6-mar

March 6, 2025

0.1 tuple

[1]:
```
t=()
print(t)
```

```
()
```

[3]:
```
print(type(t))
```

```
<class 'tuple'>
```

[5]:
```
t1=(10,20,30,40,40)
t1
```

[5]:
```
(10, 20, 30, 40, 40)
```

[7]:
```
len(t1)
```

[7]:
```
5
```

[11]:
```
t1.count(10)
```

[11]:
```
1
```

[13]:
```
t1.count(40)
```

[13]:
```
2
```

[15]:
```
t1
```

[15]:
```
(10, 20, 30, 40, 40)
```

[17]:
```
t1.index(20)
```

[17]:
```
1
```

[19]:
```
l5=['a','b','c','d','e']
```

[21]:
```
l5
```

[21]:
```
['a', 'b', 'c', 'd', 'e']
```

[23]:
```
l5[1]=10 #list is mutable =hashable
```

[25]:
```
t2=(100,3.4,'nit',True,1+2j,[1,2,3],(5,6,7))# multiple data types are allowed
t2
```

[25]:
```
(100, 3.4, 'nit', True, (1+2j), [1, 2, 3], (5, 6, 7))
```

[29]:
```
print(t)
print(t1)
print(t2)
```

```
()
(10, 20, 30, 40, 40)
(100, 3.4, 'nit', True, (1+2j), [1, 2, 3], (5, 6, 7))
```

[31]:
```
t1
```

[31]:
```
(10, 20, 30, 40, 40)
```

[33]:
```
t1[0]
```

[33]:
```
10
```

[35]:
```
t1[0]=1000 # tuple is immutable =inhashable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[35], line 1
----> 1 t1[0]=1000

TypeError: 'tuple' object does not support item assignment
```

[37]:
```
icici=(45678,"cizps7789",332000,98765)
icici
```

[37]:
```
(45678, 'cizps7789', 332000, 98765)
```

[39]:
```
icici[0]=1234
icici
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[39], line 1
```

```
----> 1 icici[0]=1234
      2 icici

TypeError: 'tuple' object does not support item assignment
```

[43]:
```
t1
```

[43]:
```
(10, 20, 30, 40, 40)
```

[46]:
```
t4=t1*3
t4
```

[46]:
```
(10, 20, 30, 40, 40, 10, 20, 30, 40, 40, 10, 20, 30, 40, 40)
```

[48]:
```
t4[:]
```

[48]:
```
(10, 20, 30, 40, 40, 10, 20, 30, 40, 40, 10, 20, 30, 40, 40)
```

[50]:
```
t2
```

[50]:
```
(100, 3.4, 'nit', True, (1+2j), [1, 2, 3], (5, 6, 7))
```

[52]:
```
t1
```

[52]:
```
(10, 20, 30, 40, 40)
```

[54]:
```
t1[:7]
```

[54]:
```
(10, 20, 30, 40, 40)
```

[56]:
```
t1[2:]
```

[56]:
```
(30, 40, 40)
```

[58]:
```
t1[0:10:2]
```

[58]:
```
(10, 30, 40)
```

[60]:
```
t1,remove(30)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[60], line 1
----> 1 t1,remove(30)
```

```
NameError: name 'remove' is not defined
```

[62]:
```
t1.add(30)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[62], line 1
----> 1 t1.add(30)

AttributeError: 'tuple' object has no attribute 'add'
```

[64]:
```
t2.index(100)
```

[64]: 0

[66]:
```
t2.index('nit')
```

[66]: 2

[68]:
```
x=5
y=2
print(x//y)
```

2

0.2 set

[70]:
```
s={}
s
```

[70]: {}

[72]:
```
type(s)
```

[72]: dict

0.3 practice

[76]:
```
tup1=() # empty tuple
```

[82]:
```
tup2=(10,30,60) # int
```

[84]:
```
tup3=(10.77,30.66,60.89) #float
```

[90]:
```
tup4=('one','two','three','four') #string
```

[92]:
```
tup5=('asif',25,(50,100),(150,90)) # nested tuples
```

[94]:
```
tup6=(100,'asif',17.765) # mixed data types
```

[98]:
```
tup7=('asif',25,[50,100],[150,90],{'john','david'},(99,22,33))
```

[100]:
```
len(tup7)
```

[100]: 6

[102]:
```
tup2[0]
```

[102]: 10

[104]:
```
tup4[0]
```

[104]: 'one'

[106]:
```
tup4[0][0]
```

[106]: 'o'

[108]:
```
tup4[-1]
```

[108]: 'four'

[110]:
```
tup5[-1]
```

[110]: (150, 90)

[114]:
```
mytuple=('one','two','three','four','five','six','seven','eight')
```

[116]:
```
mytuple[0:3]
```

[116]: ('one', 'two', 'three')

