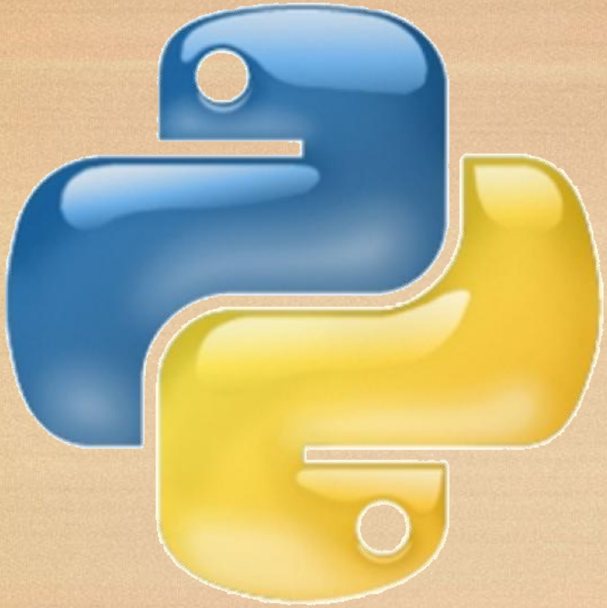
Introduction to Python



Dr. Mustafa M. Shiple

Robotics Technology Center

What is the code program ?

01 Data Types

02 Input and output instructions

03 Arithmetic and logic operation

04 Data Flow

Basic concepts

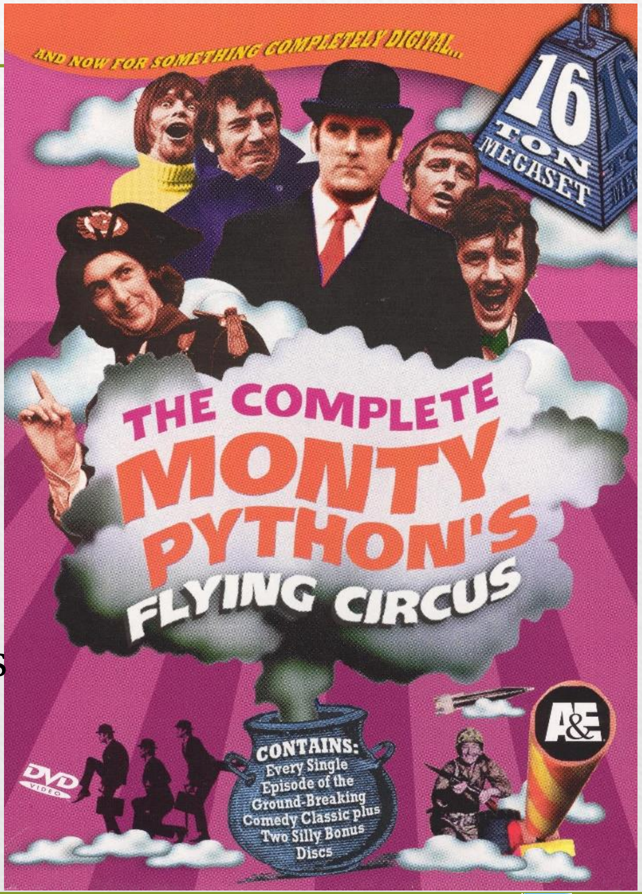
Alphabetically/lexically/syntactically/semantically Programming language/reserved words/language syntax (IL)/ meaning of program

Source file /Source code
Generated by programmer. Higher level language

COMPILATION / INTERPRETATION

Machine language
Generated by computer

Why Python?


AND NOW FOR SOMETHING COMPLETELY DIGITAL...
16 TON MEGASET
THE COMPLETE MONTY PYTHON'S FLYING CIRCUS
CONTAINS: Every Single Episode of the Ground-Breaking Comedy Classic plus Two Silly Bonus Discs
DVD VIDEO
A&E

Python applies to programming in many fields:

- Data science
- Writing system tools
- Developing applications with graphical UIs
- Writing network-based software
- Interacting with databases

Python is an interpreted language.

- Python is an interpreted language. an interpreter is a computer program that directly executes instructions written in programming or scripting language, without requiring them previously to have been compiled into a machine language program.

Python goals

- an **easy and intuitive** language just as powerful as those of the major competitors;
- **open source**, so anyone can contribute to its development;
- code that is as **understandable** as plain English;
- **suitable for everyday tasks**, allowing for short development times.

Tutorial Outline

- Basic types: numbers, strings
- Control structures
- Container types: lists, tuples, dictionaries
- Functions & procedures
- Classes & instances
- Modules & packages
- Exceptions
- Files & standard library
- What's new in python 2.0 and beyond

String

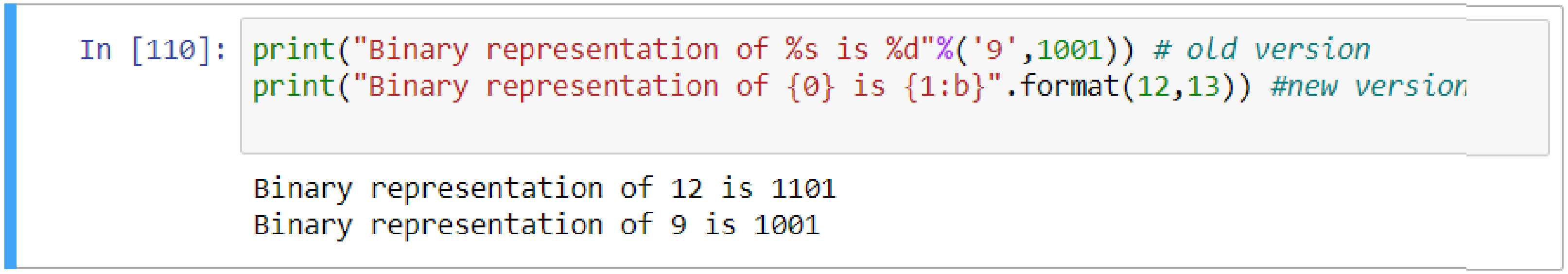
Data Containers

Strings (tuple & list)

