



Lecture 3: Programming Python

“Python is an experiment in how much freedom programmers need. Too much freedom and nobody can read another's code; too little and expressiveness is endangered.”

Anoop M. Namboodiri
IIIT Hyderabad



- Guido van Rossum
Benevolent Dictator for Life



Why Python?

- Scripting Language
- Versatile
- Popular
- Simplicity
- Modules and Frameworks
- Data Visualization, Machine Learning, Cyber Security, Web Servers,
- Extensive Online Documentation
- Community Support



Brief History of Python

- Invented in early 90s by Guido van Rossum
- Named after Monty Python (not the snake)
- Open sourced from the beginning
- A scripting language, but is much more
- Scalable, object oriented and functional from the beginning
- Used by Google from the beginning
- Highly popular



https://docs.python.org/3/

The screenshot shows a web browser window with the address bar set to docs.python.org. The page title is "Python 3.12.1 documentation". Below the title, a welcome message states: "Welcome! This is the official documentation for Python 3.12.1." The page is organized into two columns of links. The left column includes: "What's new in Python 3.12?" (with a subtitle "or all 'What's new' documents since 2.0"), "Tutorial" (with a subtitle "start here"), "Library Reference" (with a subtitle "keep this under your pillow"), "Language Reference" (with a subtitle "describes syntax and language elements"), "Python Setup and Usage" (with a subtitle "how to use Python on different platforms"), and "Python HOWTOs" (with a subtitle "in-depth documents on specific topics"). The right column includes: "Installing Python Modules" (with a subtitle "installing from the Python Package Index & other sources"), "Distributing Python Modules" (with a subtitle "publishing modules for installation by others"), "Extending and Embedding" (with a subtitle "tutorial for C/C++ programmers"), "Python/C API" (with a subtitle "reference for C/C++ programmers"), and "FAQs" (with a subtitle "frequently asked questions (with answers!)"). At the bottom left, there is a section titled "Indices and tables:".

Python 3.12.1 documentation

Welcome! This is the official documentation for Python 3.12.1.

Parts of the documentation:

- [What's new in Python 3.12?](#)
or all "What's new" documents since 2.0
- [Tutorial](#)
start here
- [Library Reference](#)
keep this under your pillow
- [Language Reference](#)
describes syntax and language elements
- [Python Setup and Usage](#)
how to use Python on different platforms
- [Python HOWTOs](#)
in-depth documents on specific topics
- [Installing Python Modules](#)
installing from the Python Package Index & other sources
- [Distributing Python Modules](#)
publishing modules for installation by others
- [Extending and Embedding](#)
tutorial for C/C++ programmers
- [Python/C API](#)
reference for C/C++ programmers
- [FAQs](#)
frequently asked questions (with answers!)

Indices and tables:



<https://docs.python.org/3/tutorial/>

The screenshot shows a web browser window displaying the Python 3.12.1 documentation. The browser's address bar shows `docs.python.org`. The page header includes navigation links: `Python » English » 3.12.1 » 3.12.1 Documentation » The Python Tutorial`. On the right, there are links for `previous`, `next`, `modules`, and `index`. Below these, there is a `Theme` dropdown set to `Auto`, a `Quick search` input field, and a `Go` button. The main content area is titled `The Python Tutorial`. It contains three paragraphs of text. The first paragraph describes Python as an easy-to-learn, powerful programming language. The second paragraph discusses the availability of the Python interpreter and standard library. The third paragraph mentions the extensibility of Python. On the left side of the page, there is a sidebar with three sections: `Previous topic` (Changelog), `Next topic` (1. Whetting Your Appetite), and `This Page` (Report a Bug, Show Source).

Python » English » 3.12.1 » 3.12.1 Documentation » The Python Tutorial

previous | next | modules | index

Theme Auto | Quick search Go |

The Python Tutorial

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python web site, <https://www.python.org/>, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off-line as well.

