Python

The Easy Way

Course Objectives

Learn about Python, its uses and really understand it.

Python Inventor

guido van rossum

Python History

Python History

Python was developed by Guido Van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C,C++, Unix shell and other scripting languages.

Python is copyrighted like Perl, Python source code is now available under GNU General Public License.

Python is now maintained by a core development team at the institute, although Guido Van Rossum still holds a vital role in directing its progress.

Python 1.0 was released in November 1994. In 2000, Python 2.0 was released. Python 2.7.11 is the latest edition of Python 2.

Python 3.0 was released in 2008, and Python 3.8 is the latest edition of Python 3.

Python History

Python is a high-level, interpreted, interactive and object oriented scripting language.

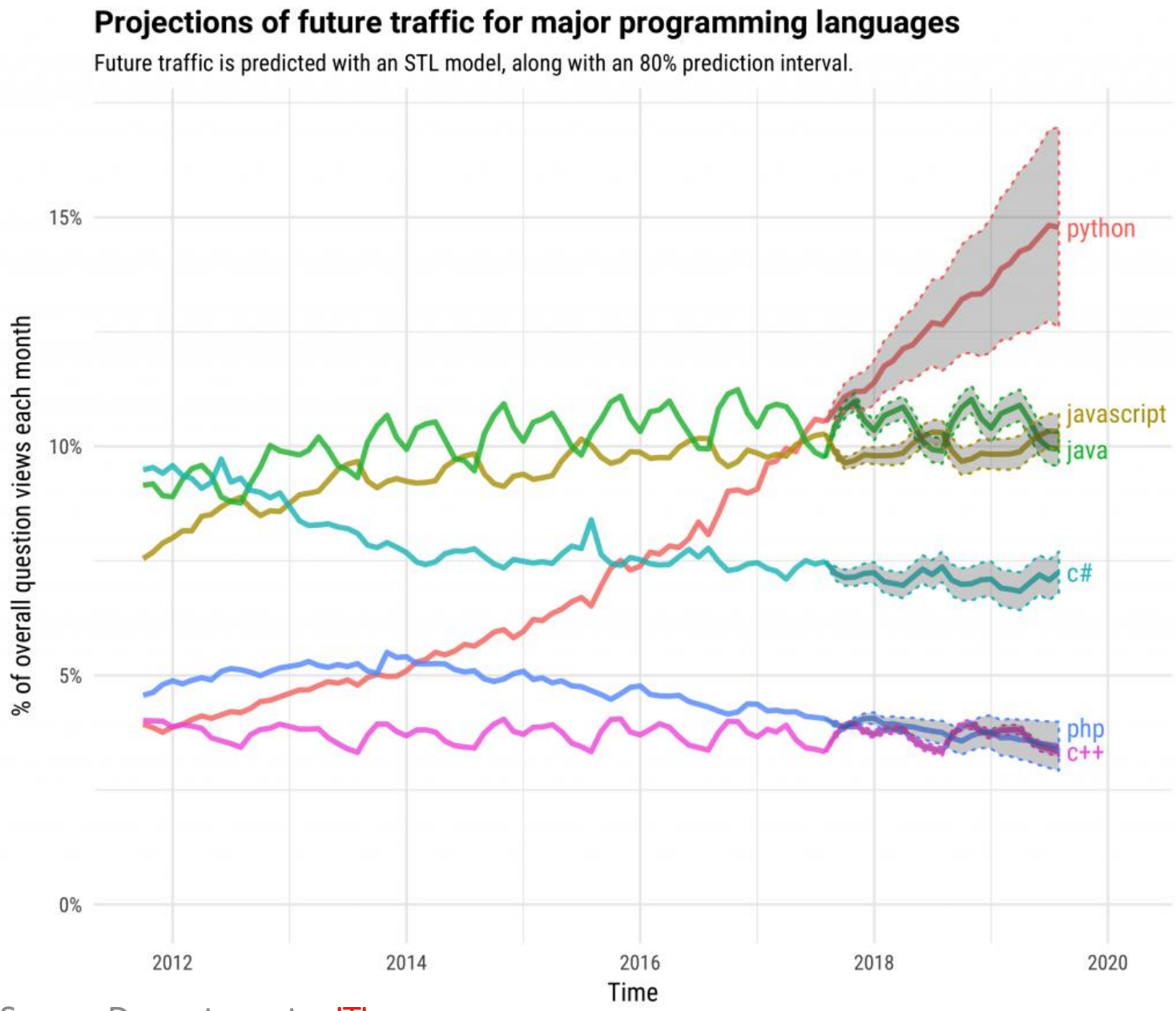
Python is designed to be highly readable. It uses English keywords frequently whereas the other languages use punctuations. It has fewer syntactical constructions than other languages.

