking the static inputs as a string
Now we will take user's input

Input

```
a = input("What language are you working on ?.")
# Inside input we write the prompt which is given to the user
```

Output

```
print("We are working on", a)
```

Op We are working on python

Input – Output

Code

```
a = input("What language are you working on ?.")
print("We are working on", a)
```

Output What language are you working on ? python language
We are working on python language

Blank value input given

Code

```
F_name = input("Please enter your first name")
L_name = input("Please enter your last name")
print("My name is {0}{1}".format(L_name, F_name))
```

Output Please enter your first name Raghav
Please enter your last name Goel
My name is GoelRaghav

{0} {1} → represents line number

Code

Sep – used for separation

```
F_name = input("Please enter your first name")
L_name = input("Please enter your last name")
City = input("Please enter your City")
Grender = input("Please enter your Grender (M/F)")
print(F_name, L_name, City, Grender, sep = '|')
```

Output

```
Please enter your first name Raghav
Please enter your last name Goel
Please enter your City Delhi
Please enter your Grender (M/F) M
Raghav|Goel|Delhi|M
```

Code

```
F_name = input("Please enter your first name")
L_name = input("Please enter your last name")
City = input("Please enter your City")
Grender = input("Please enter your Grender (M/F)")
print(F_name, L_name, city, Grender, sep = '-->')
```

Output

```
Please enter your first name Raghav
Please enter your last name Goel
Please enter your City Delhi
Please enter your Grender (M/F) M
Raghav --> Goel --> Delhi --> M
```

Integers : Any numberic value without having a fraction or decimal point is an integer

eg. – 1, 4, 6, –8, –9

Code

```
Integer = 2
print(type(Integer))
```

Output

```
<class 'int'>
```

Float : Any number which has a fraction or Decimal part will be float

Ex : 2.5, 5.78653456

Code

```
Float = 5.76544
print(type(Float))
```

Output

```
<class 'float'>
```

Code
```
a = 7
b = 3
c = a/b
print(c)
```

Output 2.333333333333335

Use of round method — round (expression, rounding value)

Code
```
a = 7
b = 3
c = round ((a/b), 3)
print(c)
```

Output 2.333

Complex Numbers — a+bi, where i is used to be an 'imaginary no.'
In python, the imaginary part is "j" and not 'i'.
ex- a+bj

Code
```
a = 1 +2j
b = 4j
c = 1 - 2j
d = -6j
print (type(a))
print (type(a))
print (type(a))
print (type(a))
```

~~Compiler~~ Output
```
< class 'complex' >
< class 'complex' >
< class 'complex' >
< class 'complex' >
```

Concept of Type Conversion
What do you mean by type conversion?
We try to move from one type of data to another by converting it in desirable data type

Code
```
no = 2
print (type(no))
```

Output < class int >

Now convert this integer into float and complex

Converting into float using an extra variable

Code

```
no = 2
nof = float(no)
print(type(nof), nof)
```

Output `<class 'float'> 2.0`

Using the same variable

Code

```
no1 = 2
no1 = float(no1)
print(type(no1), no1)
```

Output `<class 'float'> 2.0`

Converting into complex

Code

```
no1 = 2
no1 = complex(no1)
print(type(no1), no1)
```

Output `<class 'complex'> (2+0j)`

Can we convert the float into integer

```
cmp = 1+2j
cmp = int(cmp)
print(type(cmp), cmp)
```

Output

TypeError: int() argument must be a string, a bytes-like object or a real number, not 'complex'

Code

```
No_float = 2.4567
print(type(No_float), No_float)
```

Output `<class 'float'> 2.4567`

Code

```
No_float = 2.4567
No_float = int(No_float)
print(type(No_float), No_float)
```

Output < class 'int' > 2

When we convert the float into, it simply removes the decimal / fraction part.

NOTE → we can not convert complex no into any other format

We can convert string into complex ?

Code

```
a = "Raghav"
a = int(a)
```

Output ValueError : invalid literal for int() with base 10: 'Raghav'

```
a = "10" # If we have any number ie Int, Float, complex
           inside string, it will allow
         # to typecast otherwise not
```

Code

```
a = "10"
print(type(a), a)
```

Output < class 'str' > 10

Code

```
a = "10"
a = int(a)
print(type(a), a)
```

Output <class 'int'> 10

Plese try to convert a Number data type into a string data type

```
m = 201059087 # m is made as combination\ Height in cm\
                weight \ last 2 digits of pincode
```

Code

```
m = 201059087
print(type(m), m)
```

output <class 'int'> 201059087

Code

```
m = 20105 9087
m = str(m)
print (type(m), m)
```

Output <class 'str'> 20105 9087

Now m is a string
We know already we can slice and dice the string

```
# Age = 20
print (m[0:2])
```

```
# Height = 105
print (m[2:5]
```

```
# Weight = 90
print (m[5:7])
```

```
# Pin_code = 87
print (m[7:])
```

Code

```
m = 20105 9087
m = str(m)
print("Age = ", m[0:2])
print("Height = ", m[2:5])
print("Weight = ", m[5:7])
print("Pin_code = ", m[7:])
```

