PYTHON

Ayyappan Murugesan
Phone: 7010774329
WhatsApp: 8870483557

Programming Language

- System Level Programming Language
  - Assembly
  - C
  - C++

- High Level Programming Language
  - Java

- Script Level Programming Language
  - TCL
  - Bash
  - Dos

- General Purpose Programming Language
  - JavaScript
  - Python

Python

- Python is a high level and multi paradigm programming language designed by Guido van Rossum, a dutch programmer

- All the features as conventional programming languages such as C, C++ and Java have

Why Is Python So Popular?

- Python Supports Multiple Programming Paradigms
- Python Has Large Set Of Library and Tools
- Python Has a Vast Community Support
- Python is Designed For Better Code Readability
- Python Contains Fewer Lines Of Codes

Python in AI

- **Machine Learning**

  PyML, PyBrain, scikit-learn, MDP Toolkit, GraphLab Create, MIPy etc.

- **General AI**

  pyDatalog, AIMA, EasyAI, SimpleAI etc.

- **Neural Networks**

  PyAnn, pyrenn, ffnet, neurolab etc.

- **Natural Language & Text Processing**

  Quepy, NLTK, gensim

Python in BigData

1. Less is More
2. Python’s Compatibility with Hadoop
3. Ease of Learning
4. Powerful Packages
   - NumPy – Scientific Computing
   - Pandas – Data Analysis Tool
   - Scipy – Scientific and technical computing
5. Data Visualization

Python in Networking

- Python programming language is used to read, write and configure routers and switches and perform other networking automation tasks in a cost effective and secure manner.
- **PySNMP** – Cross Platform Pure Python SNMP
- **Ansible** – Simplest way to automate apps and IT infrastructure
- **Netmiko** – Multi-vendor library to simplify Paramiko SSH connections to network devices.

Organizations Using Python Language

- NASA
- Google
- Walt Disney Feature Animation
- AlphaGene, Inc.
- Red Hat
- Nokia
- IBM

Websites Developed Using Python

- Youtube
- Quora
- Instagram
- Pinterest
- Spotify
- Flipkart
- Slack
- Uber
- Cloudera
- Zenefits

Why Should I Choose Python To Learn Programming?

- Nonrestrictive Programming Syntax
- No Explicit Declaration.
- State Of The Art OOP Support.
- Powerful Debugging.

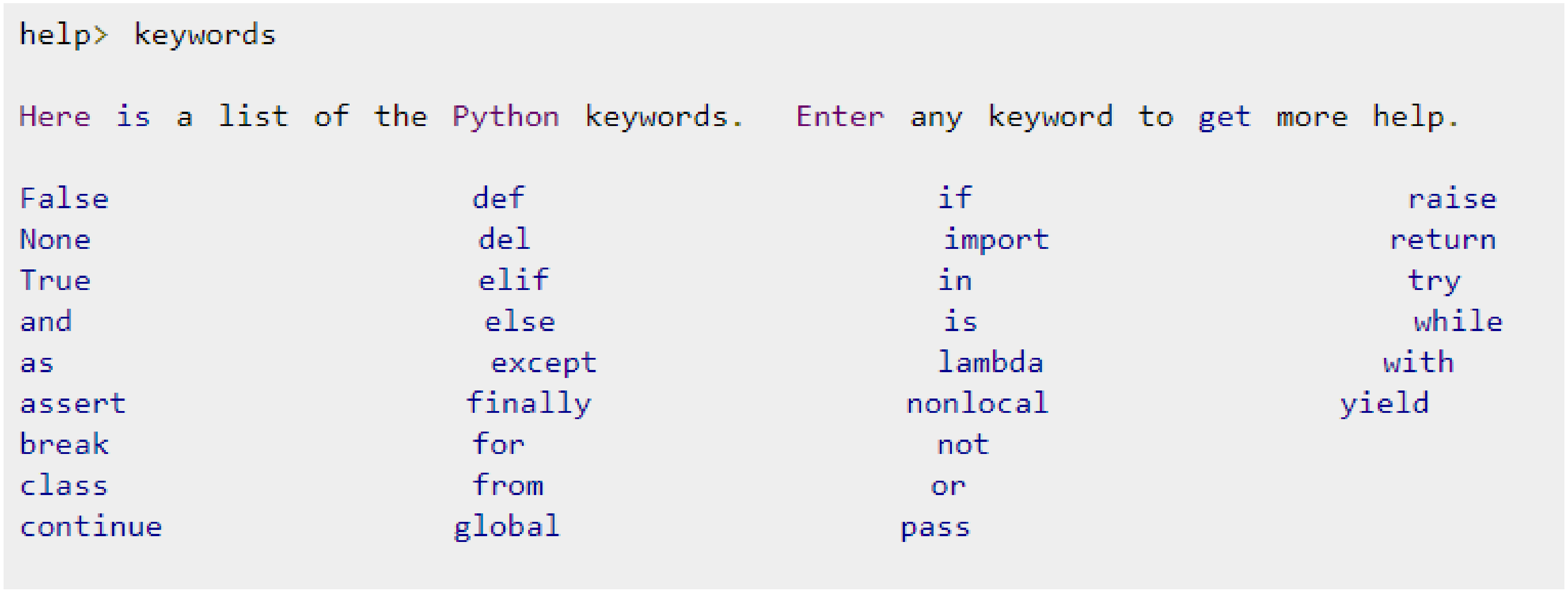
Python Versions

- Python 2.7 – For Begin
- Python 3.6 – Consistent

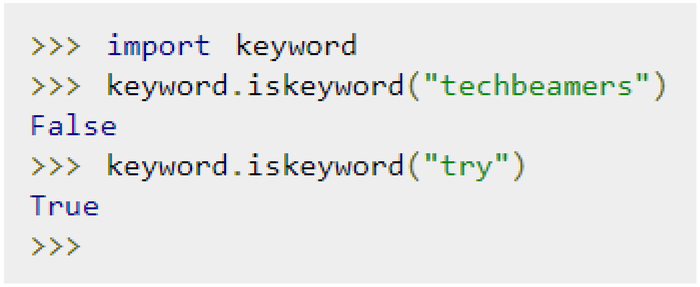
Installation

- Direct Download Executable
- Use YUM / APT-GET

Keywords



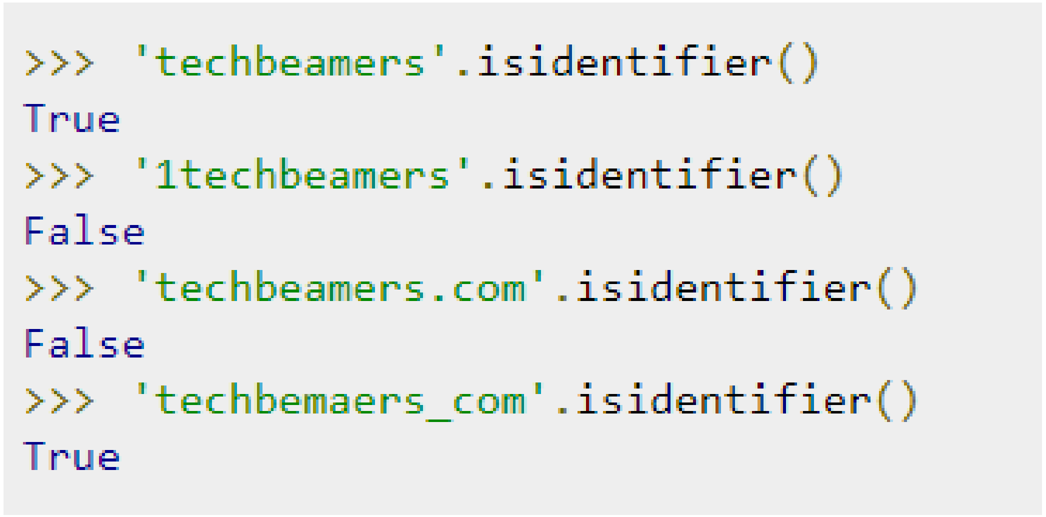
```
help> keywords

Here is a list of the Python keywords.  Enter any keyword to get more help.

False               def                 if                  raise
None                del                 import              return
True                elif                in                  try
and                 else                is                  while
as                  except              lambda              with
assert              finally             nonlocal            yield
break               for                 not
class               from                or
continue            global              pass
```



```
>>> import keyword
>>> keyword.iskeyword("techbeamers")
False
>>> keyword.iskeyword("try")
True
>>>
```