Python is interpreted, processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL & PHP.

Python is interactive, you can actually sit a Python prompt and interact with interpreter directly to write your programs.

Python is Object-Oriented, it supports Object-Oriented style or technique of programming that encapsulates code within objects.

The Incredible Growth of Python


Projections of future traffic for major programming languages
Future traffic is predicted with an STL model, along with an 80% prediction interval.
% of overall question views each month
15%
10%
5%
0%
2012
2014
2016
2018
2020
Time
python
javascript
java
c#
php
c++

Why Python

Easy To learn

Rapid Development

General Purpose Language

Python Features

Python Features

A simple language which is easier to learn:

Python has a very simple and elegant syntax, it's much easier to read and write compared to other languages like C++, Java, C#.

Python allows you to focus on the solution rather syntax

Free and open-source:

You can freely use and distribute even for commercial use.

You can make changes to Python's source code.

Portability:

It runs seamlessly on almost all platform including Windows, Mac OS and Linux.

Python Features

Extensible and Embeddable:

Suppose an application requires high performance, You can easily combine pieces of C/C++ or other languages with Python Code.

A high-level, interpreted language:

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.

When you run Python Code, It automatically converts your code to the language your computer understands, You don't need to worry about any lower-language level operations.

Python Features

Large standard libraries to solve common tasks:

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a web serve? You can use MySQLdb library using import MySQLdb.

Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

Object-Oriented:

Everything in Python is an object. Object Oriented programming (OOP) helps you solve a complex problem intuitively.

With OOP, you are able to divide these complex problems into smaller sets by creating objects.