Output

```
Age = 20
Height = 105
Weight = 90
Pin_code = 87
```

Date — 06/03/23

Now in our knife, we have one more functionality which is of jump
what is Jump/skip? It is the third argument.
which can be passed in this function

Syntax Variable_Name[start:End+1:Jump/Skip]
ex- m[0:4:2]

Code
```
m = 20105087
m = str(m)
print(m[0:4:1])
```
Output 2010

Code
```
m = 20105087
m = str(m)
print(m[0:4:2])
```
Output 21

I want you tell me what will be the output:
```
print(m)
print(m[0:9:3])
```

Code
```
m = 20105087
m = str(m)
print(m)
print(m[0:9:3])
```
Output
```
20105087
200
```

Count of char in a string
Syntax → Variable_Name.count('char_Name')

Code
```
string = 'Happy Holi Everyone'
len(string)
```
Output 19

# Count of 'h'
string.count('h')

Code
```
string = 'Happy Holi Everyone'
string.count('h')
```
Output 0

Code
```
string = 'Happy Holi Everyone'
string.count('H')
```

Output 2

Code
```
string = 'Happy Holi Everyone'
string.count(' ')
```

Output 2

WAP to ~~ask~~ users for their F_Name, L_Name, City, Age
and then display them.

Code
```
a = input("Enter F_name")
b = input("Enter L_name")
c = input("Enter city")
d = input("Enter age")
print("F_Name is ", a)
print("L_Name is ", b)
print("City is ", c)
print("Age is ", d)
```

Output
```
Enter F_name Abhishek
Enter L_name Kumar
Enter age city Ranchi
Enter age 19
F_Name is Abhishek
L_Name is Kumar
City is Ranchi
Age is 19
```

WAP to find the area of a circle
Radius will be entered by the user

```
# R = input("Enter the radius") # Default data type of input is str
# You have to typecast it to int/float
# in order to perform arithmetic calculation
```

Code
```
R = int(input("Enter the radius :"))
A = 22/7*R*R
print("The Area of the circle with radius", R, "is", A)
```

Output
```
Enter the radius : 7
The Area of the circle with radius 7 is 154.0
```

OR

Code
```
R = float(input("Enter the radius :"))
A = round(22/7*R*R, 2)
print("The Area of the circle with radius", R, "is", A)
```

Output
```
Enter the radius : 2.5
The Area of the circle with radius 2.5 is 19.64
```

#WAP to accept an integer(n) – single digit from the user
Make your program calculate the value [n+nn+nnn]
Eg – 4 ⇒ 4+44+444 = 492

By me

Code
```
n = int(input("Enter single digit No :"))
print(n+(10*n+n)+(100*n+10*n+n))
```

Output
```
Enter single digit No: 4
492
```

By Sir

```
n = int(input("Enter the single digit No. :"))
```

#We know there is a property of strings that if we multiply strings
This will repeat themselves, we will therefore typecast integers to strings

```
n = str(n)
```

For nn
```
nn = n*2
```

For nnn
```
nnn = n*3
output = print(int(n) + int(nn) + int(nnn))
```

Code
```
n = int(input("Enter the single digit No. :"))
n = str(n)
nn = n*2
nnn = n*3
print(int(n) + int(nn) + int(nnn))
```

Output
```
Enter single digit No. : 4
492
```

The moment we will move from 1 digit to 2 digits, this approach will fail.

WAP to enter 5 digit number and find the sum of all digits
Eg - 12345 , Sum → 15

Code
```
n = int(input("Please enter the 5 digit number"))
n = str(n)
a = int(n[0])
b = int(n[1])
c = int(n[2])
d = int(n[3])
e = int(n[4])
sum = a+b+c+d+e
print("The Sum of the digits is :", sum)
```

Output
```
Please enter the 5 digit number12345
The Sum of the digits is: 15
```

Boolean Data Type
Boolean means binary, Binary means Either/or, only 2 options
Boolean = True / False

Code
```
a = 3
b = 5
print(a==b)
```

Output
```
False
```

Code
```
a = 5
b = 5
print(a==b)
```

Output
```
True
```

Code
```
print(4>7)
```

Output
```
False
```

Code
```
bool(6)
```

output
```
True
```

For most cases where there is some degree or correctness or there is not false
thing, answer will be true

Code
```
bool(0)
```

Output
```
False
```

OPERATORS IN PYTHON

Operators are unused in python to perform certain operations
One values, variables, constant etc
In python we have certain categories of operators with us

1. ARITHMETIC OPERATORS
2. ASSIGMENT OPERATORS
3. COMPARISON OPERATORS
4. LOGICAL OPERATORS
5. IDENTITY OPERATORS
6. MEMBERSHIP OPERATORS
7. BITWISE OPERATORS

Arithmetic Operators
They are used to carry out the operation of numerical values
Basic Maths operation.
These will be addition, Sub, Div, floor div, Mul, Power, modulus

Date - 13/03/23

```
a = int(input("Enter the first number"))
b = int(input("Enter the second number"))
print(a+b) # Addition
print(a-b) # subtraction
print(a*b) # multiplication
print(a/b) # 'a' will be divided by 'b'
print(a//b) # Floor division, you will only get the whole number
              of the division and not the decimal part.
print(a**b) # Exponential - 'a will be raised to power b'
print(a%b) # this will give you the remainder of
             "a when divided by b"
```