Identifiers

- shapeClass
- Shape_1
- shape@1
- True
- 11ab



```
>>> 'techbeamers'.isidentifier()
True
>>> '1techbeamers'.isidentifier()
False
>>> 'techbeamers.com'.isidentifier()
False
>>> 'techbemaers_com'.isidentifier()
True
```

Variables

```
test = 10
```

```
>>> test = 10
>>> id(test)
1716585200
>>> test = 11
>>> id(test)
1716585232
>>>
```

```
 ___________                                              ________
|           |        Reference                    *                *
|    test   |______________________\  *       10       *
|           |                                 /  *                     *
|___________|                                *________*
Variable  Name                                   Object
```

```
>>> test = 10
>>> type(test)
<class 'int'>
>>> test = 'techbeamers'
>>> type(test)
<class 'str'>
>>> test = {'Python', 'C', 'C++'}
>>> type(test)
<class 'set'>
>>>
```

Statements

- Expression

```
>>> ((10 + 2) * 100 / 5 - 200)
40.0
```

- Simple Assignment Statement

```
# Syntax
variable = expression
# LHS <=> RHS
```

- Augmented Assignment Statement

```
x += y
```

- Multi-Line Statement In Python

```
# Initializing a list using the multi-line statement
>>> my_list = [1, \
... 2, 3\
... ,4,5 \
... ]
>>> print(my_list)
[1, 2, 3, 4, 5]
```

Cond..

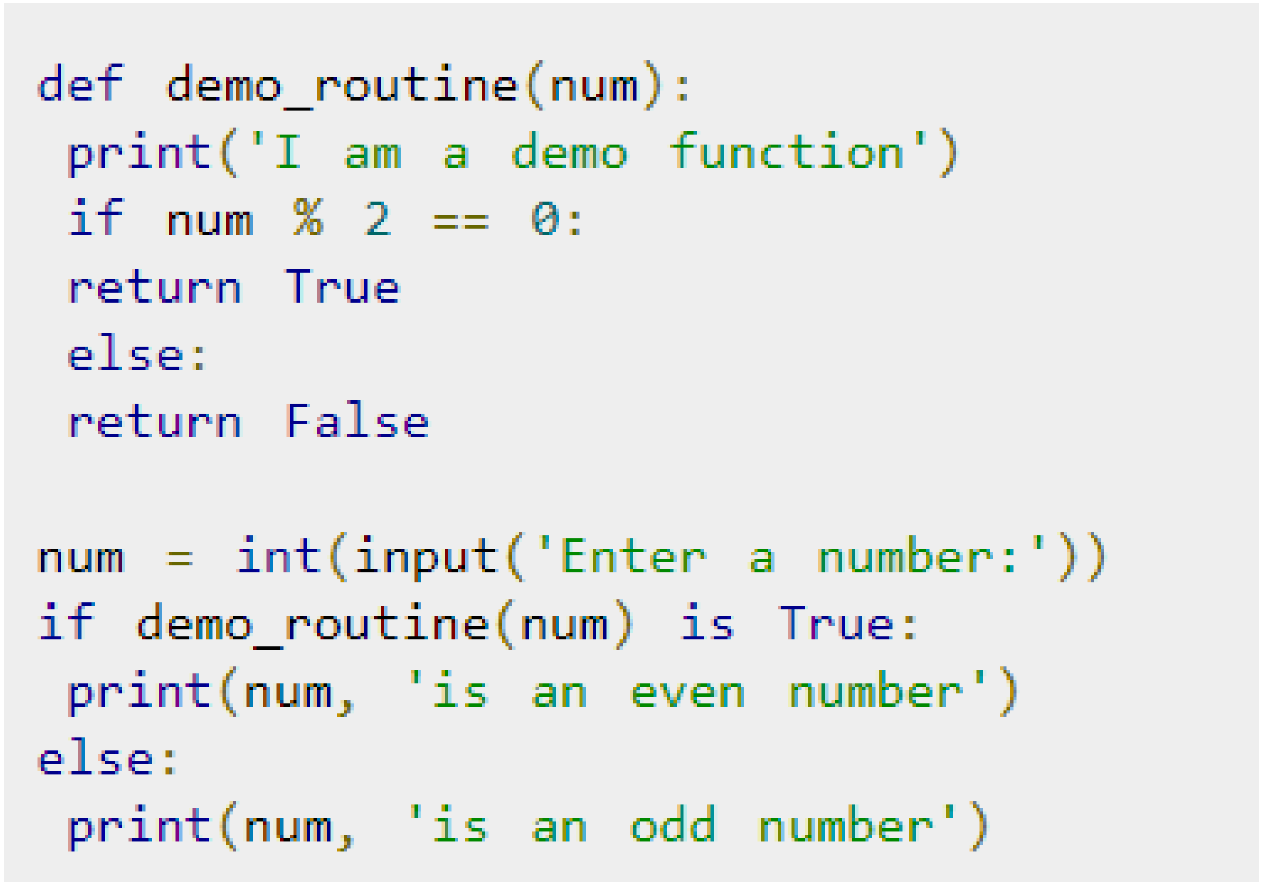
- Implicit Line Continuation

```
>>> result = (10 + 100
... * 5 - 5
... / 100 + 10
... )
>>> print(result)
519.95
```

Another Example

```
>>> subjects = [
... 'Maths',
... 'English',
... 'Science'
... ]
>>> print(subjects)
['Maths', 'English', 'Science']
>>> type(subjects)
<class 'list'>
```

Indentation



```python
def demo_routine(num):
 print('I am a demo function')
 if num % 2 == 0:
 return True
 else:
 return False

num = int(input('Enter a number:'))
if demo_routine(num) is True:
 print(num, 'is an even number')
else:
 print(num, 'is an odd number')
```

Comments

```
# Good code is self-documenting.

print("Learn Python Step by Step!")
```

```
# To Learn any language you must follow the below rules.
# 1. Know the basic syntax, data types, control structures and conditional
statements.
# 2. Learn error handling and file I/O.
# 3. Read about advanced data structures.
# 4. Write functions and follow OOPs concepts.

def main():
     print("Let's start to learn Python.")
...
```

```
def theFunction():
     '''
This function demonstrate the use of docstring in Python.
     '''
     print("Python docstrings are not comments.")

print("\nJust printing the docstring value...")
print(theFunction.__doc__)
```