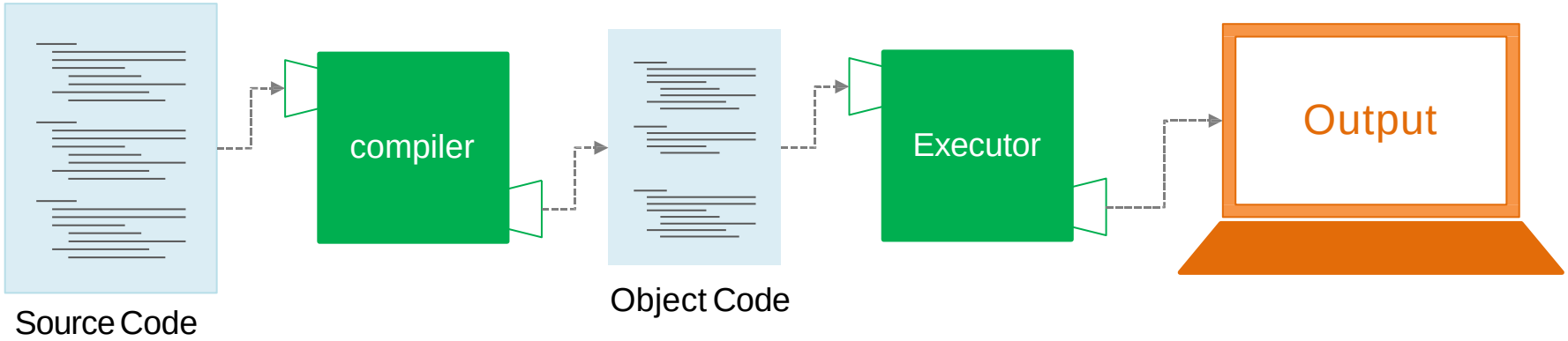
Python 2 or 3

Python 2 is the legacy, Python 3 is the future of the language

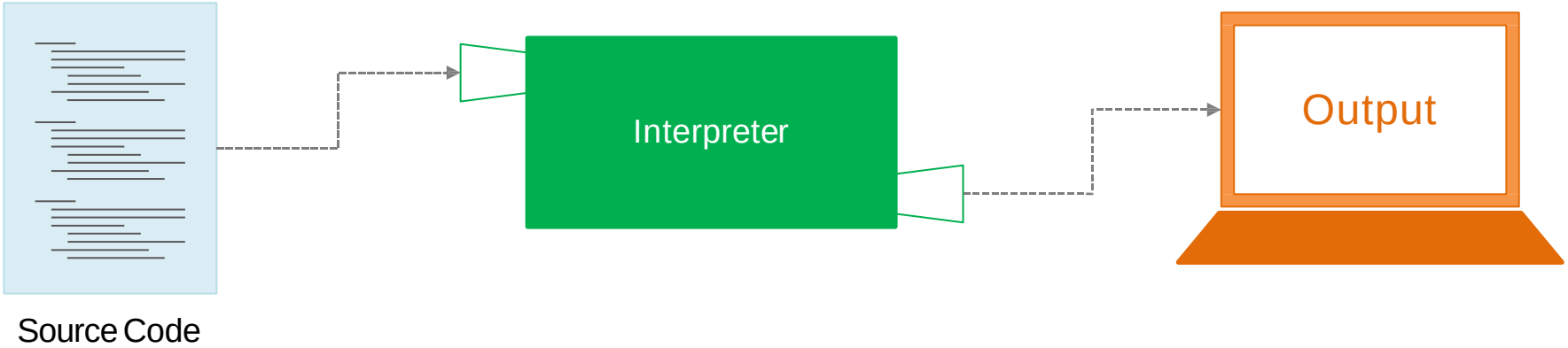
How does python work?

Compiler vs Interpreter

Compiler



Interpreter



Python is Interpreted Language

Hello World Program

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

Syntax

Python Syntax Rules

Identifiers Rules

A Python **identifier** is a name used to identify a variable, function, class, module or other object

Starts only with: a → z | A → Z | _

Can't Contain : Punctuations Characters

Can Contain: digits | a → z | A → Z | _

Python is Case Sensitive Language

Reserved Words

A Python **identifier** doesn't be one of these words

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

Line Indentations

Level 1

Level 2

```
if True:
        print("Hello, World")
else:
        print("Bye, World")
```

No ;

Just Line Indentation

Quotes …1.. 2 … 3

' ' , " " , ' ' ' ' ' ' & " " " " " " " " " "

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

Comments

```
# this is a comment
```

Variables & Data Types

Python is loosely typed language

Declare a Variable

```
Variable Identifier = Variable Value
```

```
name = "Ahmed"

age = 17

isStudent = True

age = "seventeen"
```

Data Types

- Strings
- Numbers
- Boolean
- Tuples
- Lists
- Dictionaries

```
type(variable_name)
```

Type Conversion

```
age = 17.5
int(age)          # 17
float(age)        # 17.5
str(age)          # "17.5"
```

Operators

Arithmetic

+	addition Op	`2 + 3`	`#output: 5`
-	Subtraction Op	`4 - 2`	`#output: 2`
*	Multiplication Op	`4 * 5`	`#output: 20`
/	Division Op	`16 / 5`	`#output: 3.2`
%	Modulus Op	`16 % 5`	`#output: 1`
//	Division without Fractions	`16 // 5`	`#output: 3`
**	Exponent Op	`2 ** 4`	`#output: 16`