Date - 13/03/23

Code

```
a = int (input("Enter the first number!"))
b = int (input("Enter the second number!"))
print(a+b)
print(a-b)
print(a*b)
print(a/b)
print(a//b)
print(a**b)
print(a%b)
```

Output

```
Enter the first number! 20
Enter the second number! 13
33
7
260
1.5384615384615385
1
8192000000000
7
```

Comparison operator

These operators compare the values of the two operands and return you the result in Boolean form i.e. True or false.

a) == Equal to each other
b) != Not equal
c) > Greater
d) < Less
e) >= Greater than or equal to
f) <= Less than or equal to

Example to show arithmetic and comparison operator
WAP to check whether a no. is even or odd
If any no. is divisible by 2 then the no. is said to be even
We will take the modulus of the no. instead of division.

```
a = int(input("Enter the first number!"))
if a%2==0: # if the mod of no is 0, that means it is completely
           divisible by 2 and no is even
    print("No is even")
else:
    print("No is odd")
```

Enter the first number! 55

OP No is odd

Code

```
a = int(input("Enter the first number:"))
if a%2==0:
    print("No is even")
else:
    print("No is odd")
```

Output
Enter the first number: 55
No is odd

Assignment Operators
The right expression's value is assigned to left operand

or

(a) =
(b) +=
(c) -=
(d) *=
(e) /=
(f) //=
(g) %=
(h) **=

Code

```
a = 10
b = 9
a = a+b
print(a)
h = 10
j = 9
h+=j
print(h)
```

Output
19
19

```
a = 10
b = 9
a = a-b
print(a)
h = 10
j = 9
h-=j
print(h)
```

Output
1
1

```
a = 10
b = 9
a = a*b
print(a)
h = 10
j = 9
h*=j
print(h)
```

Output
90
90

Logical operator
They work on conditions when we write 2 or more than 2 conditions
Logical operators come into picture.
1. AND - when all the given statements are correct then this will be true
2. OR - If any statement is correct out of given statement, it will be true.
3. NOT - Complement of any result.

AND

Code

```
a = 4
b = 5
c = 6
print (a<b and b<c)
print (a<b and b>c)
print (a<b and b<c or c<a)
```

Output

```
True
False
False
```

OR

Code

```
a = 4
b = 5
c = 6
print (a<b or b<c)
print (a<b or b>c)
print (a<b or b<c or c<a)
```

Output

```
True
True
True
```

NOT – Complement – opposite

Code

```
a = 4
b = 5
c = 6
print (not(a<b or b<c))
print (not(a<b or b>c))
print (not(a<b or b<c or c<a))
```

Output

```
False
False
False
```

AND – Multiplication (0*1=0) \ (0*0=0) \ (1*1=1)

OR – Addition (0+1=0) \ (0+0=0) \ (1+1=1)

NOT – opposite of result

a) AND - means, when both the conditions are true, only then the operation will take place.

b) OR - Means, when either of the conditions are true, the operation will take place

c) NOT - basically a compliment ? It reverses and the result.

Identity operators
These are used to compare equality and location of the object

Code
```
a = 10
id(a)
```
Output 43787064 48

```
b = 10
id(b)
```
Output 43787064 48

```
print(a is b)
```
Output True

Membership operators
They are typically used to find out if a element or a sequence is present in a # INDEXABLE data type.
They are
in - Returns true if the element or sequence is present/found
not in - to check in something is not present

Code # String
```
name = "Raghav Goel"
print("R" in name)
```
Output True

Code
```
name = "Raghav Goel"
print("*" in name)
```
Output False

Code
```
name = "Raghav Goel"
print("oel in name)
```
Output True

sequence as eof
False

BITWISE OPERATORS - they deal in binary coding,
means their operations are based in binary form.
Decimal to Binary to Decimal

Types of BITWISE OPERATORS
  # 1. AND
    2. OR
    3. XOR
    4. NOT
    5. BITWISE Left shift
    6. BITWISE Right shift

1. AND --> &
2. OR --> |
3. XOR --> ^
4. NOT --> ~
5. BITWISE Left shift --> <<
6. BITWISE Right shift --> >>

Date – 14/03/23

2^3	2^2	2^1	2^0
8	4	2	1

Decimal 7 5

Binary 111 101

OR
Addition

7 OR 5

7	1	1	1
5	1	0	1
7/5	1	1	1

5 OR 2

5	1	0	1
2	0	1	0
5/2	1	1	1

3 OR 2

3	0	1	1
2	0	1	0
3/2	0	1	1

#BITWISE for
7 OR 5
5 OR 2
3 OR 2

Truth Table of OR

A	B	OR
1	1	1
1	0	1
0	1	0
0	0	0

AND
multiplication

Truth Table

A	B	AND
1	1	1
1	0	0
0	1	0
0	0	0

7	1	1	1
5	1	0	1
7&5	1	0	1

XOR (^)

XOR problem was the challenge which gave rise to ARTIFICAL NETWORK DEEP NEURAL NETWORKS: Multilayered perceptions

A	B	XOR
1	1	0
1	0	1
0	1	1
0	0	0

7	1	1	1
5	1	0	1
7^5	0	1	0

BITWISE NOT

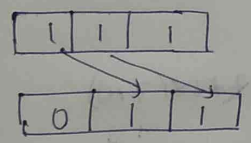
Trick --> Suppose we have to find the BITWISE NOT of 7

We will add 1 to the number and put a negative sign in front of it.