DataTypes

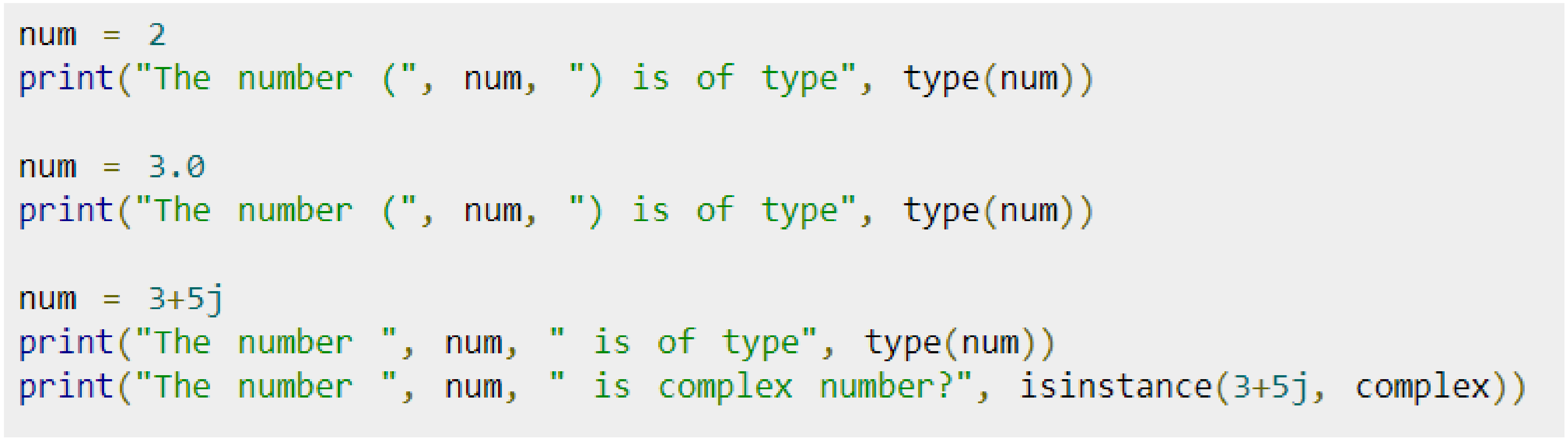
- Boolean
- Number
- String
- List
- Tuple
- Set
- Dictionary

Boolean

A Boolean is such a data type that almost every programming language has, and so is Python. Boolean in Python can have two values – **True or False**. These values are constants and can be used to

1. assign
2. compare Boolean values.

Number

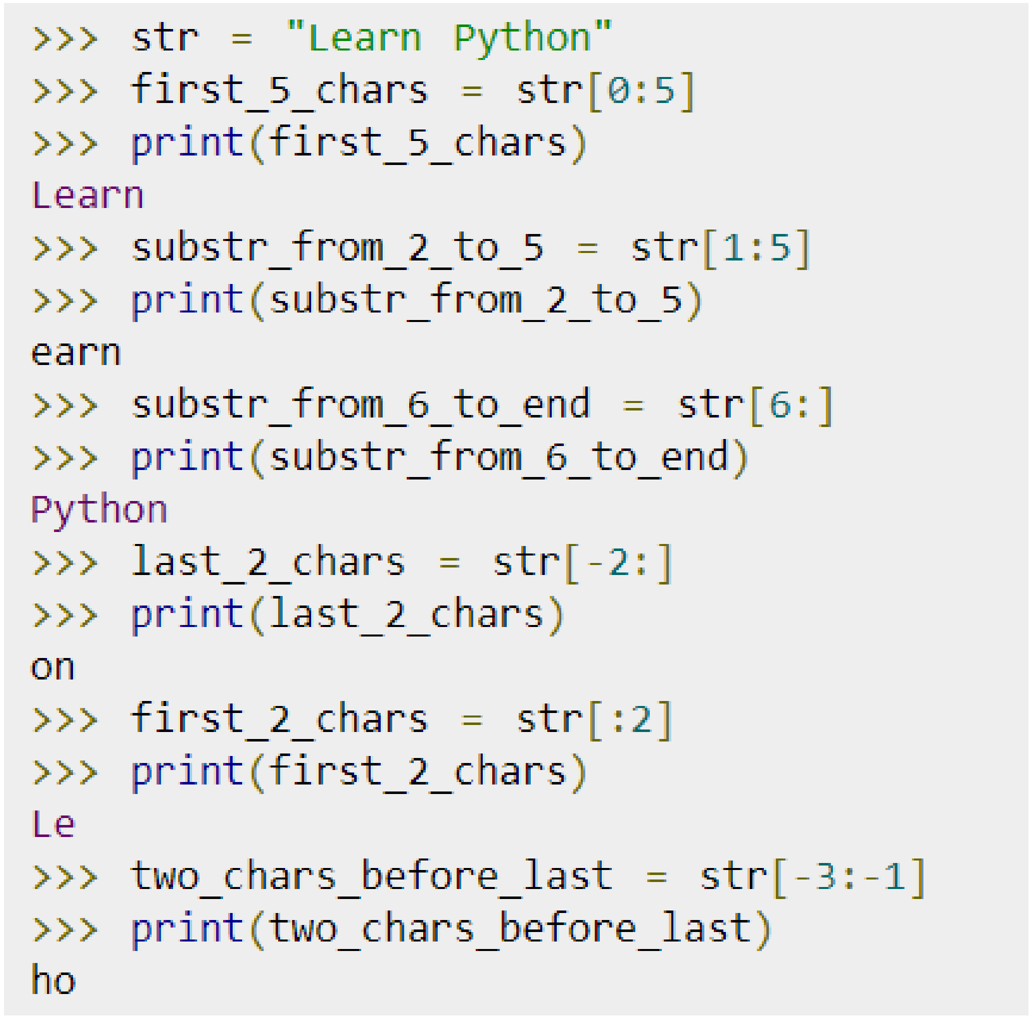


```python
num = 2
print("The number (", num, ") is of type", type(num))

num = 3.0
print("The number (", num, ") is of type", type(num))

num = 3+5j
print("The number ", num, " is of type", type(num))
print("The number ", num, " is complex number?", isinstance(3+5j, complex))
```