[118]:
```
mytuple[2:5]
```

[118]: ('three', 'four', 'five')

[120]:
```
mytuple[:3]
```

[120]: ('one', 'two', 'three')

[122]:
```
mytuple[:2]
```

[122]: ('one', 'two')

[124]:
```
mytuple[-3:]
```

[124]:
```
('six', 'seven', 'eight')
```

[126]:
```
mytuple[-2:]
```

[126]:
```
('seven', 'eight')
```

[128]:
```
mytuple[-1]
```

[128]:
```
'eight'
```

[130]:
```
mytuple[:]
```

[130]:
```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

[159]:
```
mytuple
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[159], line 1
----> 1 mytuple

NameError: name 'mytuple' is not defined
```

[161]:
```
del mytuple[0]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[161], line 1
----> 1 del mytuple[0]

NameError: name 'mytuple' is not defined
```

[163]:
```
mytuple[0]=1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[163], line 1
----> 1 mytuple[0]=1

NameError: name 'mytuple' is not defined
```

[169]:
```
mytuple=('one','two','three','four','five','six','seven','eight')
```

[171]:
```
del mytuple
```

[175]:
```
mytuple=('one','two','three','four','five','six','seven','eight')
```

[177]:
```
mytuple
```

[177]:
```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

[179]:
```
for i in mytuple:
    print(i)
```

```
one
two
three
four
five
six
seven
eight
```

[181]:
```
for i in enumerate(mytuple):
    print(i)
```

```
(0, 'one')
(1, 'two')
(2, 'three')
(3, 'four')
(4, 'five')
(5, 'six')
(6, 'seven')
(7, 'eight')
```

[183]:
```
mytuple
```

[183]:
```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

[185]:
```
'one' in mytuple
```

[185]:
```
True
```

[187]:
```
'ten' in mytuple
```

[187]:
```
False
```

[189]:
```
if 'three' in mytuple:
    print('three is present in the tuple')
else:
```

```
    print('three id not present in the tuple')
```

```
three is present in the tuple
```

[191]:
```
mytuple
```

[191]:
```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

[193]:
```
mytuple.index('one')
```

[193]:
```
0
```

[199]:
```
mytuple.index('five')
```

[199]:
```
4
```

[203]:
```
mytuple
```

[203]:
```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

[205]:
```
mytuple.index('three')
```

[205]:
```
2
```

[207]:
```
mytuple2=(43,67,99,12,6,90,67)
```

[209]:
```
sorted(mytuple2)
```

[209]:
```
[6, 12, 43, 67, 67, 90, 99]
```

[211]:
```
sorted(mytuple,reverse=True)
```

[211]:
```
['two', 'three', 'six', 'seven', 'one', 'four', 'five', 'eight']
```

[213]:
```
a = 5
b = 2
sum_result = a + b
difference_result = a - b
product_result = a * b
division_result = a / b
# Comparison operators
is_equal = a == b
is_not_equal = a != b
is_greater_than = a > b
is_less_than = a < b
# Logical operators
logical_and = (a > 0) and (b > 0)
```

```
logical_or = (a > 0) or (b > 0)
logical_not = not (a > 0)
print("Arithmetic Operators:")
print("Sum:", sum_result)
print("Difference:", difference_result)
print("Product:", product_result)
print("Division:", division_result)
print("\nComparison Operators:")
print("Is Equal:", is_equal)
print("Is Not Equal:", is_not_equal)
print("Is Greater Than:", is_greater_than)
print("Is Less Than:", is_less_than)
print("\nLogical Operators:")
print("Logical AND:", logical_and)
print("Logical OR:", logical_or)
print("Logical NOT:", logical_not)
```

```
Arithmetic Operators:
Sum: 7
Difference: 3
Product: 10
Division: 2.5

Comparison Operators:
Is Equal: False
Is Not Equal: True
Is Greater Than: True
Is Less Than: False

Logical Operators:
Logical AND: True
Logical OR: True
Logical NOT: False
```

[215]:
```
# Unary minus (-) operator
number = 10
result = -number
print("Unary Minus Operator:", result)
# Unary plus (+) operator
result = +number
print("Unary Plus Operator:", result)
# Logical NOT (!) operator
is_true = True
logical_not_result = not is_true
print("Logical NOT Operator:", logical_not_result)
```

```
Unary Minus Operator: -10
Unary Plus Operator: 10
```

```
Logical NOT Operator: False
```

[217]:
```python
# Binary addition (+) operator
a = 5
b = 3
addition_result = a + b
print("Binary Addition Operator:", addition_result)
# Binary multiplication (*) operator
multiplication_result = a * b
print("Binary Multiplication Operator:", multiplication_result)
# Logical AND (&) operator
bitwise_and_result = a & b
print("Bitwise AND Operator:", bitwise_and_result)
```