"he"+"llo"	"hello"	# concatenation
"hello"*2	"hellohello"	# repetition
"hello"[0]	"h"	# indexing
"hello"[-1]	"o"	# (from end)
"hello"[1:4]	"ell"	# slicing
len("hello")	5	# size
"he" < "je"	True	# comparison
"e" in "hello"	True	# search

String formatting

"escapes:	\n , \t"
print("%c" % 97)	a
print("%6.3f" % 2.5)	2.500
print("%+10x" % 10)	+a
print("%.*f" % (4, 1.5))	1.5000



```
In [110]: print("Binary representation of %s is %d"%('9',1001)) # old version
          print("Binary representation of {0} is {1:b}".format(12,13)) #new version
```

```
Binary representation of 12 is 1101
Binary representation of 9 is 1001
```

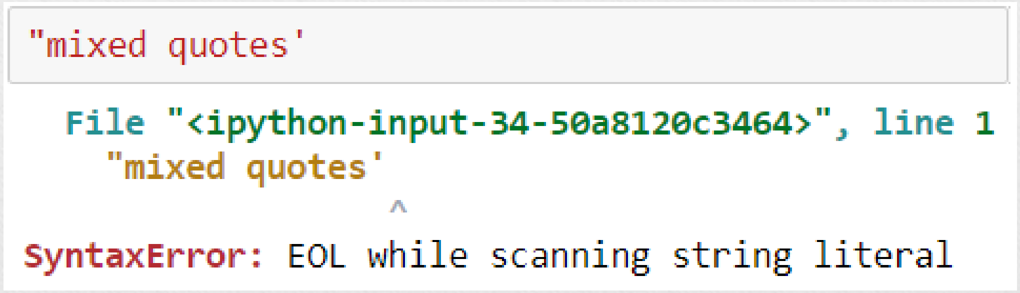
'single quotes' or “double quotes” changeable

quotes_single = 'a_string'
quotes_double = "a_string“

quotes_triple=“” a_string””
quotes_single == quotes_double== quotes_triple

True

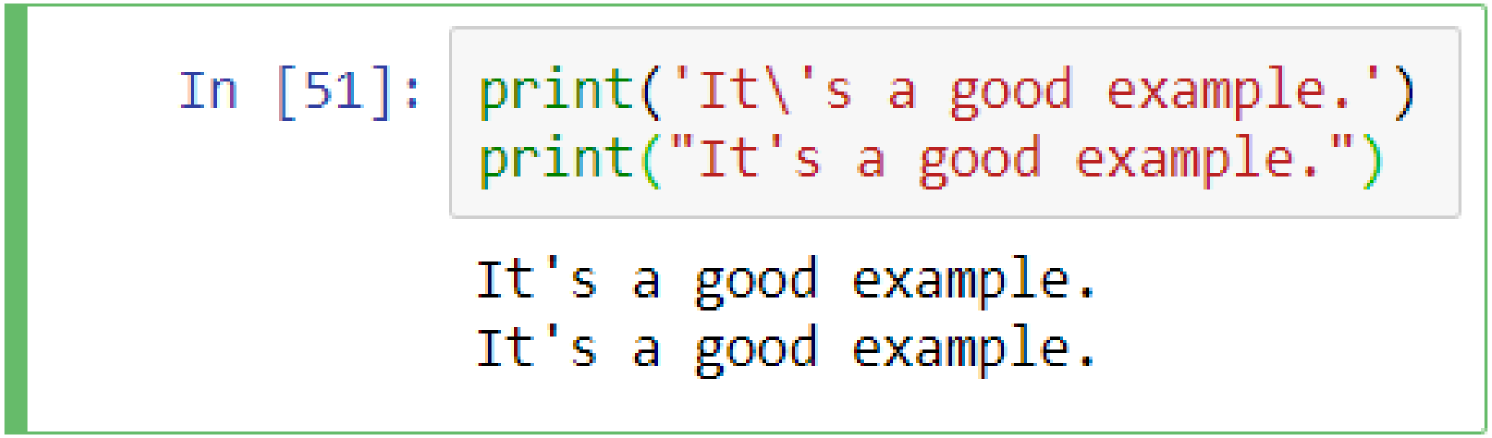
'single quotes' or “double quotes” don’t mix



```
"mixed quotes'

  File "<ipython-input-34-50a8120c3464>", line 1
    "mixed quotes'
                 ^
SyntaxError: EOL while scanning string literal
```