String

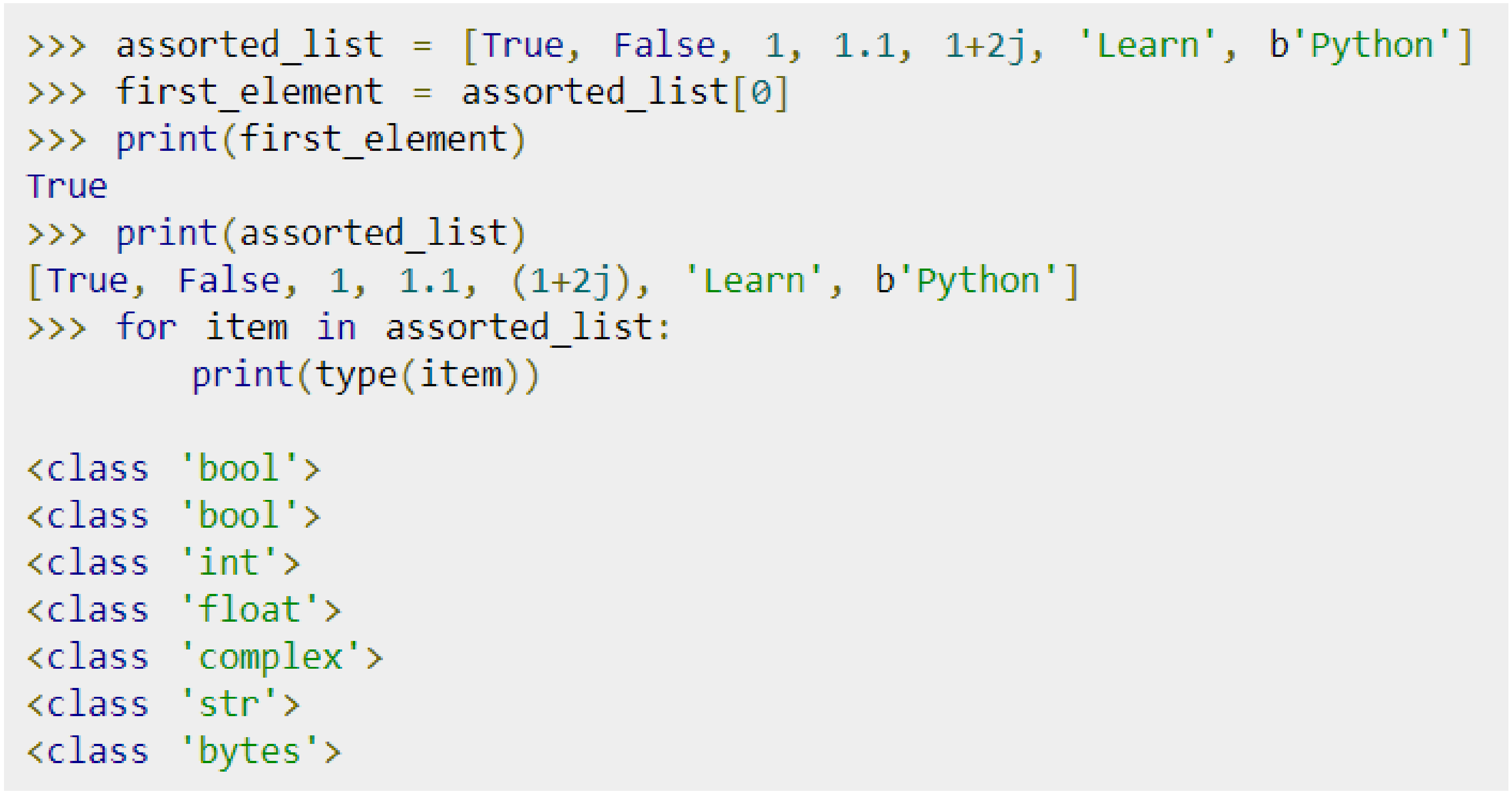
```
>>> str = 'A string wrapped in single quotes'
>>> str
'A string wrapped in single quotes'
>>> str = "A string enclosed within double quotes"
>>> str
'A string enclosed within double quotes'
>>> str = """A multiline string
starts and ends with
a triple quotation mark."""
>>> str
'A multiline string\nstarts and ends with\na triple quotation mark.'
```

Sub String Operation



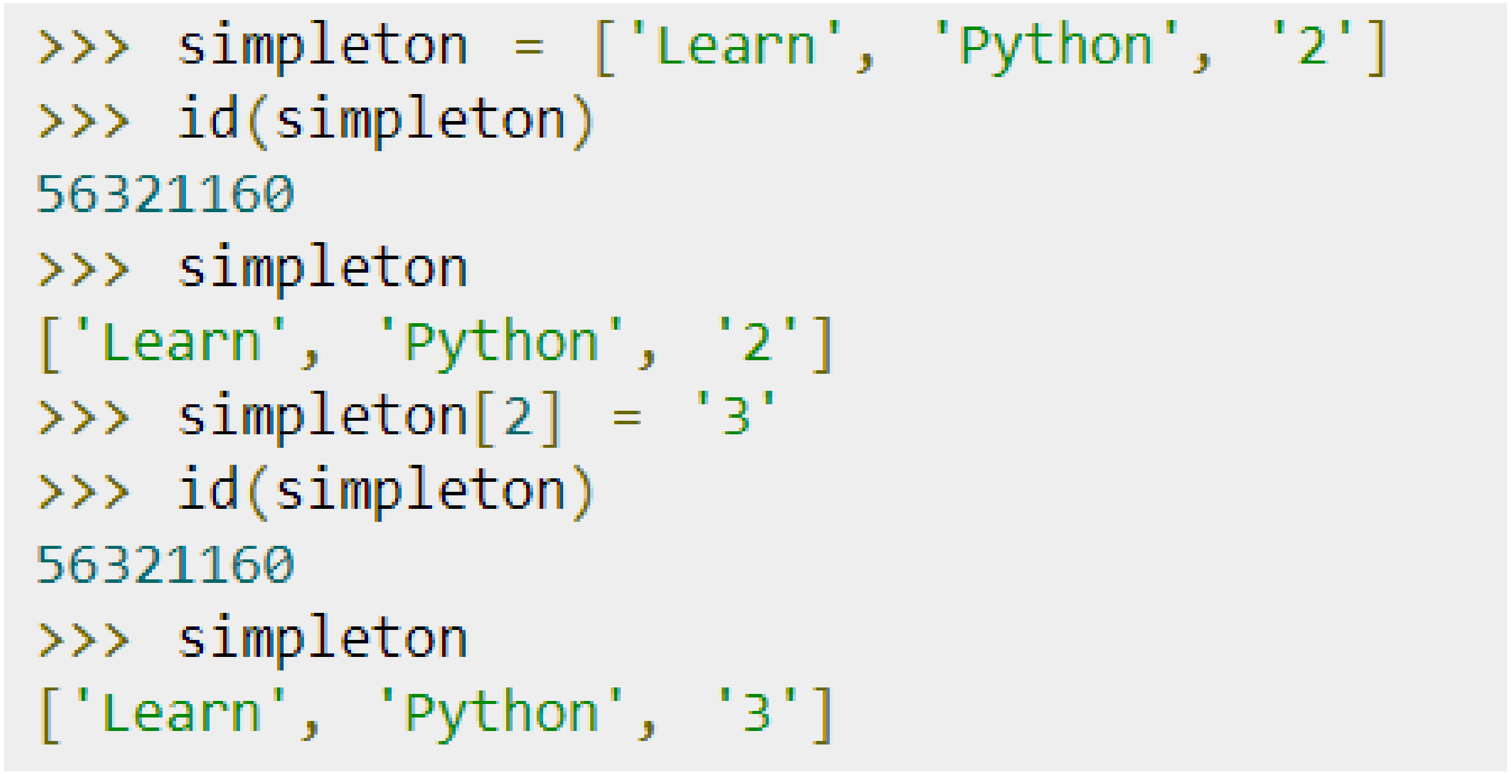
```
>>> str = "Learn Python"
>>> first_5_chars = str[0:5]
>>> print(first_5_chars)
Learn
>>> substr_from_2_to_5 = str[1:5]
>>> print(substr_from_2_to_5)
earn
>>> substr_from_6_to_end = str[6:]
>>> print(substr_from_6_to_end)
Python
>>> last_2_chars = str[-2:]
>>> print(last_2_chars)
on
>>> first_2_chars = str[:2]
>>> print(first_2_chars)
Le
>>> two_chars_before_last = str[-3:-1]
>>> print(two_chars_before_last)
ho
```