Code
```
print(~7)
print(~56)
print(~68)
print(~-7)
```

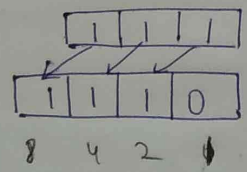
Output
```
-8
-57
-69
 8
```

Right shift (>>)

7 

= 3

Bitwise left shif (<<)

7 

8 4 2 1

Bitwise Right shift

Code
```
print(7>>1)
```

Output
```
3
```

Bitwise left shift

Code
```
print(7<<1)
print(5<<1)
```

Output
```
14
10
```

Operators precedence in Python

Operators	
( )	Parenthesis
**	Exponent
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, -	Addition, subtraction
<<, >>	Bitwise shift operators
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
==, !=, >, >=, <, <=, is, is not, in, not in	Comparison, identity, Membership operators
not	logical NOT
and	logical AND
or	logical OR

Example

exp = 4*7*(6-9+5**9)/(187*6*(7+6))

You will always move from left to right to solve any expression
except in the case of EXPONENTIALS.
d = 4**6**7**8. Here we will move from
Right to left

Code

```
exp = 4*7*(6-9+5**9)/(187*6*(7+6))
```

Output

```
3749.3086521321816
```

What will be the value of x in the following python expression.
x = int (43.55 + 2/2)

(a) 43
(b) 44
(c) 22
(d) 23

(b) 44

What is the value of the following expression?
2+4.00, 2**4.0

(a) (6.0, 16.0)
(b) (6.00, 16.00)
(c) (6, 16)
(d) (6.00, 16.0)

(a) (6.0, 16.0)

What are the values of the following python expressions?
2**(3**2)
(2**3)**2
2**3**2

(a) 64, 512, 64
(b) 64, 64, 64
(c) 512, 512, 512
(d) 512, 64, 512

(d) 512, 64, 512

~~Which of the following~~ What will be the value of the following
((((13+5)*2)-4)/2) -13 = ?

Which of the following operator has its associativity from right to left.

(a) +
(b) **
(c) /
(d) %

(b) **

Date – 15/03/23

Code
```
string1 = "Ragh\nfg"
print(string1)
```

Output
```
Ragh
fg
```

Code
```
string2 = r"Ragh\nfg"
print(string2)
```

Output
```
Ragh\nfg
```

Code
```
string3 = "Ragh\\nfg"
print(string3)
```

Output
```
Ragh\nfg
```

IF - ELSE - ELIF STATEMENTS

These statements are used for the decision making in python.
Just like general life, in python we have decision making using either this or that.

We use IF Statement to run a block of code when a certain condition is met.
Very important and one of the most frequently used conditional statement.

Syntax ---> Reserved Keyword (if) (condition):
    # Automatic Indent of 4 whitespaces (Action / what is suppose to happen)

IF - ELSE is like EITHER - OR

If your indents are aligned, python will consider that all the statements
written under that allignment are part of same conditional block
  i.e. IF_ELSE_FOR_WHILE

Example 1

Code
```
a = int(input("Enter the number"))
if (a % 2 == 0):
    print("The Number Inputted", a, "is Even")
```

Example 2
To check if a number is even or odd
with IF and Else bith

Code

```
a = int (input("Enter the number"))
if (a%2 == 0):
    print(" The Number Inputted ", a," is Even")
else:
    print (" The Number Inputted ", a," is Odd ")
```

Output

```
Enter the number 23
The Number Inputted 23 is Odd
```

Example 3
To check if a number is even or odd
with IF and Else ~~with~~ elif

Code

```
a = int ( input(" Enter the number "))
if (a == 0) :
    print("Don't know")
elif (a%2 == 0):
    print(" The Number Inputted ", a," is Even")
else:
    print (" The Number Inputted ", a," is Odd")
```

Output

```
Enter the number 0
Don't know
```

WAP to find out if an inputted no is positive or Negative

```
a = int ( input(" Enter the number "))
if (a>0):
    print(" The Number Inputted ", a," is Positive")
else:
    print(" The Number Inputted ", a," is Negative")
```

Output

```
Enter the number -2
The Number Inputted -2 is Negative
```

Now is zero a positive or a negative no
now to combat such conditions, we use ELIF
ELIF -- is a way to introduce more than 2 conditions in if-else conditions

Code

```
a = int (input(" Enter the Number "))
if (a>0):
    print(" The Number Inputted ", a," is Positive")
elif (a==0):
    print(" The Number Inputted ", a," is Zero ")
else:
    print(" The Number Inputted ", a," is Negative ")
```