```
Binary Addition Operator: 8
Binary Multiplication Operator: 15
Bitwise AND Operator: 1
```

[219]:
```python
# Example expression
result = 5 + 3 * 2 / 2 - (4 % 3) ** 2
print("Result:", result)
```

```
Result: 7.0
```

[221]:
```python
# Arithmetic operators
a = 10
b = 3
# Addition
addition_result = a + b
# Subtraction
subtraction_result = a - b
# Multiplication
multiplication_result = a * b
# Division
division_result = a / b
# Modulus
modulus_result = a % b
# Exponentiation
exponentiation_result = a ** b
print("Addition Result:", addition_result)
print("Subtraction Result:", subtraction_result)
print("Multiplication Result:", multiplication_result)
print("Division Result:", division_result)
print("Modulus Result:", modulus_result)
print("Exponentiation Result:", exponentiation_result)
```

```
Addition Result: 13
Subtraction Result: 7
```

```
Multiplication Result: 30
Division Result: 3.3333333333333335
Modulus Result: 1
Exponentiation Result: 1000
```

[223]:
```python
# Modulus operator example
number = 15
divisor = 7
# Calculate remainder using modulus
remainder = number % divisor
print(f"The remainder of {number} divided by {divisor} is: {remainder}")
```

```
The remainder of 15 divided by 7 is: 1
```

[225]:
```python
# Example with == operator
list1 = [1, 2, 3]
list2 = [1, 2, 3]
# == compares values
result1 = list1 == list2
# Example with is operator
list3 = [1, 2, 3]
list4 = [1, 2, 3]
# is checks if objects refer to the same memory location
result2 = list3 is list4
print("Using == Operator:", result1)
print("Using is Operator:", result2)
```

```
Using == Operator: True
Using is Operator: False
```

[235]:
```python
# Example with and operator
num1 = 5
num2 = 10
# and returns True if both conditions are True
result1 = (num1 > 0) and (num2 > 0)
# Example with or operator
num3 = -5
num4 = 10
# or returns True if at least one condition is True
result2 = (num3 > 0) or (num4 > 0)
# Example with not operator
flag = True
# not returns the opposite of the given condition
result3 = not flag
print("Using and Operator:", result1)
print("Using or Operator:", result2)
print("Using not Operator:", result3)
```

```
Using and Operator: True
Using or Operator: True
Using not Operator: False
```

[237]:
```python
num1 = 5 # 0101 in binary
num2 = 3 # 0011 in binary
result_and = num1 & num2 # 0101 & 0011 = 0001 (1 in decimal)
# Example with bitwise OR (|) operator
result_or = num1 | num2 # 0101 | 0011 = 0111 (7 in decimal)
# Example with bitwise XOR (^) operator
result_xor = num1 ^ num2 # 0101 ^ 0011 = 0110 (6 in decimal)
# Example with left shift (<<) operator
result_left_shift = num1 << 1 # 0101 << 1 = 1010 (10 in decimal)
# Example with right shift (>>) operator
result_right_shift = num1 >> 1 # 0101 >> 1 = 0010 (2 in decimal)
print("Bitwise AND:", result_and)
print("Bitwise OR:", result_or)
print("Bitwise XOR:", result_xor)
print("Left Shift:", result_left_shift)
print("Right Shift:", result_right_shift)
```

```
Bitwise AND: 1
Bitwise OR: 7
Bitwise XOR: 6
Left Shift: 10
Right Shift: 2
```

[239]:
```python
# Example with the 'in' operator
fruits = ['apple', 'banana', 'orange']
# Check if 'banana' is in the list
is_banana_in_list = 'banana' in fruits
# Check if 'grape' is in the list
is_grape_in_list = 'grape' in fruits
# Example with the 'not in' operator
# Check if 'watermelon' is not in the list
is_watermelon_not_in_list = 'watermelon' not in fruits
# Check if 'orange' is not in the list
is_orange_not_in_list = 'orange' not in fruits
print("'banana' in fruits:", is_banana_in_list)
print("'grape' in fruits:", is_grape_in_list)
print("'watermelon' not in fruits:", is_watermelon_not_in_list)
print("'orange' not in fruits:", is_orange_not_in_list)
```

```
'banana' in fruits: True
'grape' in fruits: False
'watermelon' not in fruits: True
'orange' not in fruits: False
```

[243]:
```python
# Example with the 'is' operator
x = [1, 2, 3]
y = [1, 2, 3]
z = x
# Check if x and y refer to the same object
are_x_and_y_same = x is y
# Check if x and z refer to the same object
are_x_and_z_same = x is z
# Example with the 'is not' operator
# Check if x and y do not refer to the same object
are_x_and_y_not_same = x is not y
# Check if x and z do not refer to the same object
are_x_and_z_not_same = x is not z
print("x is y:", are_x_and_y_same)
print("x is z:", are_x_and_z_same)
print("x is not y:", are_x_and_y_not_same)
print("x is not z:", are_x_and_z_not_same)
```