Assignment

`=`	assign	`x = 4`	`#output: 4`
`+=`	add and assign	`x += 3`	`#output: 7`
`-=`	subtract and assign	`x -= 2`	`#output: 5`
`*=`	multiply and assign	`x *= 6`	`#output: 30`
`/=`	divide and assign	`x /= 2`	`#output: 15`
`%=`	get modulus and assign	`x %= 8`	`#output: 7`
`//=`	floor divide and assign	`x //= 3`	`#output: 2`
`**=`	get exponent and assign	`x **= 4`	`#output: 16`

Comparison

```
a <op> b
```

==	return True if a equals b
>=	return True if a equals or greater than b
<=	return True if a equals or lesser than b
!=	return True if a not equals b
<>	return True if a not equals b
>	return True if a greater than b
<	return True if a lesser than b

== Examples

When using == Python assume that:

```
True = 1, False = 0
```

```
2 == "2"                    #output: False
True == "True"              #output: False
False == 0                  #output: True
True == 1                   #output: True
True == 2                   #output: False
```

Boolean Operators

Expression (Logic Gate) Expression

Logic Gates

`and`	AND Logic Gate
`or`	OR Logic Gate
`not`	Not Logic Gate

```
True and False                    #output: False

True or False                     #output: True

not False                         #output: True

not (True == 2)                   #output: True

(False == 0) and (True == 1)      #output: True
```

Falsy Values

None, **False**, 0 , Empty collections: “”, (), [], {}

More Examples

```
2 and 1                    #output: 1
2 or 1                     #output: 2
not 4                      #output: False
not 0                      #output: True
2 and 0                    #output: 0
0 and 2                    #output: 0
"Google" and 1             #output: 1
"" and "Go"                #output: ""
False or 0                 #output: 0
```

Control Flow

Conditions & Loops

If statement

```
if (x == 2):
    print("Two")
elif (x == 3):
    print("Three")
else:
    print("others")
```

for …in

```
languages = ['JavaScript', 'Python', 'Java']
for l in languages:
        print(l)
```

```
Output:
JavaScript
Python
Java
```

Range Function

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

Examples

```
range(5)            [0,1,2,3,4]
range(0,5,1)        [0,1,2,3,4]
range(1,10,2)       [1,3,5,7,9]
```

```
for i in range(10):
    print(i)
```

0	1	2	3	4	5	6	7	8	9

while

```
dayCount = 0
while dayCount < 4:
        print("We are learning Python")
        dayCount += 1
```

```
Output:
We are learning Python
We are learning Python
We are learning Python
We are learning Python
```

DayCount
1
2
3
4

Break Statement

```
for i in range(10):
    if (i == 5):
        break
    print(i)
```

0 1 2 3 4

Continue Statement

```
for i in range(10):
    if (i == 5):
        continue
    print(i)
```

0	1	2	3	4		6	7	8	9

Else Statement

```
for i in range(10):
    if (i == 5):
        continue
    print(i)
else:
    print(10)
```

0 1 2 3 4 6 7 8 9 10

Pass Statement

```
for i in range(10):
    if (i == 5):
        pass
    print(i)
```