Output

```
Enter the number 4
The Number Inputted 4 is Positive
```

Date — 15/03/22

Code

```
a = int (input ("Enter a "))
b = int (input ("Enter b "))
if (a>b):
    print (a, "is greater than ", b)
elif (a==b):
    print (a, "is equal to ", b)
else:
    print (a, "is greater than ", b)
```

Output

```
Enter a 4
Enter b 4
4 is equal to 4
```

Date - 20/03/23

WAP where we take input for 3 different subjects (Maths, Physics, Chemistry) from user and print the grade based on AVERAGE MARKS

Average	Grades
# 91 - 100	A+
# 81 - 90	A
# 71 - 80	B+
# 61 - 70	B
# Rest all	C

Approach
Input the marks
Calculate the mean
Use if-else-elif to categorize

Code

```
a = int(input("Enter marks of Maths"))
b = int(input("Enter marks of Physics"))
c = int(input("Enter marks of Chemistry"))
avg = round((a+b+c)/3, 0)
print("Your average marks are: ", avg)

if avg>=91:
    print("Your Grade is A+")
elif avg<=90 and avg>=80:
    print("Your Grade is A")
elif avg<=80 and avg>=71:
    print("Your Grade is B+")
elif avg<=70 and avg>60:
    print("Your Grade is B")
else:
    print("Your Grade is C")
```

Output

```
Enter marks of Maths 77
Enter marks of Physics 84
Enter marks of Chemistry 96
Your average marks are : 86
Your Grade is A
```

Whenever we have a range of values as a condition, we will use AND
This same if else statements are the very basic and important way
of logic for DECISION TREE

LOOPS in python — Loop means to do something over and over again

```
# Print Raghav Goel thousand times
print("Raghav Goel")
print("Raghav Goel")
print("Raghav Goel")
print("Raghav Goel")
print("Raghav Goel")
        .
        .
        .
```

Output

```
Raghav Goel
Raghav Goel
Raghav Goel
Raghav Goel
Raghav Goel
```

At the end, technology is a facilitator, Reduce complexity and manual job.
What is LOOP?
What are looping statements?
Looping statements is a kind of control statement which keeps on executing
a "STATEMENT" or a "BLOCK OF STATEMENT/CODE", "N" times till the "STOPPING CONDITION"
is met or satisfied.

Control statements are of 2 types:
#1 --> SELECTION STATEMENTS - IF, ELSE, Break, Pass, Continue
#2 --> LOOPING STATEMENTS - WHILE, FOR

WHILE LOOP

""" In Python, WHILE loop has 3 parts:

1. INITIALISATION --> from where does the WHILE loop start.
2. CONDITION --> This will be our STOPPING or FULLFILLMENT condition.
3. INCREMENT/DECREMENT --> In which direction does the execution move.

Our choice of INCREMENT AND DECREMENT will determine from where does the INITIALISATION starts and what the condition will be.

NOTE: Never ever forget to put a stopping condition otherwise the system will enter the finite loop i.e. will keep on running till perpetuality.

WAP to print your name 5 times

Approach 1 -->

Code

```
print("raghav goel")
print("raghav goel")
print("raghav goel")
print("raghav goel")
print("raghav goel")
```

Output

```
raghav goel
raghav goel
raghav goel
raghav goel
raghav goel
```

Approach 2

Code

```
name = input("Enter your full name: ")
print(name * 5, " ")
```

Output

```
Enter your full name : raghav goel
raghav goelraghav goelraghav goelraghav goelraghav goel
```

Approach 3 --> Using while loop

Code

```
name = input("Enter your full name : ")
i = 0
while i < 5 :
    print(name)
    i = i+1
```

Output

```
Enter your full name : Raghav Goel
Raghav Goel
Raghav Goel
Raghav Goel
Raghav Goel
Raghav Goel
```

General Conventions for a while loop.

1. Initialisation is done using a variable "i"

2. Usually it will start from 0 and usually use increament the value of i

INCREMENT

Code

```
name = input("Enter your full name :")
i = 0 # Initialisation Statement
while i < 5: # Conditional Statement / Stopping statement
    print(name)
    i = i+1  # i will become 1, then 2, then 3, then 4
             2  then
             3  # the moment it become 5
             4  # the execution will stop
             5
```

Infinite loop Don't Do THIS

Code

```
name = input("Enter your full name: ")
 i = 5 # Initialization Statement, the value of i is 0
while i < 5: Conditional Statement/ Stopping statement
    print(name)
    i = i - 1  # Decreament
               # i will become 4, then 3, then 2, then 1, then
               # the moment it becomes 0 and continue
```

DECREMENT

Code

```
name = input("Enter your full name :")
i = 5
while i>0:
  print(name)
  i = i-1  # Decreament
           # i will become 4, then 3, then 2, then 1  then  0
           # the moment it becomes 0
           # the execution will stop.
```

Output

```
Enter your full name: Raghav Goel
Raghav Goel
Raghav Goel
Raghav Goel
Raghav Goel
Raghav Goel
```

WAP to print the f_name and l_name of the user, by USER defined times.