Escape or change

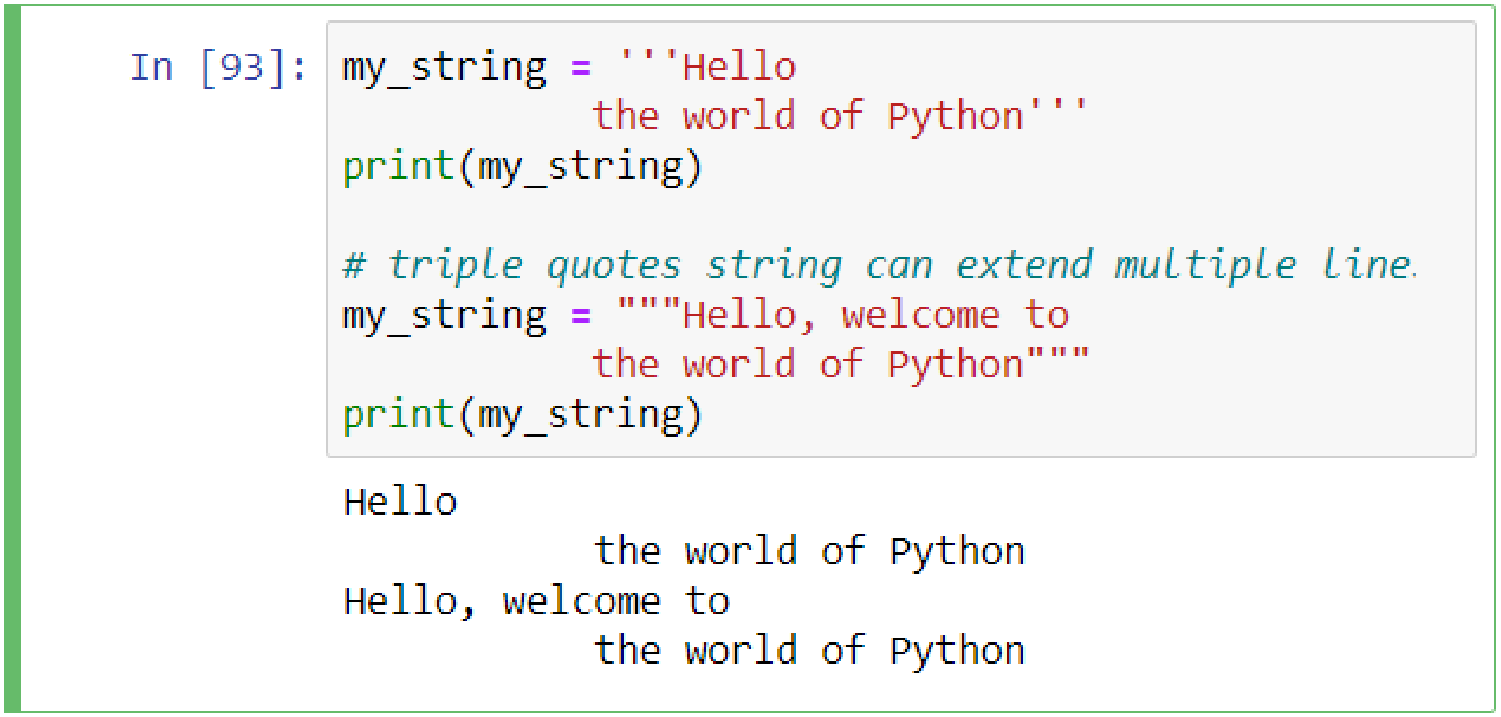


```
In [51]: print('It\'s a good example.')
         print("It's a good example.")

         It's a good example.
         It's a good example.
```

“‘triple quotes‘” or “‘””” triple quotes’”””

- For multi-line statements



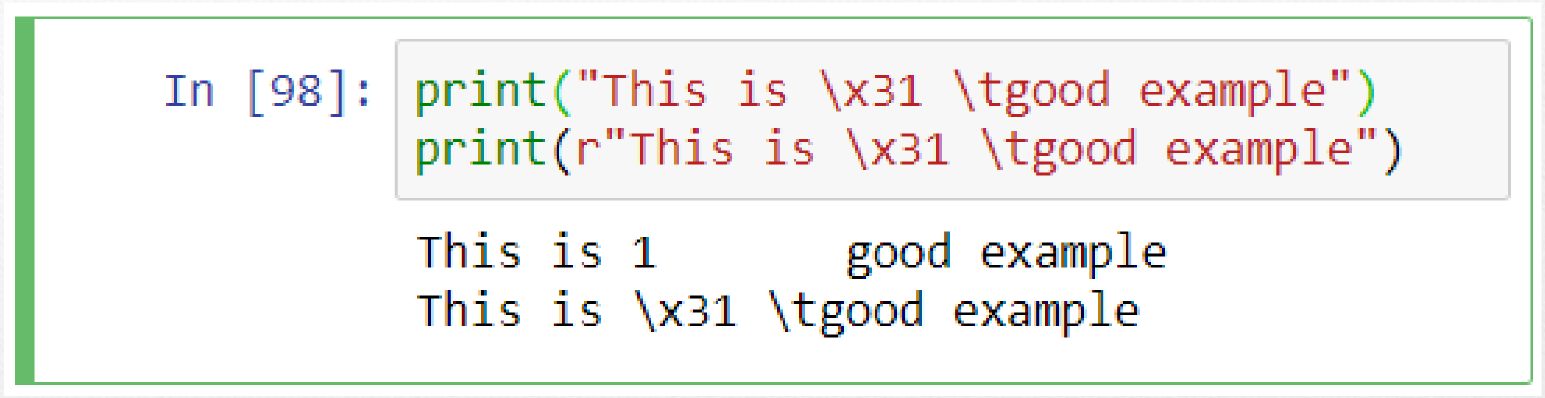
```
In [93]: my_string = '''Hello
                    the world of Python'''
         print(my_string)

         # triple quotes string can extend multiple line.
         my_string = """Hello, welcome to
                    the world of Python"""
         print(my_string)

         Hello
                    the world of Python
         Hello, welcome to
                    the world of Python
```