Previous topic
Changelog

Next topic
1. Whetting Your Appetite

This Page
Report a Bug
Show Source



Advantages of Python

- Easy to Learn and write error-free code
 - You can start coding today
- Encourages and Insists Coherence (clean/readable code)
 - Limited ways to do a specific task (unlike perl)
 - Forces you to indent
 - Language avoid unnecessary steps (declaration, semicolons)
- Powerful (batteries included)
 - Provides an ever-growing number of powerful modules
- Provides flexibility
 - You can extend the language with additional modules
 - Can make hybrid systems
- Provides Speed
 - Can compile to portable byte code
- Widely used / growing fast



Uses of Python

- Shell tools
 - System admin tools, Command line programs
- Text processing
- Rapid prototyping and development
- Integration of modules from different languages
- Graphical user interfaces
- Database access
- Distributed programming
- Internet scripting



What Gives?

- Slower than C; like any other scripting language
 - Although efficient built-in algorithms and Data structures might offset this
- Delayed error notification
- Lack of profiling tools



Installing Python

- Pre-installed on Unix systems (Linux, Mac OSX).
- Binaries available for Windows
- Latest stable versions are 3.11.7 and 3.12.1
 - We will stick with 3.10 as it works with 3.11 and 3.12
- Several editors and IDEs
 - VIM / Emacs
 - IDLE
 - PyCharm
 - VS Code



Python Interpreter

- Just type python on the command shell (or invoke IDLE)
- Python prompts with a '>>>'

```
>>> 5 + 4 * 3
```

```
17
```

```
>>> Ctrl+D
```




Running Programs on UNIX

1. Call python program via the python interpreter

```
% python fact.py
```

2. Make a python file directly executable by

- Adding the appropriate path to your python interpreter as the first line of your script

```
#!/usr/bin/python
```

- Making the file executable

```
% chmod a+x fact.py
```

- Invoking file from Unix command line

```
% fact.py
```




Let us Jump Right In

```
#!/usr/bin/python
```

```
x = 34 - 23
```

```
# A comment
```

```
y = "Hello"
```

```
# Another one
```

```
z = 3.45
```

```
if z == 3.45 or y == "Hello":
```

```
    x = x + 1
```

```
    y = y + " World" # String concatenation
```

```
print(x)
```

```
print(y)
```




Understanding the Code

- Indentation matters to code meaning
 - Block structure indicated by indentation; starts with a **:**
- First assignment to a variable creates it
 - Variable types need not to be declared.
 - Python figures out the object/variable types on its own.
- Assignment is **=** and comparison is **==**
- For numbers, arithmetic operators are as in C
 - Special use of **+** for string
- Logical operators are words (**and**, **or**, **not**) not symbols
- The basic printing command is **print**
- Comments start with a **#**. Rest of the line is ignored



Let's look at it again

```
#!/usr/bin/python
```

```
x = 34 - 23
```

```
# A comment
```

```
y = "Hello"
```

```
# Another one
```

```
z = 3.45
```

```
if z == 3.45 or y == "Hello":
```

```
    x = x + 1
```

```
    y = y + " World"    # String concatenation
```

```
print x
```

```
print y
```



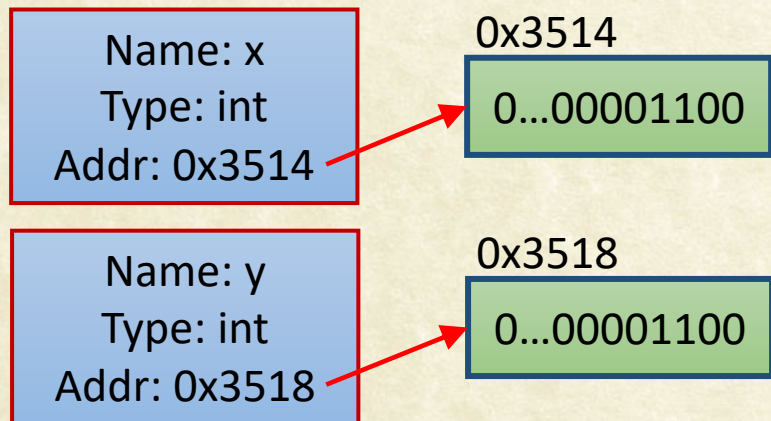

Variables in Python

- Variables are not declared; just assigned
- They are created the first time you use it
- Variables are references to objects
- Type info is with the object, not the reference
- Everything in Python is an object (even functions!!)
- The reference may change during execution
- When an object is no-longer referenced, it is deleted automatically (garbage collection)