Code

```
f_name = input("Enter the first name ")
l_name = input("Enter the last name ")
n = int(input("How many times you wish to print the name:"))
i = 0
while i<n:
    print(f_name, l_name)
    i = i+1
```

Output

```
Enter the first name Raghav
Enter the last name Goel
How many times you wish to print the name: 2
Raghav Goel
Raghav Goel
```

```
f_name = input("Enter the first name: ")
l_name = input("Enter the last name: ")
n = int(input("How many times you wish to print the name: "))
i = n
while i > 0:
    print(f_name, l_name)
    i = i - 1
```

Output

```
Enter the first name: r
Enter the last name : g
How many times you wish to print the name: 6
r g
r g
r g
r g
r g
r g
```

Date - 21/03/23

WAP to display the square of first 15 numbers using WHILE LOOP
Output will be : 1, 4, 9, 16, 25, 36, . . . 225 (5-Mins)

Solution

Code

```
i = 1
while i<=15:
    square = i*i
    print("Square of", i, "is", square)
    i = i + 1
```

Output

```
Square of 1 is 1
Square of 2 is 4
Square of 3 is 9
Square of 4 is 16
Square of 5 is 25
Square of 6 is 36
Square of 7 is 49
Square of 8 is 64
Square of 9 is 81
Square of 10 is 100
Square of 11 is 121
Square of 12 is 144
Square of 13 is 169
Square of 14 is 196
Square of 15 is 225
```

Code

```
i = 1
while i<=15:
    print("Square of", i, "is", i*i)
    i = i + 1
```

Output — Same output

```
Square of 1 is 1
Square of 2 is 4
Square of 3 is 9
Square of 4 is 16
Square of 5 is 25
Square of 6 is 36
Square of 7 is 49
Square of 8 is 64
Square of 9 is 81
Square of 10 is 100
Square of 11 is 121
Square of 12 is 144
Square of 13 is 169
Square of 14 is 196
Square of 15 is 225
```

WAP to find the sum of all the numbers till the user defined no
eg : user enters 5
Output will be : 5+4+3+2+1+0 = 15

Code

```
n = int(input("Enter the number: "))
sum = 0
i = 0
while i<=n:
    sum = sum + i
    i = i + 1
print("The sum of the numbers is:", sum)
```

Output

```
Enter the number: 5
The sum of the numbers is 15
```

DON'T DO IT

```
n = int( input("Enter the number"))
i = 0
while i<=n:
    sum = sum+i
    i = i+1
print(i)
```

Enter the number 5

Type Error

Sum is to be printed not i

WAP to find out the factorial of a user inputted number.

User entered 4

Output will be : 4*3*2*1 = 24

Code

```
n = int(input("Enter the number: "))
fact = 0  # Most common mistake, initialize Fact with 1
i = 0  # 2nd common mistake, intialize i with i = 1
```

Code

```
n = int(input("Enter the number: "))
fact = 1
i = 1
while i<=n:
    fact = fact + i
    i = i + 1
print("The Factorial of the number is: ", fact)
```

Output

Enter the number: 4

The Factorial of the numbers is: 24

#WAP to allow a user to input a number and you have to find the sum of all the digits
of the inputted number

eg : 20081992
output = 2+0+0+8+1+9+9+2 = 32

Code

```
n = int input("Enter the number: "))
s = 0
while n!=0:
    mod = n%10    # 456%10 = 6
    s = s + 10    # ~~floor value of div~~ is sum will be = s + mod = 6
    n = n//10     # floor division value will be 456//10 = 45
print("The sum of the digits will be",s)
```

Output

```
Enter the number: 67
The sum of the digits will be: 13
```

Armstrong no, is a no. whose digits when
multiplied with themselves (len of number times) and then added
are equal to the number itself.

eg: 407
len = 3
4^3 + 0^3 + 7^3 = 64 + 0 + 343 = 407
407 = 407 it is an armstrong number

eg : 46
len = 2
4^2 + 6^6 = 16+32 = 52
52 != 46 hence 46 is not a armstrong number

eg : 153
len = 3
1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153
153 is an armstrong number

Date - 22/03/23

3 digit armstrong numbers.

Code

```
n = int(input("Enter the number to be checked: ")) # will never be 0
length = len(str(n)) # will give the order of inputted no.
s = 0 # Initializing sum for calculating armstrong no.
i = n
while i > 0:
    mod = i % 10    # 456 % 10 = 6
    s = s + mod**length # s = 0 + 6^3
    i = i // 10 # Floor Division value will be = 456//10 = 45
# To check for Armstrong no we will use (if-else)
print("Order of inputed number", length)
if n==s:
    print("Inputted no", n, "is an armstrong no. ")
else:
    print("Inputted no", n, "is not an armstrong no. ")
```

Enter the number to be checked : 9474
Order of inputted number 4
Inputted no 9474 is an armstrong no.