List



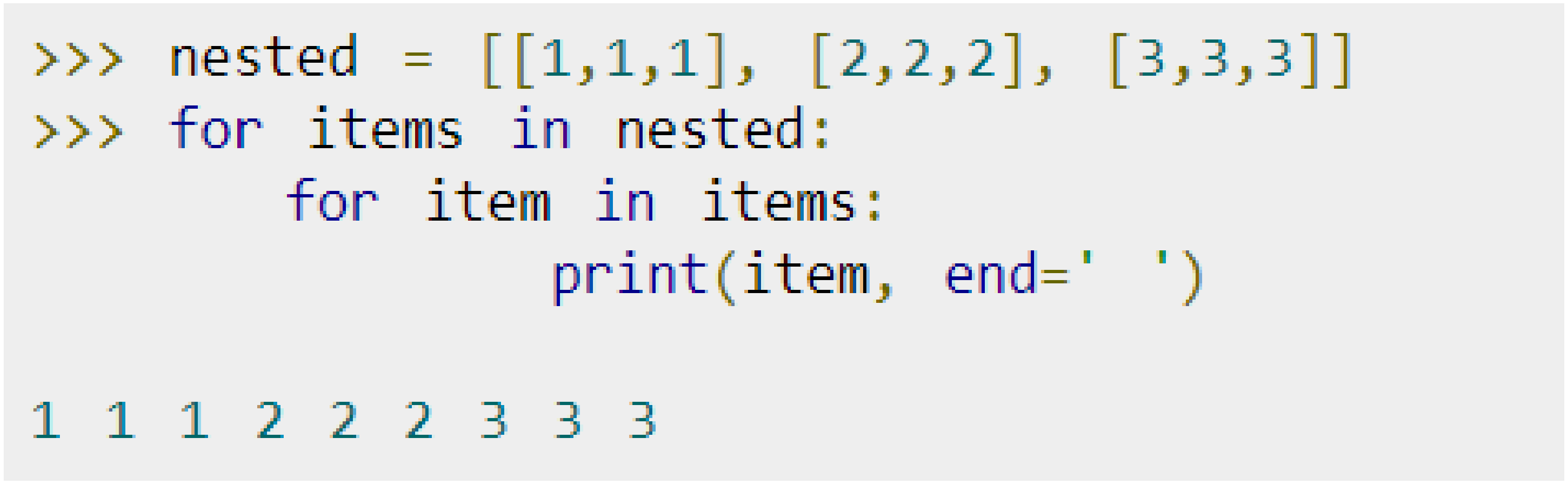
```
>>> assorted_list = [True, False, 1, 1.1, 1+2j, 'Learn', b'Python']
>>> first_element = assorted_list[0]
>>> print(first_element)
True
>>> print(assorted_list)
[True, False, 1, 1.1, (1+2j), 'Learn', b'Python']
>>> for item in assorted_list:
        print(type(item))

<class 'bool'>
<class 'bool'>
<class 'int'>
<class 'float'>
<class 'complex'>
<class 'str'>
<class 'bytes'>
```

Assign Value



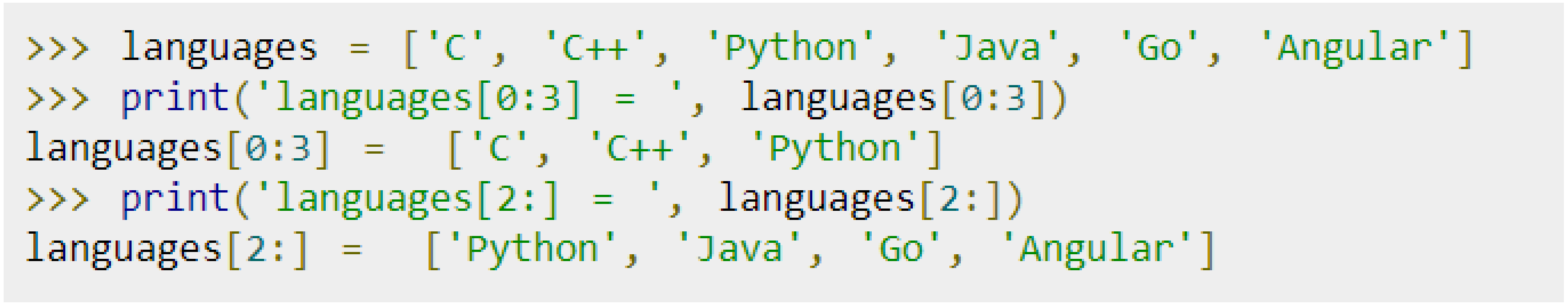
```
>>> simpleton = ['Learn', 'Python', '2']
>>> id(simpleton)
56321160
>>> simpleton
['Learn', 'Python', '2']
>>> simpleton[2] = '3'
>>> id(simpleton)
56321160
>>> simpleton
['Learn', 'Python', '3']
```

Nesting List



```
>>> nested = [[1,1,1], [2,2,2], [3,3,3]]
>>> for items in nested:
        for item in items:
                print(item, end=' ')

1 1 1 2 2 2 3 3 3
```

Slicing A List



```
>>> languages = ['C', 'C++', 'Python', 'Java', 'Go', 'Angular']
>>> print('languages[0:3] = ', languages[0:3])
languages[0:3] =  ['C', 'C++', 'Python']
>>> print('languages[2:] = ', languages[2:])
languages[2:] =  ['Python', 'Java', 'Go', 'Angular']
```

Tuple

```python
# Defining a tuple without any element
pure_tuple = ()
print (pure_tuple)
```

```
# Output- ()
```

Nesting

```python
# Creating a tuple with nested tuples
first_tuple = (3, 5, 7, 9)
second_tuple = ('learn', 'python 3')
nested_tuple = (first_tuple, second_tuple)
print(nested_tuple)
```

```
# Output - ((3, 5, 7, 9), ('learn', 'python 3'))
```

Repetition Operators

```
# How does repetition work with tuples
sample_tuple = ('Python 3',)*3
print(sample_tuple)
```

```
# Output - ('Python 3', 'Python 3', 'Python 3')
```

Slicing In Tuples

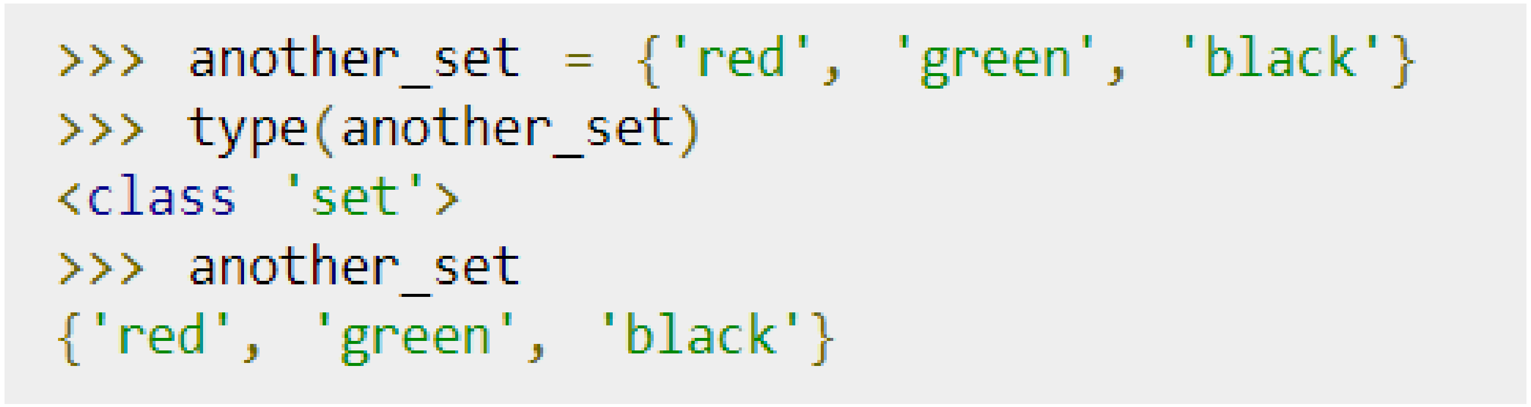
```
# How does slicing work with tuples

sample_tuple = (0 ,1, 2, 3, 4)

tuple_without_first_item = sample_tuple[1:]
print(tuple_without_first_item)

tuple_reverse = sample_tuple[::-1]
print(tuple_reverse)

tuple_from_3_to_5 = sample_tuple[2:4]
print(tuple_from_3_to_5)
```

```
# Output -
(1, 2, 3, 4)
(4, 3, 2, 1, 0)
(2, 3)
```