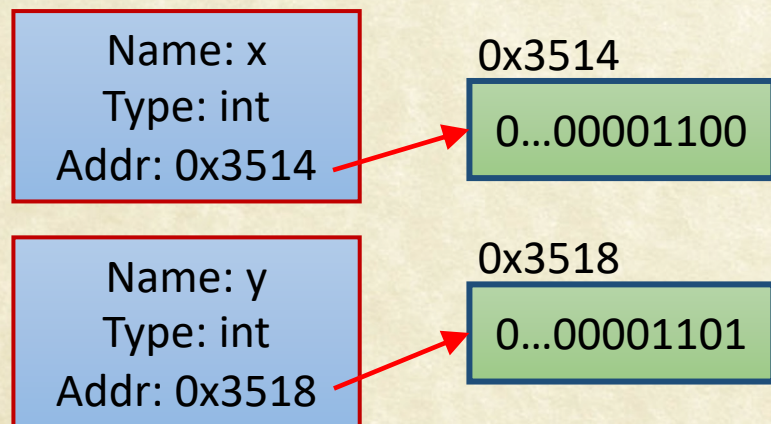


Variables: C vs. Python

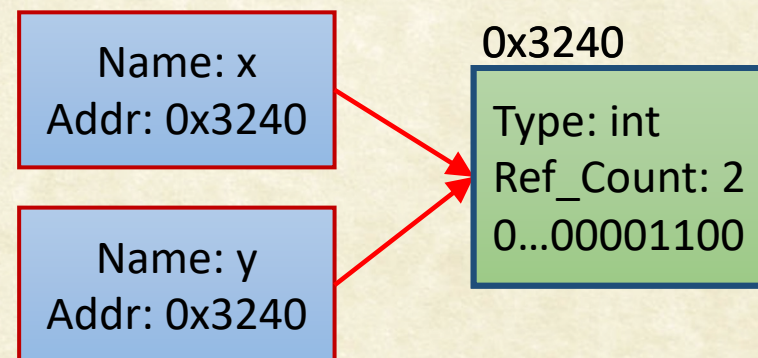
```
int x = y = 12;
```



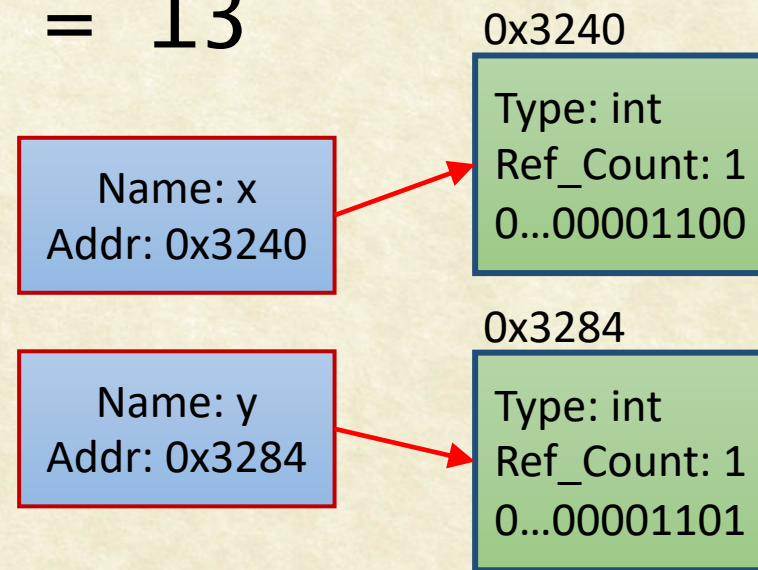
```
y = 13;
```



```
x = y = 12
```



```
y = 13
```