WAP to take the user input in form of nos and over program ans to those nos
till the user hits a zero.

eg: 5,6,54,67,87,78, "0" = SUM(5,6,54,67,87,78)

Code

```
n = int(input("Enter the numbers to be added, or press zero to add "))
s = 0
while n != 0:
        s = s + n
        n = int(input("Enter the numbers to be added, or press zero to add "))
print("Sum of the inputted numbers is: ", s)
```

Output

```
You need to enter zero after you are done inputting numbers to add them
Enter the numbers to be added: 4
Enter the numbers to be added, or press zero to add 4
Enter the numbers to be added, or press zero to add 2
Enter the numbers to be added, or press zero to add 0
Sum of the inputted numbers is: 10
```

RANGE in PYTHON
What is a Range?
Range is a pre defined function in python which
is iterable over some user defined START and END point.
Range in python has 3 parameters defined.
SYNTAX for RANGE
Variable = range(1st para, 2nd para, 3rd para)
FIRST WAY to use RANGE in python is entering only 1st para

```
way1 = range(10)
print(way1)
```

In above representation, we are entering the following:
It range will start from "0"
Range will iterate from "0" till the inputted "Value - 1"
It will increment the values with "1 unit"

SECOND WAY to use the RANGE in Python is entering 1st para 2nd para ~~and 3rd para~~

```
way2 = range(1, 10)
print(way2)
```

In the above representation, we are entering the following:
It range will start from "1"
Range will iterate from "1" till the inputted "Value - 1" "9"
It will increment the values with "1 unit"

```
range(1,10)
```

THIRD WAY to use the RANGE in Python is entering 1st para and 2nd para and 3rd para

```
way3 = range(1, 10, 2)
print(way3)
```

In the above representation, we are entering the following:
The Range will start from "1"
The Range will iterate from "1" till the inputted (value-1) "9"
It will increment the values with 3rd para units - "2 units"

```
range(1,10,2)
```

FOR LOOPS IN PYTHON

This is the second type of loop which we will discuss.
. The major difference b/w while and FOR loop is that for loop works in bound, which means it will work over the iterable object and range of values, unlike while loops
Because of above, FOR loops will never become infinite loops, unlike while loops.
FOR loops in python are very different form that of C:

SYNTAX for FOR LOOPS in PYTHON

"for" "Variable" "in" Iterable_Obj":
print/ condition

Default variable is "i". Again it is a norm not a rule

```
way2
range(1,10)
for i in way2:
    print(i)
```

Output

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

```
for i in range(1,10,2)
    print(i)
```

Output

```
1
3
5
7
9
```

```
name = "Raghav G" # String is an iterable object
for i in name:
    print(i)
```

Output

```
R
a
g
h
a
v
G
```

"i" is like a cursor which shifts 1 place in every iteration.

WAP to find the factorial of user inputted no.

Solution

```
n = int(input("Enter the number till you wish to calculate the Factorial: "))
fact = 1
for i in range(1, n+1):
    fact = fact * i
print("The Factorial of", n, "is", fact)
```

Output

```
Enter the number till you wish to calculate the Factorial: 4
The Factorial of 4 is 24
```

WAP to find the sum till the user inputted no.

Code

```
n = int(input("Enter the number till you wish to calculate the sum: "))
s = 0
for i in range(1,n+1):
    s = s + i
print("The Sum till", n, "is", s)
```

Output

```
Enter the number till you wish to calculate the sum: 3
The Sum till 5 is 15
```

WAP to print number between (0-50) which are divisible by 3

```
for i in range (0,51):
    if i%3 == 0:
        print(i)
```

```
0
3
6
9
12
15
18
21
24
27
30
```

```
for i in range(1,51)
```

WAP to paint the multiplication table of user inputted no.
output:
4
1*4 = 4
2*4 = 8 . . . till
10*4 = 40

Code

```
n = int(input("Enter the number for multiplication table till 10:"))
for i in range(1,11):
    print(i," * ", n, " = ", n*i)
```

Output

```
enter the number for multiplication table till 10: 4
1 * 4 = 4
2 * 4 = 8
3 * 4 = 12
4 * 4 = 16
5 * 4 = 20
6 * 4 = 24
7 * 4 = 28
8 * 4 = 32
9 * 4 = 36
10 * 4 = 40
```

"FOR" loops can also be very useful in solving problems which involves reverse

Code

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

Output

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

WAP for finding out first 10 even numbers in reverse using for loop.
For solving the above, we need to understand it well

```
for i in range(20, 0, -1):
    if (i%2 == 0):
        print(i)
```

```
for i range(20, 0, -2):
    print(i)
```

Output

```
20
18
16
14
12
10
8
6
4
2
```

Q. WAP for finding out first 10 ODD numbers in reverse using for loop.

```
for i in range(20,0,-1):
    if (i%2 != 0):
        print(i)
```

```
for i in range(19, 0, -2):
    print(i)
```

Output

```
19
17
15
13
11
9
7
5
3
1
```

3 very important KEYWORDS:

1. BREAK
2. CONTINUE
3. PASS

All these 3 keywords play a very crucial role on the code which we write

1. BREAK Keyword

BREAK - It provides us with opportunity to EXIT the existing loop
when an external condition is MET/TRIGGERED

Simple example of BREAK keyword.
Lets WAP to print first 10 numbers.

```
for i in range(1,11):
    print(i)
```

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

Code

```
for i in range(1,11):
    if i == 5:
        break
    print(i)
```

Output

```
1
2
3
4
```

Code

```
for i in range(1,11):
    if i % 7 == 0:
        break
    print(i)
```

Output

```
1
2
3
4
5
6
```

BREAK is used to terminate a program or an execution
when an external condition is met

We all will design a McDonald outlet where lies are selling Burgers.

