



If you're starting out with Python and are looking for a fun and comprehensive tutorial, check out my YouTube tutorials. I have two Python tutorials. If you have no or little programming experience, I suggest you check out my [Python tutorial for beginners](#). Otherwise, if you know the basics (eg variables, functions, conditional statements, loops) and are looking for a tutorial that gets straight to the point and doesn't treat you like a beginner, check out my [Python tutorial for programmers](#).

If you enjoy this post, please spread the love by sharing this post with others.

Variables

```
1 | a = 1          # integer
2 | b = 1.1        # float
3 | c = 1 + 2j     # complex number (a + bi)
4 | d = "a"        # string
5 | e = True       # boolean (True / False)
```

Strings

```
01 | x = "Python"
02 | len(x)
03 | x[0]
04 | x[-1]
05 | x[0:3]
06 |
07 | # Formatted strings
08 | name = f"{first} {last}"
09 |
10 | # Escape sequences
11 | \" \"' \\ \n
12 |
13 | # String methods
14 | x.upper()
```

[JavaScript](#)

[Mobile](#)

[Python](#)

[React](#)

[React Native](#)

Popular Posts

[React vs. Angular: The Complete Comparison](#)

[53 Python Exercises and Questions for Beginners](#)

[React Lifecycle Methods – A Deep Dive](#)

[React file upload: proper and easy](#)

```

14 | x.upper()
15 | x.lower()
16 | x.title()
17 | x.strip()
18 | x.find("p")
19 | x.replace("a", "b")
20 | "a" in x

```

[way, with NodeJS!](#)

[4 Common Mistakes with the Repository Pattern](#)

Type Conversion

```

1 | int(x)
2 | float(x)
3 | bool(x)
4 | string(x)

```

[5 C# Collections that Every C# Developer Must Know](#)

[React Functional or Class Components: Everything you need to know](#)

Falsy Values

```

1 | 0
2 | ""
3 | []

```

Conditional Statements

```

01 | if x == 1:
02 |     print("a")
03 | elif x == 2:
04 |     print("b")
05 | else:
06 |     print("c")
07 |
08 | # Ternary operator
09 | x = "a" if n > 1 else "b"
10 |
11 | # Chaining comparison operators
12 | if 18 <= age < 65:

```

Loops

```

1 | for n in range(1, 10):
2 |     print(n)
3 |
4 | while n < 10:
5 |     print(n)
6 |     n += 1

```

Functions

```

01 | def increment(number, by=1):
02 |     return number + by
03 |
04 | # Keyword arguments
05 | increment(2, by=1)
06 |
07 | # Variable number of arguments

```

```

07 # variable number of arguments
08 def multiply(*numbers):
09     for number in numbers:
10         print number
11
12
13 multiply(1, 2, 3, 4)
14
15 # Variable number of keyword arguments
16 def save_user(**user):
17     ...
18
19
20 save_user(id=1, name="Mosh")

```

Lists

```

01 # Creating lists
02 letters = ["a", "b", "c"]
03 matrix = [[0, 1], [1, 2]]
04 zeros = [0] * 5
05 combined = zeros + letters
06 numbers = list(range(20))
07
08 # Accessing items
09 letters = ["a", "b", "c", "d"]
10 letters[0] # "a"
11 letters[-1] # "d"
12
13 # Slicing lists
14 letters[0:3] # "a", "b", "c"
15 letters[:3] # "a", "b", "c"
16 letters[0:] # "a", "b", "c", "d"
17 letters[:] # "a", "b", "c", "d"
18 letters[:2] # "a", "c"
19 letters[::-1] # "d", "c", "b", "a"
20
21 # Unpacking
22 first, second, *other = letters
23
24 # Looping over lists
25 for letter in letters:
26     ...
27
28 for index, letter in enumerate(letters):
29     ...
30
31 # Adding items
32 letters.append("e")
33 letters.insert(0, "-")
34
35 # Removing items
36 letters.pop()
37 letters.pop(0)
38 letters.remove("b")
39 del letters[0:3]
40