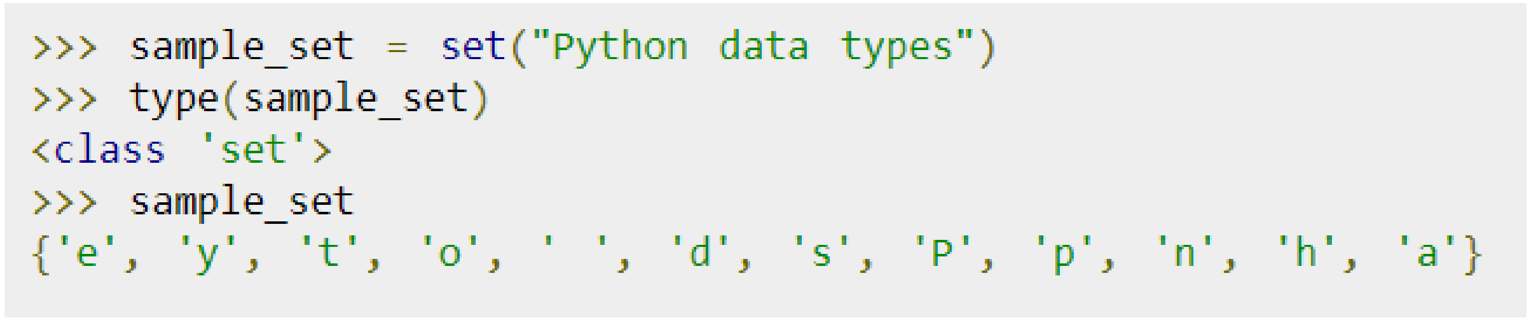
List Vs Tuples

Tuples do differ a bit from the list as they are **immutable**. Python **does not allow to modify** a tuple after it is created. We can not add or remove any element later. Instead, Python expects us to create a new one with the updated sequence of elements.

Sets

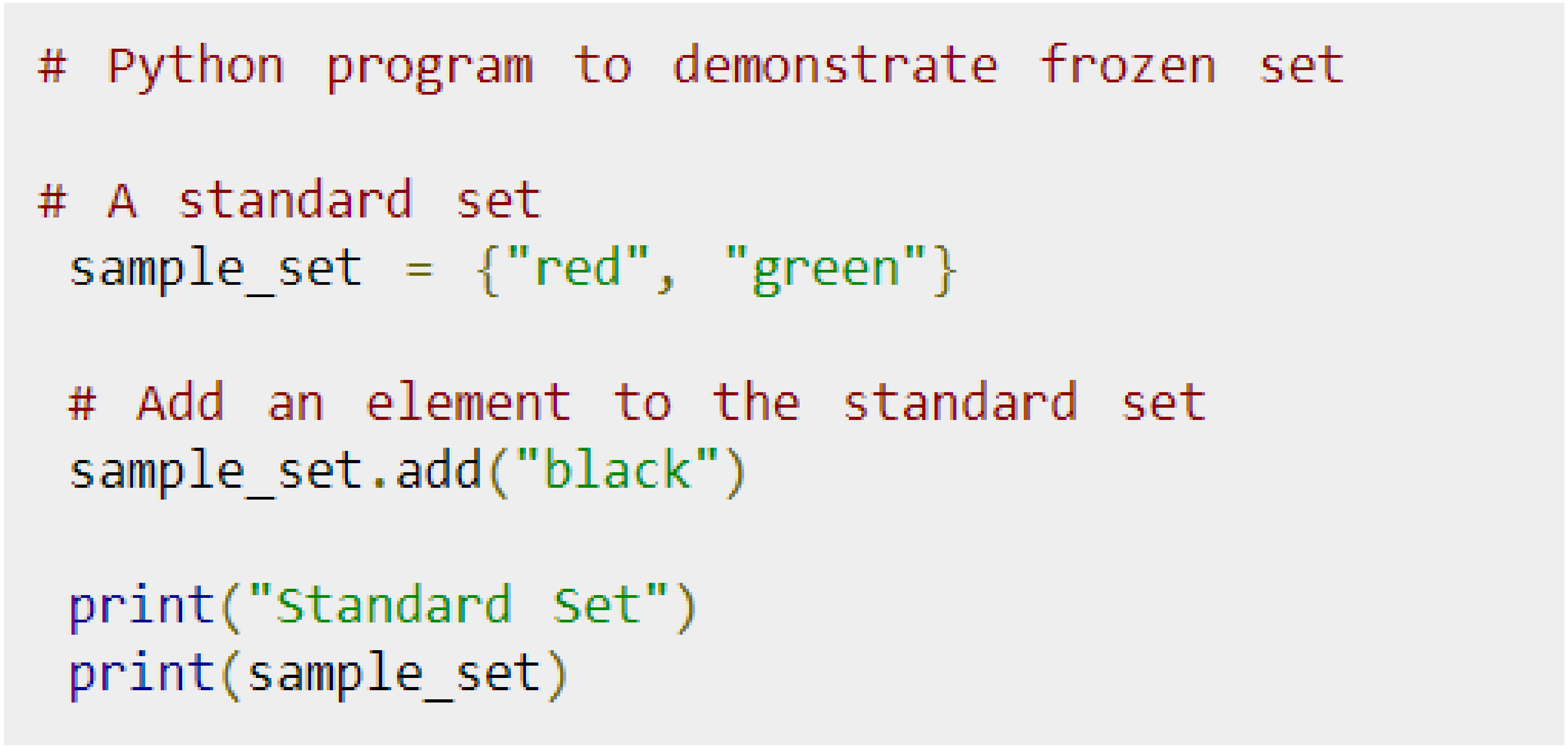


```
>>> another_set = {'red', 'green', 'black'}
>>> type(another_set)
<class 'set'>
>>> another_set
{'red', 'green', 'black'}
```



```
>>> sample_set = set("Python data types")
>>> type(sample_set)
<class 'set'>
>>> sample_set
{'e', 'y', 't', 'o', ' ', 'd', 's', 'P', 'p', 'n', 'h', 'a'}
```

Add or Remove



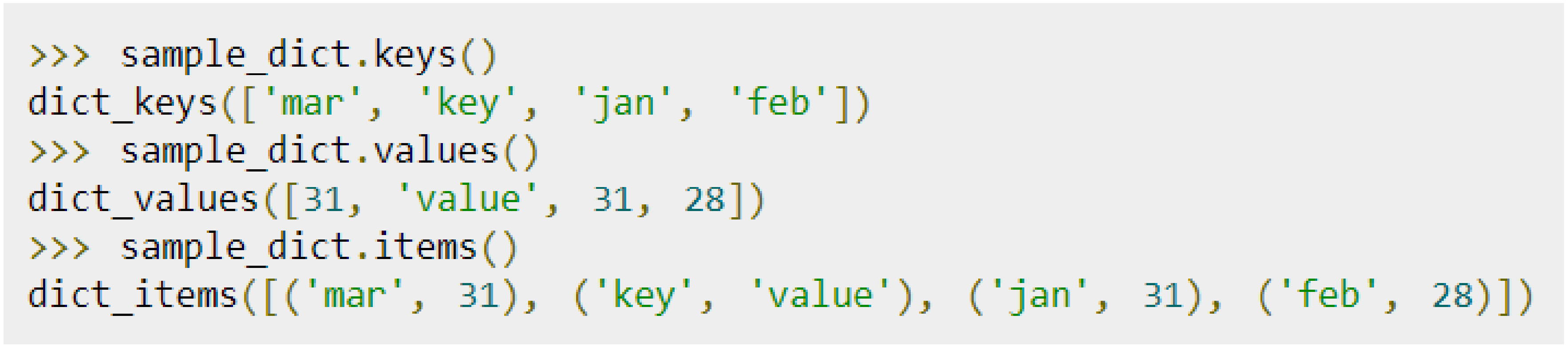
```
# Python program to demonstrate frozen set

# A standard set
 sample_set = {"red", "green"}

 # Add an element to the standard set
 sample_set.add("black")

 print("Standard Set")
 print(sample_set)
```