Numbers are Immutable

Code

```
x = 2.5
```

```
y = x
```

```
y = y+3
```

```
print (x, y)
```

What Happens

• x → 2.5

• y

• x → 2.5 y → 5.5

Output: 2.5 5.5



Numbers

- Integer: Equivalent of long in C
- Long Integer: An unbounded integer value
- Float: Equivalent to double precision in C
- Type Conversion
 - `int(x)` converts `x` to an integer
 - `float(x)` converts `x` to a floating point
- `type(var_name)` gives the type

```
>>> 132224
132224

>>> 132323 ** 2
17509376329L

>>> 3.1416
3.1416

>> int(2.0)
2

>> float(2)
2.0
```




Arithmetic Operators

- Basic arithmetic operators are the same as in C
 - ✧ **+** addition
 - ✧ **-** subtraction (and unary negation)
 - ✧ ***** multiplication
 - ✧ **/** division
 - ✧ **%** modulo division
 - ✧ **()** paranthesis
- ****** indicates power operator: **3**2** is **9**
- **//** floored quotient: **5 // 2.3** is **2.0**
- **Brackets -> Exponential -> Mult/Div -> Add/Sub**



Arithmetic Expressions: Practice

- | | |
|---------------------------|--------|
| • $3 + 2 * 5 - 1$ | • 12 |
| • $3 + 2 * 5 - 1 / 2$ | • 12.5 |
| • $3 + 2 * 5 - 1 / 2.$ | • 12.5 |
| • $3 + 2 * (5 - 1) / 2.0$ | • 7.0 |
| • $5 + 2 * 5 \% 7$ | • 8 |
| • $3 + 2 * 5 ** 2 \% 7$ | • 4 |
| • $3 + 2 * 5 // 2.8$ | • 6.0 |



Boolean Expressions

Things that are False

- The boolean value: **False**
- The numbers 0 (integer), 0.0 (float) and 0j (complex)
- The empty string ""
- The empty list [], empty dictionary {} and empty set set()

Things that are True

- The boolean value: **True**
- All non-zero numbers
- Any string containing at least one character
- A non-empty data structure



Boolean Expressions

- Any expression that has a truth value
- Comparison operators
 - Result of a comparison of two objects is a boolean
 - "<" , ">" , "==" , ">=" , "<=" , "!=" have evident meanings
 - "is" ["not"]: true if both operands refer to the same object
 - ["not"] "in": checks collection membership (more later)
 - Order of evaluation: Left to Right.
 - Comparison operators have lower precedence than Arithmetic
- Logical operators
 - The results of logical operators on boolean is a boolean
 - and, or, not : have evident meanings
 - Order of evaluation: not, and, or
 - Logical operators have lower precedence than comparisons



if: The Conditional Statement

```
if    boolean-expression:
    statements
elif  boolean-expression:
    statements
else:
    statements
```

- elif stands for else if
- Evaluates the expressions one by one until one is found to be true; then that set of statements (suite) is executed (and no other part of the if statement is executed or evaluated). If all expressions are false, the statements of the else clause, if present, is executed.



if Statement: Examples

```
if num % 10 == 0:
    print("Number is a multiple of 10")
elif num % 2:
    print("Number is odd")
else:
    print("Number is even")

if a <= 5 and c >= 10 or d == "done" and b != 5:
    print("Right Conditions")

If 8 <= Time <= 17:
    print("Working Time")
```




Let us Code: **if**

- Write a code that prints if a number is divisible by 11 or not

```
if num % 11:  
    print("num is not divisible by 11")  
else:  
    print("num is divisible by 11")
```

- Write a code that decides if a floating-point number, x, is in the open interval (a,b)

```
if a < x < b:  
    print("x is in the open interval ({a},{b})")  
else:  
    print("x is outside (a,b)")
```




Coding Practice: **if**

1. Given a circle (center, x and y , and radius r), decide if a given point, (x_1, y_1) is within the circle or not.
2. Write the code that decides if a number is an integer, a complex number or a float
3. Given two integers, write a code that decides if one is a factor of the other or not