Raw String

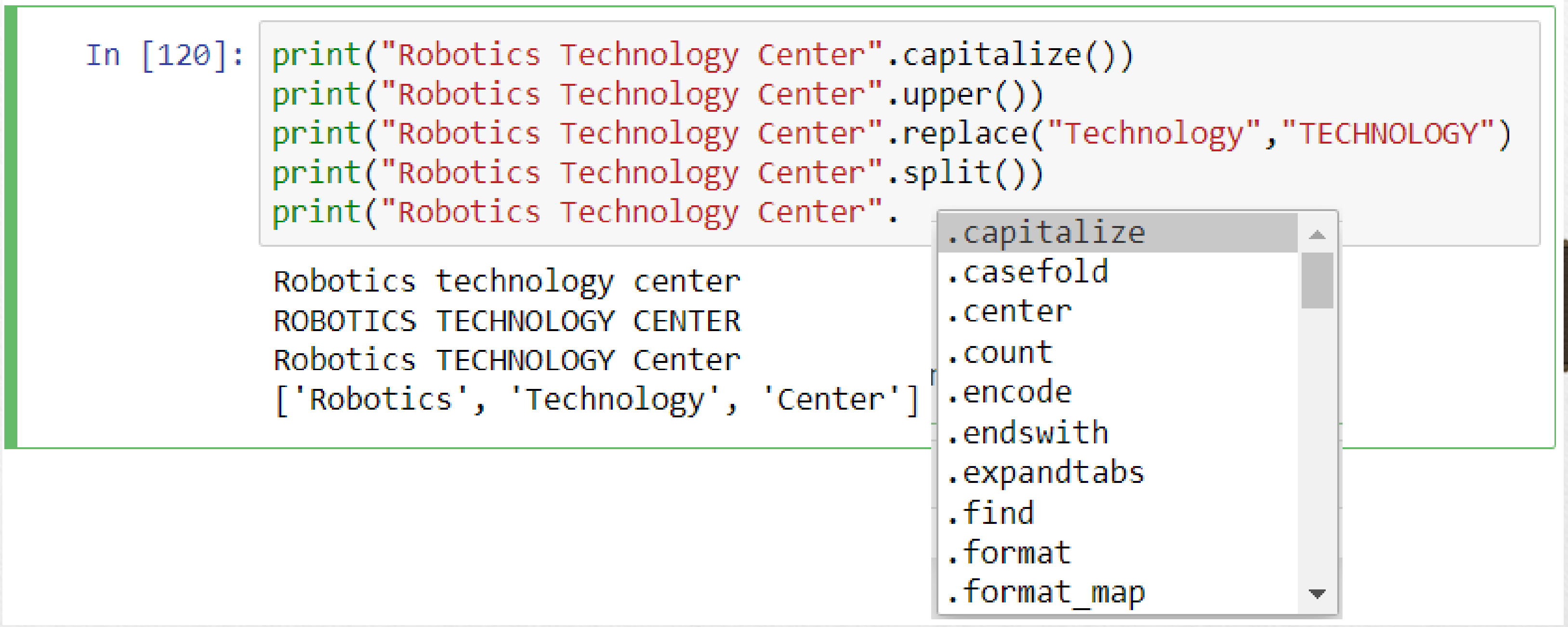
- To ignore escape sequence



```
In [98]: print("This is \x31 \tgood example")
         print(r"This is \x31 \tgood example")

         This is 1       good example
         This is \x31 \tgood example
```

String Methods



```
In [120]: print("Robotics Technology Center".capitalize())
          print("Robotics Technology Center".upper())
          print("Robotics Technology Center".replace("Technology","TECHNOLOGY"))
          print("Robotics Technology Center".split())
          print("Robotics Technology Center".

          Robotics technology center
          ROBOTICS TECHNOLOGY CENTER
          Robotics TECHNOLOGY Center
          ['Robotics', 'Technology', 'Center']
```

Tips (reverse)

```
sentence = "This is just a test"
sentence[::-1]
```

```
tset a tsuj si sihT
```

Exercise :

Write a code to insert in the middle:
Str_1=“Oldstring‟
Str_2=“NewString”

Output = "OldsNewStringtring"

Print arguments (end)

```
print("My name is", "Python.", end=" ")
print("Monty Python.")
```

Try \n,\t, without space

Print arguments (sep)

```
print("My", "name", "is", "Monty", "Python.", sep="-")
```

Try this:-

```
print("My", "name", "is", sep="_", end="*")
print("Monty", "Python.", sep="*", end="*\n")
```

Guess

```
x=20

y=30

z=40

print("The values are", x, y, z,
  end='!!!',sep='!***')
```

Inputs

Input statement

- Syntax

```
<variable> = input(<prompt>)
Type(variable)= text
```

- Example

```
weather = input("What is your name?")
```

Checkpoint: What is the exact output of this code?

```
name = input("What is your name? ")
print("My name", name*3, end='!!!',sep='…')
```

Data Types

Numerical systems

- Decimal → 235
- Hexadecimal → 0x123
- Octal → 0o256
- Binary → 0b11011

Decimal Numbers

- int – Integer: -5, 10, 77
- float – Floating Point numbers: 3.1457, 0.34

Check by :-

```
type(15)  → <type 'int'>
type(5.3)  → <type 'float'>
```

Mathematical operators

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Implicit Data casting

Operators	Meaning
()	Parentheses
**	Exponent
+x, -x, ~x	Unary plus, Unary minus, Bitwise NOT
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, -	Addition, Subtraction
<<, >>	Bitwise shift operators
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
==, !=, >, >=, <, <=, is, is not, in, not in	Comparisons, Identity, Membership operators
not	Logical NOT
and	Logical AND
or	Logical OR

