Python Course

Python beginner

WHO AM I?

- Cloud Architecture
- Devops Master
- Python Devloper
- Linux Sysadmin
-
-
-
-
- **Create by**: **Hadi Tayanloo**
- **Phone** : **+98-912-8387233**
- **Linkedin**: linkedin.com/in/htayanloo/

Course title

- Python Intro
- Python Get Started
- Python Syntax
- Python Variables
- Python Numbers
- Python Casting
- Python Strings
- Python Operators
- Python Lists
- Python Tuples
- Python Sets
- Python Dictionaries
- Python If...Else
- Python While Loops
- Python For Loops
- Python Functions
- Python Lambda
- Python Arrays
- Python Classes/Objects
- Python Inheritance
- Python Iterators
- Python Modules
- Python Dates
- Python JSON
- Python RegEx
- Python PIP
- Python Try...Except

Python Syntax

Execute Python Syntax:

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

Python Syntax

Execute Python Syntax:

C:\Users\Your Name>python myfile.py

Python Syntax

Python Indentations

Good

```
if 5 > 2:
  print("Five is greater than two!")
```

Bad:

```
if 5 > 2:
print("Five is greater than two!")
```

Python Syntax

Comments

```
#This is a comment.
print("Hello, World!")
```

```
"""This is a
multiline docstring."""
print("Hello, World!")
```

Python Variables

- **Creating Variables**
-
- x = 5
- y = "John"
- print(x)
- print(y)

Python Variables

- **Creating Variables**
-
- x = 4 # x is of type int
- x = "Sally" # x is now of type str
- print(x)

Python Variables

- x = "Python is "
- y = "awesome"
- z = x + y
- print(z)

Python Variables

- x = 5
- y = 10
- print(x + y)

Python Numbers

- Python Numbers
-
- x = 1   # int
- y = 2.8 # float
-

Python Numbers

- Python Numbers
-
- x = 1
- y = 35656222554887711
- z = -3255522
-
- print(type(x))
- print(type(y))
- print(type(z))
-

Python Casting

- **Integers:**
  - x = **int**(1)   # x will be 1
  - y = **int**(2.8) # y will be 2
  - z = **int**("3") # z will be 3
  -
- **Floats:**
  - x = **float**(1)   # x will be 1.0
  - y = **float**(2.8)   # y will be 2.8
  - z = **float**("3")   # z will be 3.0
  - w = **float**("4.2") # w will be 4.2
  -
- **Strings:**
  - x = **str**("s1") # x will be 's1'
  - y = **str**(2)   # y will be '2'
  - z = **str**(3.0) # z will be '3.0'

Python Strings

- a = "Hello, World!"
- print(a[1])
-
- b = "Hello, World!"
- print(b[2:5])
-
- a = " Hello, World! "
- print(a.strip()) # returns "Hello, World!"
-

Python Strings

- a = "Hello, World!"
- print(len(a))
-
- b = "Hello, World!"
- print(b.lower())
-
- a = " Hello, World! "
- print(a.upper())
-

Python Strings

- a = "Hello, World!"
- print(a.replace("H", "J"))
-
- b = "Hello, World!"
- print(a.split(",")) # returns ['Hello', ' World!']
-
-

Python Strings

- print("Enter your name:")
- x = input()
- print("Hello, ", x)

Python 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

Python Assignment Operators

Operator	Example	Same As
Assign(=)	x=5	x=5
Add and Assign(+=)	X += 3	X= x+3
Subtract and Assign(-=)	X -= 3	X =x - 3
Multiply and Assign(*=)	X *= 3	X = x * 3
Divide and Assign(/=)	X /= 3	X= x / 3
Exponent and Assign(**=)	X %= 3	X= x % 3
Floor-Divide and Assign(//=)	X //=3	X = x //3
Exponentiation(**)	X **= 3	X = x ** 3
Binary AND(&)	X &= 3	X = x & 3
Binary OR(|)	X |= 3	X = x | 3
Binary XOR(^)	X ^= 3	X = x ^ 3
Binary Left-Shift(<<)	X >>= 3	X = x>> 3
Binary Right-Shift(>>)	X <<=3	X = x<< 3

Python Operators << >>

```
>>> bin(0b1111 << 1)
'0b11110'

>>> bin(0b1111 << 2)
'0b111100'

>>> bin(0b1111 << 3)
'0b1111000'

>>> bin(0b1111 << 4)
'0b11110000'
```

Python Comparison Operators

Operator	name	Same As
==	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

Python Logical Operators

Operator	name	Same As
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Python Identity Operators

Operator	name	Same As
is	Returns true if both variables are the same object	x is y
is not	Returns true if both variables are not the same object	x is not y

Python Membership Operators

Operator	name	Same As
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Python Bitwise Operators

Operator	name	Same As
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

Python Lists

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.