Dictionaries

```
>>> sample_dict = {'key':'value', 'jan':31, 'feb':28, 'mar':31}
>>> type(sample_dict)
<class 'dict'>
>>> sample_dict
{'mar': 31, 'key': 'value', 'jan': 31, 'feb': 28}
```

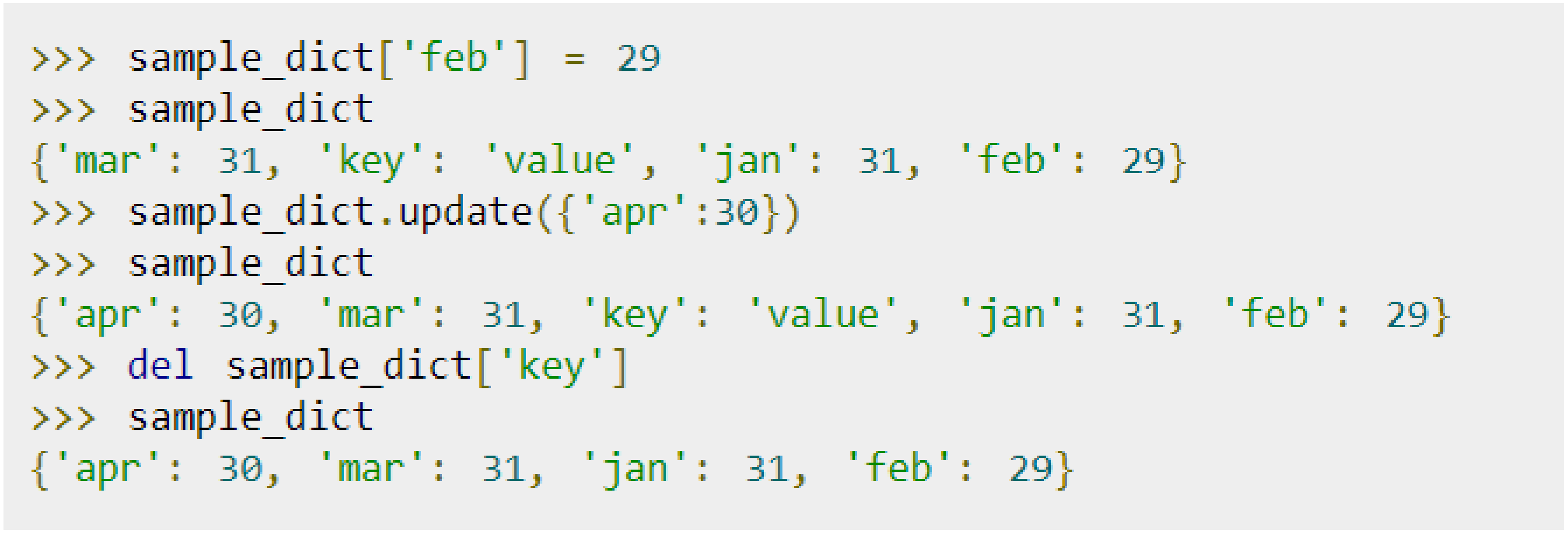
```
>>> sample_dict['jan']
31
>>> sample_dict['feb']
28
```

Dict Basic Functions



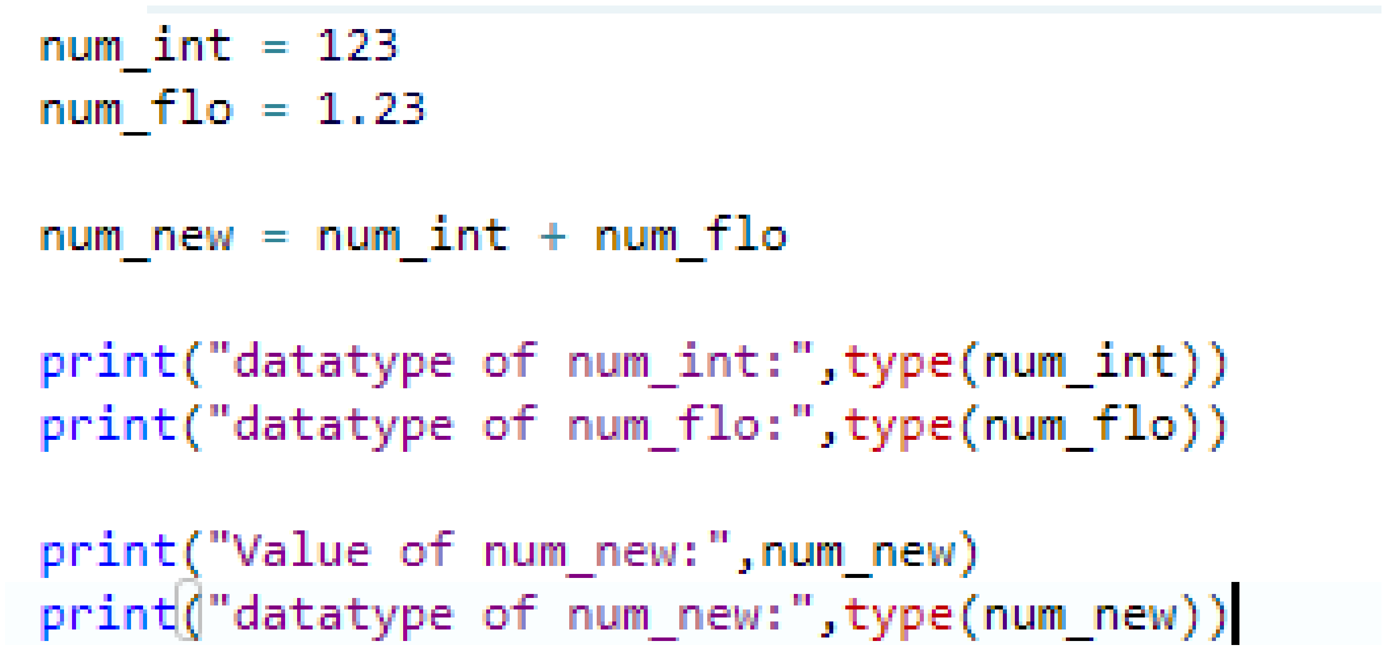
```
>>> sample_dict.keys()
dict_keys(['mar', 'key', 'jan', 'feb'])
>>> sample_dict.values()
dict_values([31, 'value', 31, 28])
>>> sample_dict.items()
dict_items([('mar', 31), ('key', 'value'), ('jan', 31), ('feb', 28)])
```