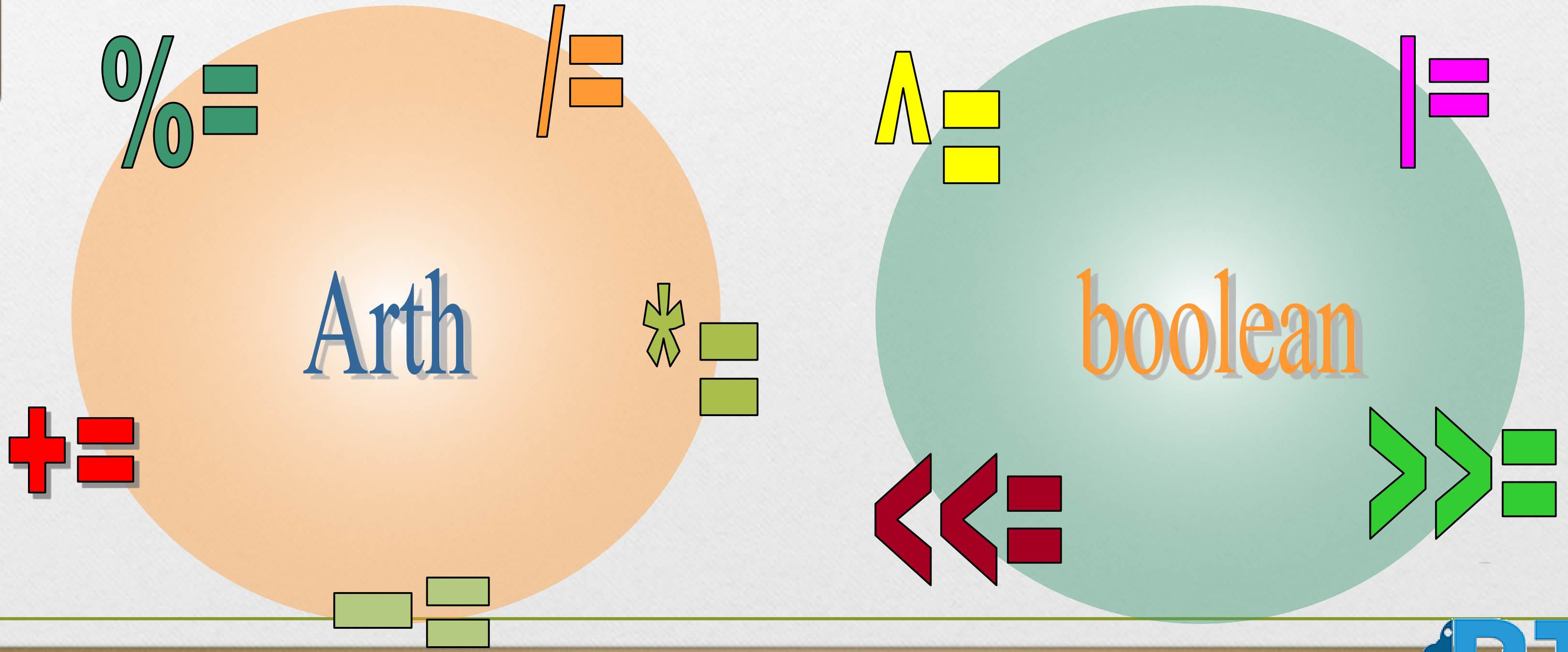
Associativity

Tips

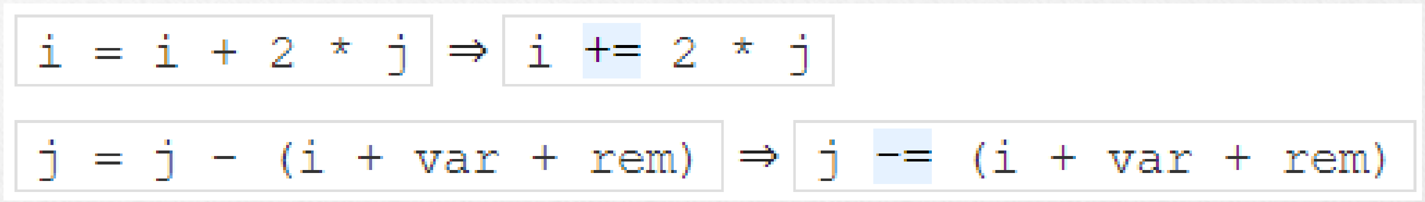
• x = y = z = 1	→associativity
• x, y, z= 1, 2, 3	→associativity
• x = y = z += 1	→non
• x < y < z	→(x < y) and (y < z)
• 2**-2	→0.25 !!! (- higher tan **)

Expressions compressed forms

X = X + 2 ; ⇒ X += 2 ;



Advanced



```
i = i + 2 * j ⇒ i += 2 * j
j = j - (i + var + rem) ⇒ j -= (i + var + rem)
```

Operator Overloading!

- Syntax based on position :
  - `3 + 3` → 6.
  - `"Hi" + "NTI"` → "HiNTI"
  - `"Hi NTI" * 3` produces "`Hi NTIHi NTIHi NTI`"
  - `"test %f" % 34` → "test 34"

Explicit Data casting (Conversion)

`int(3.3)`	→ 3	#float to int
`float(3)`	→ 3.0	#int to float
`str(3.3)`	→ “3.3”	#float to str
`float(“3.5”)`	→ 3.5	#str to float
`int(“7”)`	→ 7	#str to int
`Ord`(“A”)	→97	#ASCII of Letter
`Chr(97)`	→”A”	#Letter of ASCII