Python Lists

- **List**
  - thislist = ["apple", "banana", "cherry"]
  - print(thislist)
  -
- **Access Items**
  - thislist = ["apple", "banana", "cherry"]
  - print(thislist[1])
  -
- **Change Item Value**
  - thislist = ["apple", "banana", "cherry"]
  - thislist[1] = "blackcurrant"
  - print(thislist)
  -

Python List

- **Loop Through a List**
  - thislist = ["apple", "banana", "cherry"]
  - for x in thislist:
  - print(x)
- **Check if Item Exists**
  - thislist = ["apple", "banana", "cherry"]
  - if "apple" in thislist:
  - print("Yes, 'apple' is in the fruits list")
- **List Length**
  - thislist = ["apple", "banana", "cherry"]
  - print(len(thislist))
-

Python List

- **Add Items**
  - thislist = ["apple", "banana", "cherry"]
  - thislist.append("orange")
  - **print**(thislist)
- **insert**
  - thislist = ["apple", "banana", "cherry"]
  - thislist.insert(1, "orange")
  - **print**(thislist)
- **Remove Item**
  - thislist = ["apple", "banana", "cherry"]
  - thislist.remove("banana")
  - **print**(thislist)
-

Python List

- **Add pop**
  - sthislist = ["apple", "banana", "cherry"]
  - thislist.pop()
  - **print**(thislist)
- **Del Item**
  - thislist = ["apple", "banana", "cherry"]
  - **del** thislist[0]
  - **print**(thislist)
- **Del list**
  - thislist = ["apple", "banana", "cherry"]
  - **del** thislist
-

Python Tuple

- **Tuple**
  - thistuple = ("apple", "banana", "cherry")
  - **print**(thistuple)
- **Access Tuple Items**
  - thistuple = ("apple", "banana", "cherry")
  - **print**(thistuple[1])
- **Change Tuple Values**
  - thistuple = ("apple", "banana", "cherry")
  - thistuple[1] = "blackcurrant"
  - # The values will remain the same:
  - **print**(thistuple)
- **Loop Through a Tuple**
  - thistuple = ("apple", "banana", "cherry")
  - **for** x **in** thistuple:
  - **print**(x)

Python Tuple

- **Check if Item Exists**
  - thistuple = ("apple", "banana", "cherry")
  - **if** "apple" **in** thistuple:
  - **print**("Yes, 'apple' is in the fruits tuple")
- **Tuple Length**
-
  - thistuple = ("apple", "banana", "cherry")
  - **print**(**len**(thistuple))
- **Remove Items**
-
  - thistuple = ("apple", "banana", "cherry")
  - **del** thistuple
  - **print**(thistuple) #this will raise an error because the tuple no longer exists
- **The tuple() Constructor**
-
  - thistuple = **tuple**(("apple", "banana", "cherry")) # note the double round-brackets
  - **print**(thistuple)

Python Sets

- **Set**
  - thisset = {"apple", "banana", "cherry"}
  - print(thisset)
- **Access Items**
  - thisset = {"apple", "banana", "cherry"}
  - for x in thisset:
  - print(x)
-
- **Change Items**
  - Once a set is created, you cannot change its items, but you can add new items.

Python Sets

- **Add Items**
-
  - thisset = {"apple", "banana", "cherry"}
  - thisset.add("orange")
  - print(thisset)
  -
- **update Items**
  - thisset = {"apple", "banana", "cherry"}
  - thisset.update(["orange", "mango", "grapes"])
  - print(thisset)
-
- **Get the Length of a Set**
  - thisset = {"apple", "banana", "cherry"}
  -
  - print(len(thisset))
-

Python Sets

- **Remove Item or discard**
  - thisset = {"apple", "banana", "cherry"}
  - thisset.remove("banana")
  - print(thisset)
- **pop Items**
  - thisset = {"apple", "banana", "cherry"}
  - x = thisset.pop()
  - print(x)
  - print(thisset)
- **clear**
  - thisset = {"apple", "banana", "cherry"}
  - thisset.clear()
  - print(thisset)

Python Sets

- **del set**
  - thisset = {"apple", "banana", "cherry"}
  -
  - del thisset
  -
  - print(thisset)
- **The set() Constructor**
  - thisset = set(("apple", "banana", "cherry")) # note the double round-brackets
  - print(thisset)
-