"First step will be: User will come to our counter/vending machine and ask us for some burgers.
We will then provide a way to enter the inputted quantity and run a WHILE loop and provide users with that quantity / burgers.

Code

```
How_Many_Burgers_Do_you_Want = int(input("How_Many_Burgers_Do_you_Want? "))
Which_Burger_You_want = input("Which Burger: ")
i = 1
while i <= How_Many_Burgers_Do_you_Want:
    print(i, Which_Burger_You_want)
    i = i + 1
```

Output

```
How_Many_Burgers_Do_you_Want : 5
Which Burger: Veg Burger
1 Veg Burger
2 Veg Burger
3 Veg Burger
4 Veg Burger
5 Veg Burger
```

Now it is possible that we have 10 burgers in stock.
and the user asks for more than 10 burgers.

We have a scenarios:
#1. We give the user the burgers which we have / Available burgers.
Customer gets pissed off and does not buy any burger

Code 2

Customer gets pissed off and does not buy any burger

Code

```
Burgers_in_Stock = 7
How_Many_Burgers_Do_you_Want = int(input("How_Many_Burgers_Do_you_Want: "))
Which_Burger_You_Want = input("Which Burger: ")
i = 1
if How_Many_Burgers_Do_you_Want <= Burgers_in_Stock:
    while i <= How_Many_Burgers_Do_you_Want:
        print(i, which_Burger_You_Want)
        i = i+1
else:
    print("Sorry sir, we are out of stock")
```

Output

```
How_Many_Burgers_Do_you_Want: 8
Which Burger: Veg
Sorry Sir, we are out of stock.
```

#1. We give the user the burger which we have/Available burgers

Code

```
Burgers_in_Stock =
How_Many_Burgers_Do_you_Want = int(input("How_Many_Burgers_Do_you_Want: "))
Which_Burger_You_Want = input("Which Burger: ")
i = 1
while i <= How_Many_Burgers_Do_you_want:
    if i > Burgers in Stock:
        print("Sorry sir/madam we only have 7 Burgers")
        break
    print("Which burger you want")
    i = i + 1
print("Thank you and have a nice day")
```

Code

Date - 28/03/23

CONTINUE
CONTINUE is used to simply continue the code and SKIP a particular EVENT/ITERATION
In case of BREAK, we wanted to exit the loop when an external condition was met,
whereas in case of CONTINUE, we will skip it and not exit the loop.

EXAMPLE

Break

Code

```
for i in range (0, 21):
    if i%.5==0:
        break
    print(i)
```

→ Output

```
1
2
3
4
```

Continue

Code

```
for i range (1,50):
    if i%.5==0
        continue
    print(i)
```

Output

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

WAP to find all the numbers from 0-20 except 3, 6 and their multiples

Code

```
for i in range(0,21)
    if i %. 3 == 0:
        Continue
    print(i)
```

Output

```
1
2
4
5
7
8
10
11
13
14
16
17
19
20
```

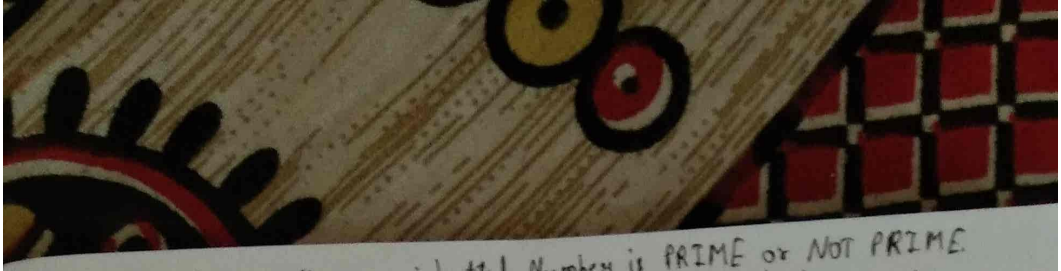
WAP to iterate over range 0-70. For all the multiples of 3 print "PYTHON"
and for all the multiples of 7, print "ROCKS". And for the common multiples print
"PYTHON ROCKS"

Code

```
for i in range(1,71):
    if (i%3==0 and i%7==0):
        print("PYTHON ROCKS")
        continue
    elif i%3==0:
        print("PYTHON")
        continue
    elif i%7==0:
        print("ROCKS")
        continue
    print(i)
```

Output

```
PYTHON ROCKS
1
2
PYTHON
4
5
PYTHON
ROCKS
8
PYTHON
10
11
PYTHON
13
ROCKS
PYTHON
.
.
.
67
68
PYTHON
ROCKS
```



WAP to find whether an inputted Number is PRIME or NOT PRIME.
What are Prime Nos? A No. which is divisible by itself and 1 is PRIME NO.
Ex - 2, 3, 5, 7, 11, 13, 17, 23, 29, ...

```
n = int(input("Enter the No to be checked for Prime : "))
if n == 1:
    print("Enter value greater than 1")
elif n>1:
    for i in range(2,n):
```