Exercise

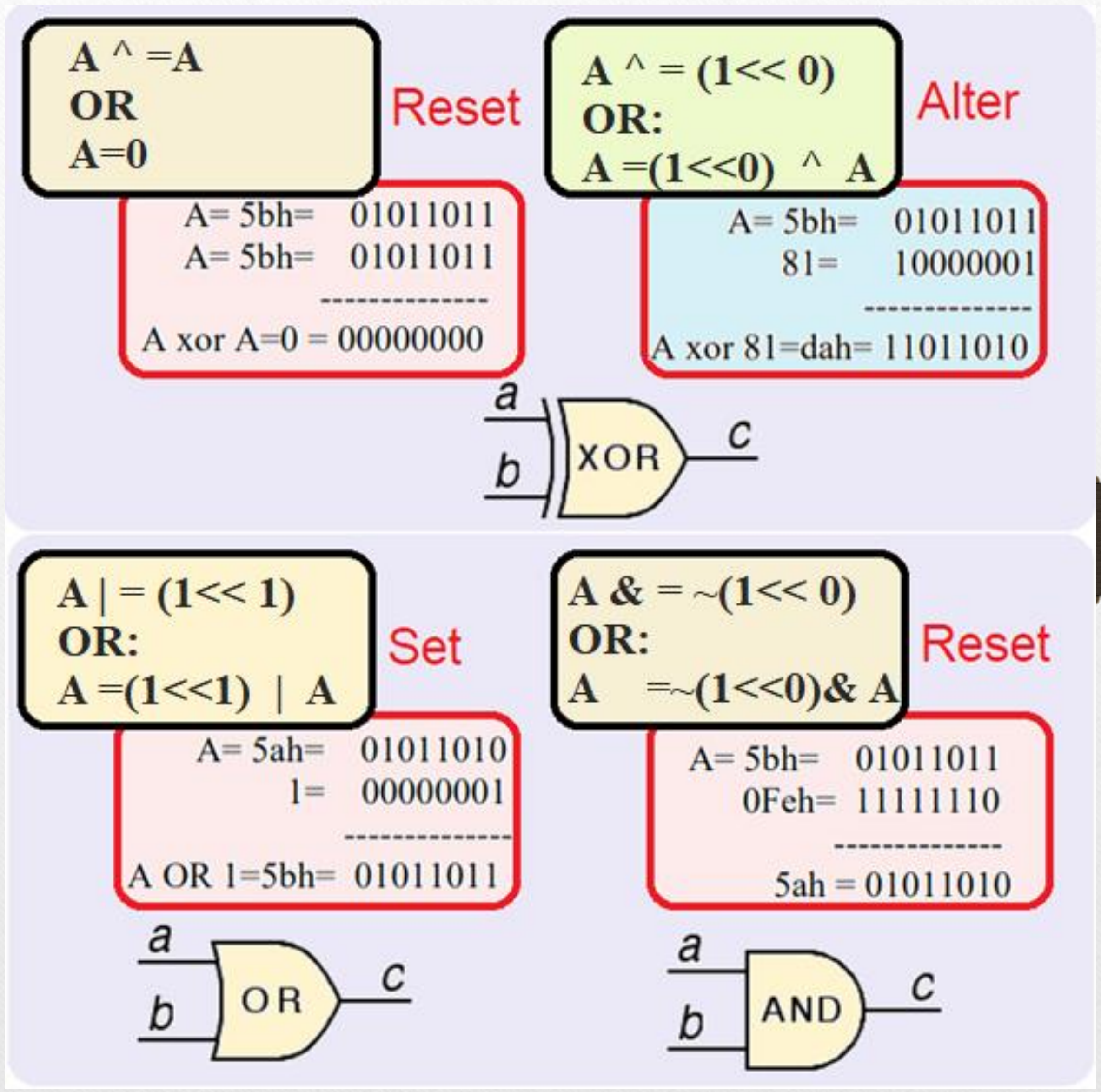
$$6x^2 - 17x + 12 = 0$$

$$X = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Boolean Datatype

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Bit manipulation



```
A ^ =A
OR
A=0
```
Reset
```
A= 5bh=     01011011
A= 5bh=     01011011
            -------------
A xor A=0 = 00000000
```

```
A ^ = (1<< 0)
OR:
A =(1<<0)  ^  A
```
Alter
```
A= 5bh=        01011011
81=            10000001
               -------------
A xor 81=dah=  11011010
```

```
A | = (1<< 1)
OR:
A =(1<<1)  |  A
```
Set
```
A= 5ah=          01011010
1=               00000001
                 -------------
A OR 1=5bh=      01011011
```

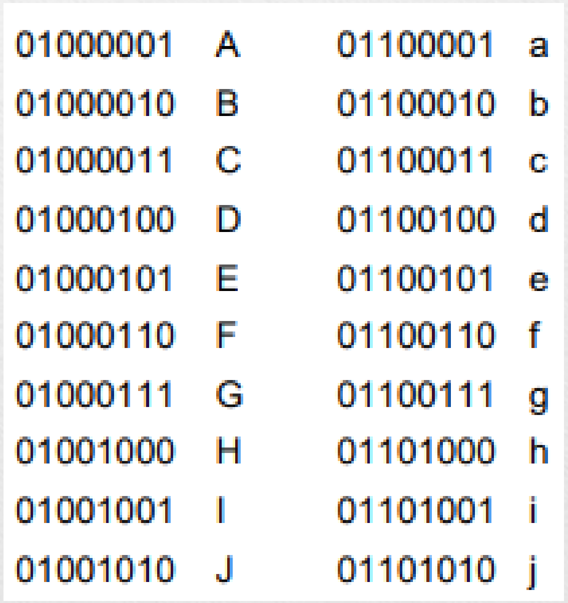
```
A & = ~(1<< 0)
OR:
A    =~(1<<0)& A
```
Reset
```
A= 5bh=   01011011
0Feh=     11111110
          -------------
5ah =     01011010
```

Bitwise operators

Operator	Name
&	AND
|	OR
^	XOR
~	NOT
<<	Zero fill left shift
>>	Signed right shift

Exercise

- Write a code to convert upper case to lower case
- Write a code to convert lower to upper case