```
x is y: False
x is z: True
x is not y: True
x is not z: False
```

[245]:
```python
# Example of the ternary conditional operator
temperature = 25
weather = "Sunny" if temperature > 20 else "Cloudy"
print("Weather today:", weather)
```

```
Weather today: Sunny
```

[247]:
```python
# Example of using comparison operators
x = 5
y = 10
# Less than
result_lt = x < y
# Greater than
result_gt = x > y
# Less than or equal to
result_lte = x <= y
# Greater than or equal to
result_gte = x >= y
# Not equal to
result_ne = x != y
# Equal to
result_eq = x == y
# Output the results
print("x < y:", result_lt)
```

```
print("x > y:", result_gt)
print("x <= y:", result_lte)
print("x >= y:", result_gte)
print("x != y:", result_ne)
print("x == y:", result_eq)
```

```
x < y: True
x > y: False
x <= y: True
x >= y: False
x != y: True
x == y: False
```

[249]:
```
# Example of using assignment operators
x = 5
# Simple assignment
y = x
print("y (simple assignment):", y)
# Compound assignment (add and assign)
x += 3
print("x (after x += 3):", x)
# Compound assignment (subtract and assign)
x -= 2
print("x (after x -= 2):", x)
# Compound assignment (multiply and assign)
x *= 4
print("x (after x *= 4):", x)
# Compound assignment (divide and assign)
x /= 2
print("x (after x /= 2):", x)
```

```
y (simple assignment): 5
x (after x += 3): 8
x (after x -= 2): 6
x (after x *= 4): 24
x (after x /= 2): 12.0
```

[251]:
```
# Example with strings
string_example = "Python"
char_to_check = "t"
# Check if a character is present in the string
is_present = char_to_check in string_example
print(f"Is '{char_to_check}' present in '{string_example}'? {is_present}")
# Example with lists
list_example = [1, 2, 3, 4, 5]
element_to_check = 3
# Check if an element is present in the list
```

```
is_present = element_to_check in list_example
print(f"Is {element_to_check} present in {list_example}? {is_present}")
```

```
Is 't' present in 'Python'? True
Is 3 present in [1, 2, 3, 4, 5]? True
```

[253]:
```
# Example with strings
string_example = "PythonProgramming"
# Extract a substring using slicing
substring = string_example[6:16] # From index 6 to 15
print("Substring:", substring)
# Example with lists
list_example = [1, 2, 3, 4, 5, 6, 7, 8, 9]
# Extract a sublist using slicing
sublist = list_example[2:7] # From index 2 to 6
print("Sublist:", sublist)
```

```
Substring: Programmin
Sublist: [3, 4, 5, 6, 7]
```

[255]:
```
# Example
base = 2
exponent = 3
# Calculate the result of exponentiation
result = base ** exponent
print("Result:", result)
```

```
Result: 8
```

[257]:
```
# Example
dividend = 10
divisor = 3
# Perform floor division
result = dividend // divisor
print("Result:", result)
```

```
Result: 3
```

[259]:
```
# Example
num1 = 5
# Add 3 to num1 using +=
num1 += 3
print("Result:", num1)
```

```
Result: 8
```

[261]:
```
# Example
is_python_fun = True
```

```
# Use not to negate the boolean value
not_python_fun = not is_python_fun
# Display the results
print("Original Value:", is_python_fun)
print("Negated Value:", not_python_fun)
```

```
Original Value: True
Negated Value: False
```

[263]:
```
my_tuple=(1,2,3)
print('tuple:',my_tuple)
```

```
tuple: (1, 2, 3)
```

[265]:
```
result=(5+3)*2
print("result:",result)
```

```
result: 16
```

[269]:
```
result1=10//3
print(result1)
```

```
3
```

[271]:
```
result2=10/3
print(result2)
```

```
3.3333333333333335
```

[273]:
```
remainder=10%3
print(remainder)
```

```
1
```

[275]:
```
r=2+3*4
r
```

[275]: 14

[277]:
```
r=(2+3)*4
r
```

[277]: 20

[279]:
```
n1='alice'
n2='bob'
print(n1<n2)
```

```
True
```

[285]:
```
age=30
is_adult=age==18
```

[287]:
```
x=None
is_empty=x is None
```

[289]:
```
r=5+2*3
r
```

[289]:
```
11
```

[291]:
```
is_student=True
is_adult=False
enrolled= is_student and is_adult
can_register=is_student or is_adult
```

[293]:
```
registered=(not is_adult)
```

[295]:
```
r=2*(3+4)
r
```

[295]:
```
14
```

[ ]:
```

```