0	1	2	3	4	5	6	7	8	9

input Function

```
input(prompt_message)
```

Example

```
name = input("What's your Name? ");
print(name);
```

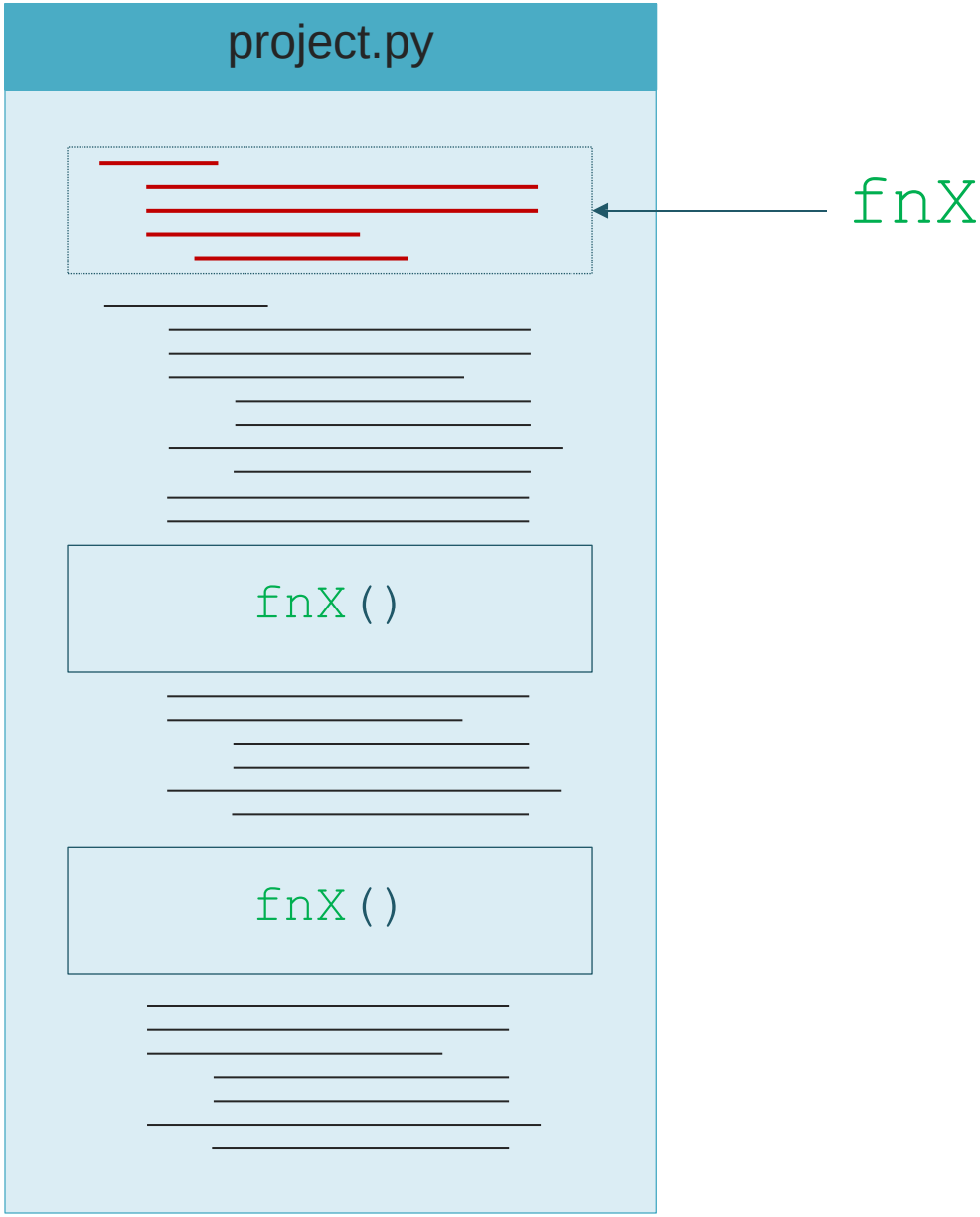
```
Output:

What's your name? Mahmoud
Mahmoud
```