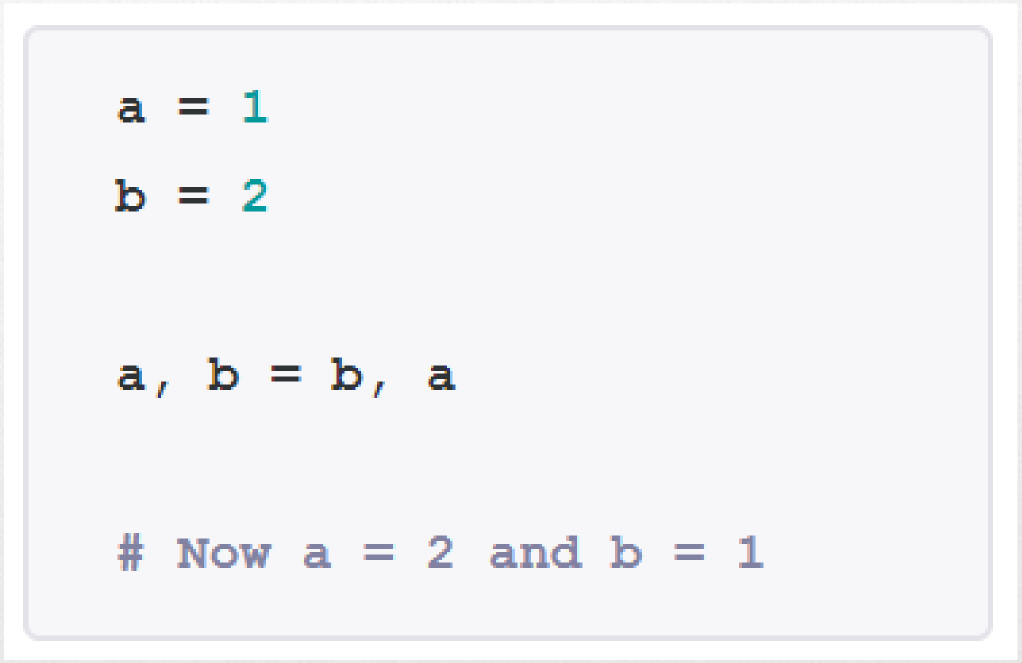


01000001	A	01100001	a
01000010	B	01100010	b
01000011	C	01100011	c
01000100	D	01100100	d
01000101	E	01100101	e
01000110	F	01100110	f
01000111	G	01100111	g
01001000	H	01101000	h
01001001	I	01101001	i
01001010	J	01101010	j

Exercises

- Change 3ed bit to be 1 for any given number?
- Multiply a given number by 8?
- Swap two numbers by xor?

swapping



```
a = 1
b = 2

a, b = b, a

# Now a = 2 and b = 1
```

Identifier

- Rules for identifiers :
  - first char alphabetic [a-z,A-Z] or underscore (_).
  - has only alphabetic, digit, underscore chars.
  - cannot duplicate a reserved word.
  - Variables are case sensitive.

Control Structure

Control Structures

```
if  condition:
    statements
elif  condition:
    statements] …
else:
    statements
```

```
while condition:

    statements
```

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

```
break
continue
```

Grouping Indentation

In Python:

```
for i in range(20):
    if i%3 == 0:
        print i
        if i%5 == 0:
            print "Bingo!"
    print "---"
```

In C:

```
for (i = 0; i < 20; i++)
{
    if (i%3 == 0) {
        printf("%d\n", i);
        if (i%5 == 0) {
            printf("Bingo!\n"); }
    }
    printf("---\n");
}
```

```
0
Bingo!
---
---
---
3
---
---
---
6
---
---
---
9
---
---
---
12
---
---
---
15
Bingo!
---
---
---
18
---
---
```

If ,elif, else

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

Exercise

Write a C program to check whether a given number is positive or negative.

Multiple conditions

OPERATOR	DESCRIPTION
and	Logical AND: True if both the operands are true
or	Logical OR: True if either of the operands is true
not	Logical NOT: True if operand is false

Short versions

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

```
print("A") if a > b else print("B")
```

Multiple conditions (and/or)

```
if a > b and c > a:
  print("Both conditions are True")
```

For Structure : sequence

for *var* in *sequence*:
    statements

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

List

```
for x in  "banana" :
```

String

```
for x in range(6):
```

0 → 5 Range (max value)

```
for x in range(2,6):
```

2 → 5 Range (min,max)

```
for x in range(2,6,2):
```

2 4 (min,max,step)

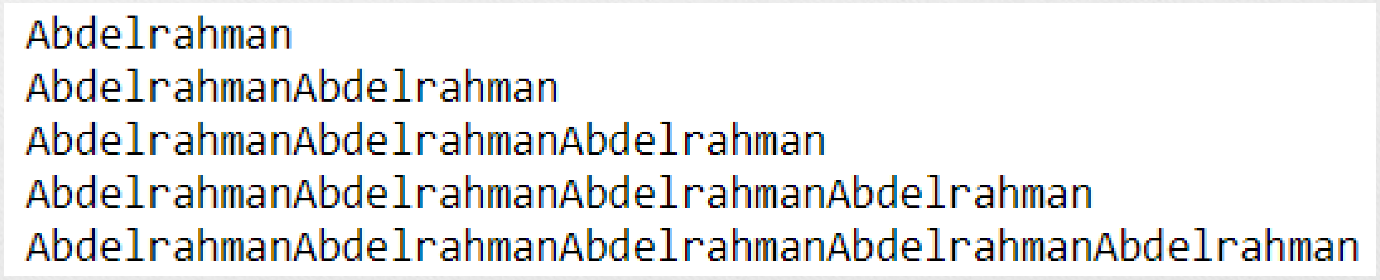
While Structure

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

Step

Exercise

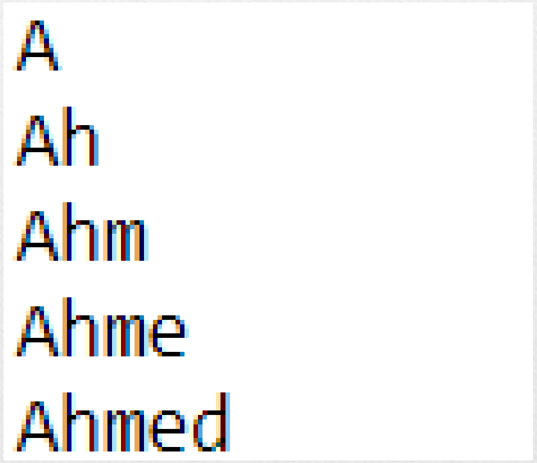
Write a code to print your name as?



```
Abdelrahman
AbdelrahmanAbdelrahman
AbdelrahmanAbdelrahmanAbdelrahman
AbdelrahmanAbdelrahmanAbdelrahmanAbdelrahman
AbdelrahmanAbdelrahmanAbdelrahmanAbdelrahmanAbdelrahman
```