Python Dictionaries

- **Dictionary**
  - thisdict = {
  - "brand": "Ford",
  - "model": "Mustang",
  - "year": 1964
  - }
  - **print**(thisdict)
- **Accessing Items**
  - x = thisdict["model"]
  - or
  - x = thisdict.get("model")
- **Change Values**
  - thisdict = {
  - "brand": "Ford",
  - "model": "Mustang",
  - "year": 1964
  - }
  - thisdict["year"] = 2018
-

Python Dictionaries

- **Loop Through a Dictionary**
  - for x in thisdict:
  - print(x)
- **Accessing Items values**
  - for x in thisdict.values():
  - print(x)
- **Access items**
  - for x, y in thisdict.items():
  - print(x, y)

Python Dictionaries

- **Check if Key Exists**
-
  - thisdict = {
  - "brand": "Ford",
  - "model": "Mustang",
  - "year": 1964
  - }
  - if "model" in thisdict:
  - print("Yes, 'model' is one of the keys in the thisdict dictionary")
  -
- **Dictionary Length**
-
  - print(len(thisdict))
- **Adding Items**
-
  - thisdict = {
  - "brand": "Ford",
  - "model": "Mustang",
  - "year": 1964
  - }
  - thisdict["color"] = "red"
  - print(thisdict)

Python Dictionaries

- **Removing Items**
-
  - thisdict = {
  - "brand": "Ford",
  - "model": "Mustang",
  - "year": 1964
  - }
  - thisdict.pop("model")
  - **print**(thisdict)
  -
- **popitem**
-
  - thisdict = {
  - "brand": "Ford",
  - "model": "Mustang",
  - "year": 1964
  - }
  - thisdict.popitem()
  - **print**(thisdict)
- **del Items**
  - thisdict = {
  - "brand": "Ford",
  - "model": "Mustang",
  - "year": 1964
  - }
  - **del** thisdict["model"]
  - **print**(thisdict)

Python If...Else

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

Python If...Else

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

Python If...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")
```

Python If...Else

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

Python While Loops

- while **loops**
- for **loops**
-

Python While Loops

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

Python While Loops

- **The break Statement**
  - i = 1
  - while i < 6:
  - print(i)
  - if i == 3:
  - break
  - i += 1

Python While Loops

- **The continue Statement**
-

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

Python For Loops

- **For Loop**
  - fruits = ["apple", "banana", "cherry"]
  - for x in fruits:
  - print(x)
-
- **Looping Through a String**
  - for x in "banana":
  - print(x)
- **The break Statement**
-
  - fruits = ["apple", "banana", "cherry"]
  - for x in fruits:
  - print(x)
  - if x == "banana":
  - break

Python For Loops

- **The continue Statement**
-
  - fruits = ["apple", "banana", "cherry"]
  - for x in fruits:
  - if x == "banana":
  - continue
  - print(x)
- **The range() Function**
  - for x in range(6):
  - print(x)
  -
  - for x in range(2, 30, 3):
  - print(x)
-

Python For Loops

- **Else in For Loop**
-
  - for x in range(6):
  - print(x)
  - else:
  - print("Finally finished!")
-
- **Nested Loops**
-
  - adj = ["red", "big", "tasty"]
  - fruits = ["apple", "banana", "cherry"]
  -
  - for x in adj:
  - for y in fruits:
  - print(x, y)

Python Functions

- **Creating a Function**
  - `def my_function():`
  - `  print("Hello from a function")`

- **Calling a Function**

  - `def my_function():`
  - `  print("Hello from a function")`
  -
  - `my_function()`

- **Parameters**
-
  - `def my_function(x):`
  - `  return 5 * x`
  -
  - `print(my_function(3))`
  - `print(my_function(5))`
  - `print(my_function(9))`

Python Functions

- **Default Parameter Value**
  - `def my_function(country = "Norway"):`
  - `  print("I am from " + country)`
  -
  - `my_function("Sweden")`
  - `my_function("India")`
  - `my_function()`
  - `my_function("Brazil")`
-
- **Passing a List as a Parameter**
  -
  - `def my_function(food):`
  - `  for x in food:`
  - `    print(x)`
  -
  - `fruits = ["apple", "banana", "cherry"]`
  -
  - `my_function(fruits)`
-
- **Return Values**
  - `def my_function(x):`
  - `  return 5 * x`
  -
  - `print(my_function(3))`
  - `print(my_function(5))`
  - `print(my_function(9))`

Python Lambda

Python Arrays

Python Classes/Objects

Python Inheritance

Python Iterators

Python Modules

Python Dates

Python JSON

Python RegEx

Python PIP

Python Try...Except