Functions

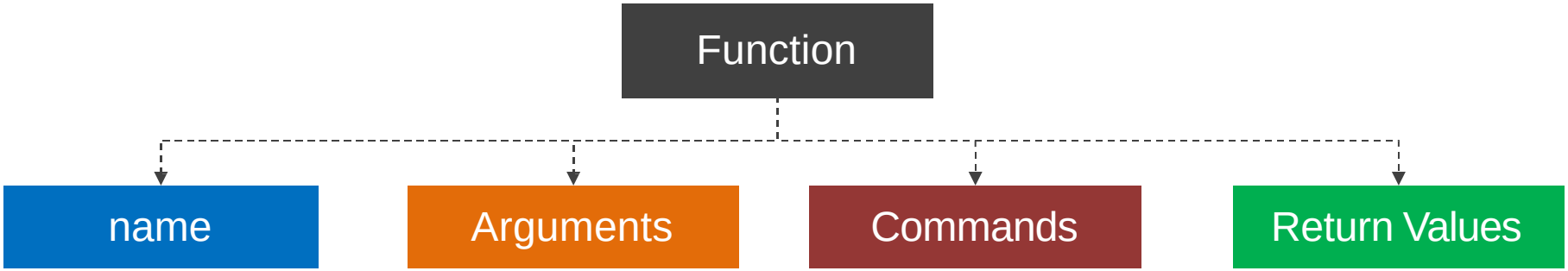
Make your code more generic

Intro



Defining

```
def fnName:
    pass
```




```
def measureTemp ( temp ):
    if temp < 37:
        return "Too Cold"
    elif temp > 37:
        return "Too Hot"
    return "Normal"
```

```
measureTemp(37)
# "Normal"
```

Default Arguments

```
def doSum(x, y = 2,z = 3):
    sum = x + y + z
    print(sum)
```

Calling It

```
doSum(2)            # output: 7
doSum(2,4)          # output: 9
doSum(2,4,10)       # output: 16
```

*arguments

```
def doSum(*args):
    sum = 0
    for i in args:
        sum += i;
    print(sum)
```

Calling It

```
doSum(2,6)          # output: 8
doSum(2,4,5,15)     # output: 26
```

**keywords

```
def doSum(**kwargs):
    for k in kwargs:
        print(kwargs[k])
```

Calling It

```
doSum(x = 2, y = 26)     # output: 2
                                  26
```

Strings

Play with Strings

How To

```
name = "Ahmed"
```

—or—

```
name = 'Ali'
```

Play !

```
name = "Ahmed "

print(name) # Ahmed

fullName = "Mohamed " + name * 3 + " Ali";

print(fullName) # Mohamed Ahmed Ahmed Ahmed Ali

nameIntro = ( "I'm " fullName );

print(nameIntro) # I'm Mohamed Ahmed Ahmed Ahmed Ali

print(name[4])    # d

print(name[1:3]) # hm

print(name[:4])   # Ahme

print(name[6])    # Index Error
```

Methods

```
name = "information technology institute"
name.capitalize() # Information Technology Institute
len(name) #32
order = "Go read info about his work info in " + name
order.replace("info", "",2)
# Go read about his work in information technology institute
digits,containDigits = "0102002932", "Tel0102002932"
digits.isDigit() # True
containDigits.isDigit() # False
```

String Formatting

```
str.format(*args,**kwargs)
```

Example

```
intro = "My Name is {0}"
intro.format('Ahmed')
# My Name is Ahmed
intro = "My Name is {1}, I work at {0}"
intro.format('ITI','Ali')
# My Name is Ali, I work at ITI
intro = "My Name is {name}, I work at {place}"
intro.format(name='Ahmed', place='ITI')
# My Name is Ahmed, I work at ITI
```

Numbers

Play with Numbers

Types

int	18
long *	503340343L
float	18.5
complex	19+4j

* Available in Python 2 only

Type Conversion

int	`int("18")`
long	`long(18.5)`
float	`float(15)`
complex	`complex(4,5)`

Play !

```
w, x, y, z = 4, 4.4, 4.6, 15

round(x)                          #output: 4

round(y)                          #output: 5

min(x,y,z)                        #output: 4.4

max(x,y,z)                        #output: 15
```

Tips and Tricks

Swap Variables

Traditional Way

```
x = 4
y = 5
temp = x
x = y
y = temp
```

Python Way

```
x,y = 4,5
x,y = y,x
```

Enhanced Print

```
print("I'm", end=" ")
print("Ahmed", end=". ")
print("I", "love", "python")
```

Output:

```
I'm Ahmed. I Love Python
```

Shorthand If

```
Do a command if this condition is true else do other command
```

Example

```
canFly = True
bird = "Dove" if canFly else "Penguin"
# bird = "Dove"
```

Thank You