Exercise

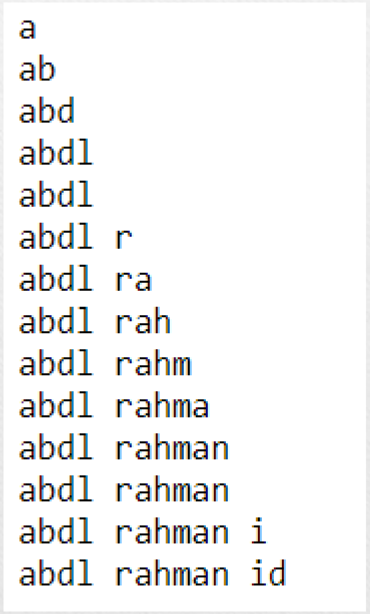
Write a code to print your name as?



```
A
Ah
Ahm
Ahme
Ahmed
```

Exercise

Write a code to print your name with out "e" as?
Example = "abdel rahman eid"



```
a
ab
abd
abdl
abdl
abdl r
abdl ra
abdl rah
abdl rahm
abdl rahma
abdl rahman
abdl rahman
abdl rahman i
abdl rahman id
```

Continue, Break, Else (for/while)

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

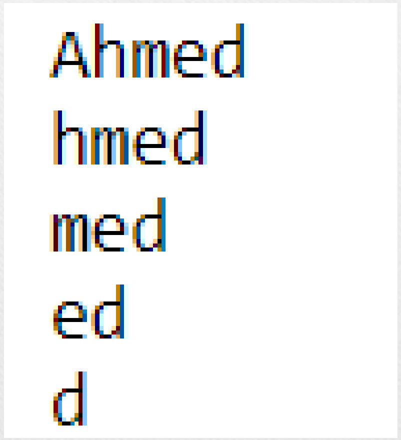
```
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

```
for x in fruits:
  if x == "banana":
    break
  print(x)
```

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

Exercise

Write a code to ?



```
Ahmed
hmed
med
ed
d
```

Exercise

Write a code to ?

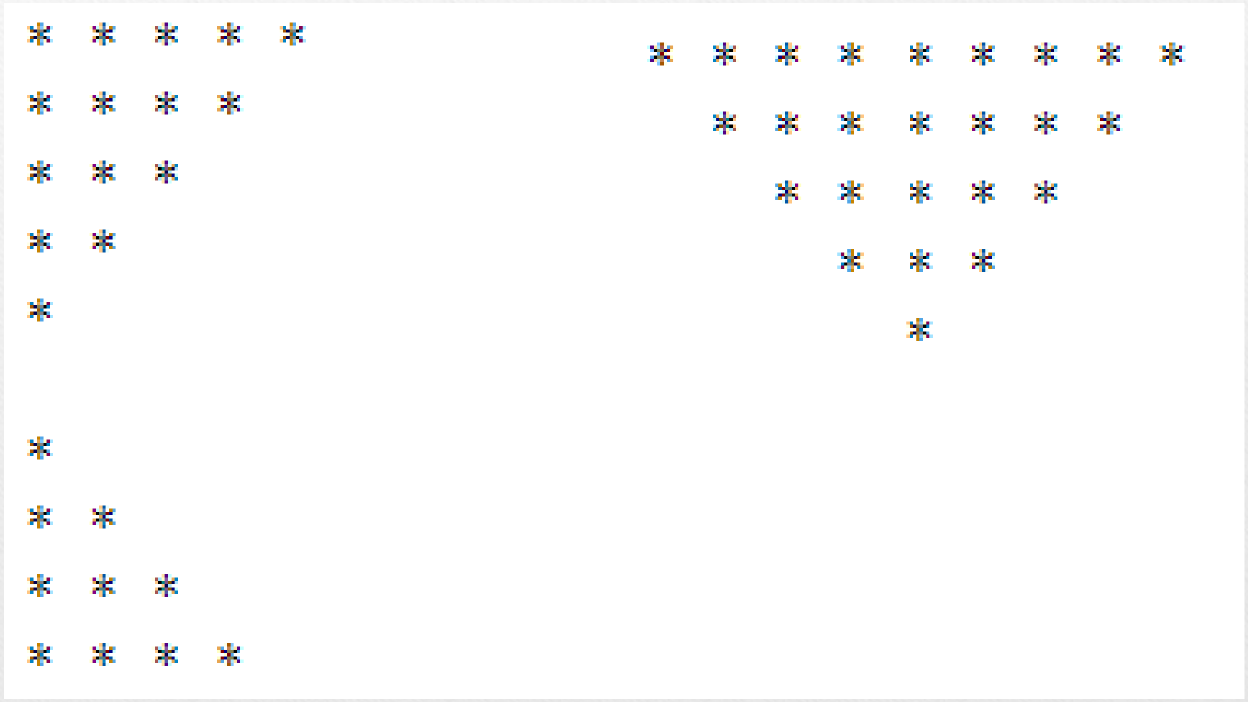
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

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

Exercise

Divisible by Three: Write a for loop that prints out all numbers from 1 to 100
that are divisible by three.

Quiz



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


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