```

```

41 # Finding items
42 if "f" in letters:
43     letters.index("f")
44
45 # Sorting lists
46 letters.sort()
47 letters.sort(reverse=True)
48
49 # Custom sorting
50 items = [
51     ("Product1", 10),
52     ("Product2", 9),
53     ("Product3", 11)
54 ]
55
56 items.sort(key=lambda item: item[1])
57
58 # Map and filter
59 prices = list(map(lambda item: item[1], items))
60 expensive_items = list(filter(lambda item: item[1]
61
62 # List comprehensions
63 prices = [item[1] for item in items]
64 expensive_items = [item for item in items if item[1]
65
66 # Zip function
67 list1 = [1, 2, 3]
68 list2 = [10, 20, 30]
69 combined = list(zip(list1, list2)) # [(1, 10), (

```

Tuples

```

01 point = (1, 2, 3)
02 point[0:2] # (1, 2)
03 x, y, z = point
04 if 10 in point:
05     ...
06
07 # Swapping variables
08 x = 10
09 y = 11
10 x, y = y, x

```

Arrays

```

1 from array import array
2
3 numbers = array("i", [1, 2, 3])

```

Sets

```

01 first = {1, 2, 3, 4}
02 second = {1, 5}
03

```

```

04 first | second # {1, 2, 3, 4, 5}
05 first & second # {1}
06 first - second # {2, 3, 4}
07 first ^ second # {2, 3, 4, 5}
08
09 if 1 in first:
10     ...

```

Dictionaries

```

01 point = {"x": 1, "y": 2}
02 point = dict(x=1, y=2)
03 point["z"] = 3
04 if "a" in point:
05     ...
06 point.get("a", 0) # 0
07 del point["x"]
08 for key, value in point.items():
09     ...
10
11 # Dictionary comprehensions
12 values = {x: x * 2 for x in range(5)}

```

Generator Expressions

```

1 values = (x * 2 for x in range(10000))
2 len(values) # Error
3 for x in values:

```

Unpacking Operator

```

1 first = [1, 2, 3]
2 second = [4, 5, 6]
3 combined = [*first, "a", *second]
4
5 first = {"x": 1}
6 second = {"y": 2}
7 combined = {**first, **second}

```

Exceptions

```

01 # Handling Exceptions
02 try:
03     ...[SEP]
04 except (ValueError, ZeroDivisionError):
05     ...
06 else:
07     # no exceptions raised
08 finally:
09     # cleanup code
10
11 # Raising exceptions
12 if x < 1: [SEP]
13     raise ValueError("...")

```

```

14
15 # The with statement
16 with open("file.txt") as file:
17     ...

```

Classes

```

01 # Creating classes
02 class Point:
03     def __init__(self, x, y):
04         self.x = x
05         self.y = y
06
07     def draw(self):
08         ...
09
10 # Instance vs class attributes
11 class Point:
12     default_color = "red"
13
14     def __init__(self, x, y):
15         self.x = x
16
17 # Instance vs class methods
18 class Point:
19     def draw(self):
20         ...
21
22     @classmethod
23     def zero(cls):
24         return cls(0, 0)
25
26
27 # Magic methods
28     __str__()
29     __eq__()
30     __cmp__()
31     ...
32
33 # Private members
34 class Point:
35     def __init__(self, x):
36         self.__x = x
37
38
39 # Properties
40 class Point:
41     def __init__(self, x):
42         self.__x = x
43
44     @property
45     def x(self):
46         return self.__x
47
48     @property.setter
49     def x(self, value):
50         self.__x = value

```

```
50 self.__x = value
51
52 # Inheritance
53 class FileStream(Stream):
54     def open(self):
55         super().open()
56     ...
57
58 # Multiple inheritance
59 class FlyingFish(Flyer, Swimmer):
60     ...
61
62 # Abstract base classes
63 from abc import ABC, abstractmethod
64
65 class Stream(ABC):
66     @abstractmethod
67     def read(self):
68         pass
69
70 # Named tuples
71 from collections import namedtuple
72
73 Point = namedtuple("Point", ["x", "y"])
74 point = Point(x=1, y=2)
```



Mosh

Hi! My name is Mosh Hamedani. I'm a software engineer with two decades of experience and I've taught over three million people how to code or how to become professional software engineers through my YouTube channel and online courses. It's my mission to make software engineering accessible to everyone.



Share this:



Related

[53 Python Exercises and Questions for Beginners](#)

October 22, 2018
In "Python"

[My Brand New Python Tutorial \(6 Hours\)](#)

February 18, 2019
In "Python"

[4 Common Mistakes with the Repository Pattern](#)

April 3, 2017
In "C# / .NET"