Add or Remove



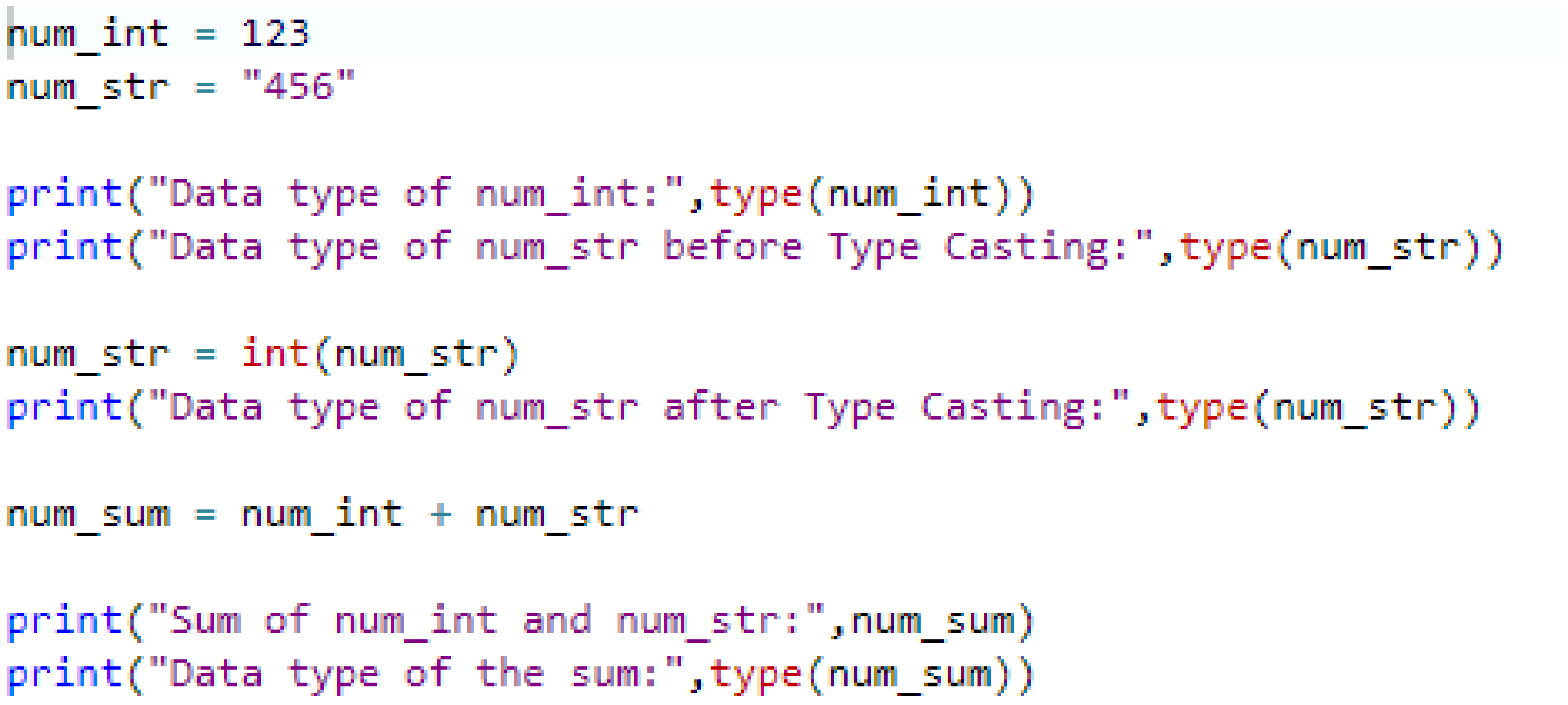
```
>>> sample_dict['feb'] = 29
>>> sample_dict
{'mar': 31, 'key': 'value', 'jan': 31, 'feb': 29}
>>> sample_dict.update({'apr':30})
>>> sample_dict
{'apr': 30, 'mar': 31, 'key': 'value', 'jan': 31, 'feb': 29}
>>> del sample_dict['key']
>>> sample_dict
{'apr': 30, 'mar': 31, 'jan': 31, 'feb': 29}
```

Implicit Type Conversion



```python
num_int = 123
num_flo = 1.23

num_new = num_int + num_flo

print("datatype of num_int:",type(num_int))
print("datatype of num_flo:",type(num_flo))

print("Value of num_new:",num_new)
print("datatype of num_new:",type(num_new))
```

Explicit Type Conversion



```python
num_int = 123
num_str = "456"

print("Data type of num_int:",type(num_int))
print("Data type of num_str before Type Casting:",type(num_str))

num_str = int(num_str)
print("Data type of num_str after Type Casting:",type(num_str))

num_sum = num_int + num_str

print("Sum of num_int and num_str:",num_sum)
print("Data type of the sum:",type(num_sum))
```

Example

```
print('Interest Calculator:')

amount = float(input('Principal amount ?'))
roi = float(input('Rate of Interest ?'))
yrs = int(input('Duration (no. of years) ?'))

total = (amount * pow(1 + (roi/100), yrs))
interest = total - amount
print('\nInterest = %0.2f' %interest)
```

Operators

- Arithmetic operators
- Comparison operators
- Logical operators
- Bitwise operators
- Assignment operators
- Identity operators
- Membership operators

Arithmetic Operators

Operator	Purpose	Usage
+	Addition – Sum of two operands	a+b
–	Subtraction – Difference between the two operands	a-b
*	Multiplication – Product of the two operands	a*b
/	Float Division – Quotient of the two operands	a/b
//	Floor Division – Quotient of the two operands (Without fractional part)	a//b
%	Modulus – Integer remainder after division of ‘a’ by ‘b’	a%b
**	Exponent – Product of ‘a’ by itself ‘b’ times (a to the power of b)	a**b

Comparison Operators

Operator	Purpose	Usage
>	Greater than – if the left operand is greater than the right, then it returns true.	a>b
<	Less than – if the left operand is less than the right, then it returns true.	a<b
==	Equal to – if two operands are equal, then it returns true.	a==b
!=	Not equal to – if two operands aren't equal, then it returns true.	a!=b
>=	Greater than or equal – if the left operand is greater than or equal to the right, then it returns true.	a>=b
<=	Less than or equal – if the left operand is less than or equal to the right, then it returns true.	a<=b

Logical Operators

Operator	Purpose	Usage
and	if ‘a’ is false, then ‘a’, else ‘b’	a and b
or	if ‘a’ is false, then ‘b’, else ‘a’	a or b
not	if ‘a’ is false, then True, else False	not a

Bitwise Operators

Operator	Purpose	Usage
&	Bitwise AND – compares two operands on a bit level and returns 1 if both the corresponding bits are 1	a & b
|	Bitwise OR – compares two operands on a bit level and returns 1 if any of the corresponding bits is 1	a | b
~	Bitwise NOT – inverts all of the bits in a single operand	~a
^	Bitwise XOR – compares two operands on a bit level and returns 1 if any of the corresponding bits is 1, but not both	a ^ b
>>	Right shift – shifts the bits of ‘a’ to the right by ‘b’ no. of times	a >> b
<<	Left shift – shifts the bits of ‘a’ to the left by ‘b’ no. of times	a << b

Assignment Operators

Operator	Example	Similar to
=	a=4	a=4
+=	a+=4	a=a+4
-=	a-=4	a=a-4
*=	a*=4	a=a*4
/=	a/=4	a=a/4
%=	a%=4	a=a%4
**=	a**=4	a=a**4
&=	a&=4	a=a&4
|=	a|=4	a=a|4
^=	a^=4	a=a^4
>>=	a>>=4	a=a>>4
<<=	a<<=4	a=<<4

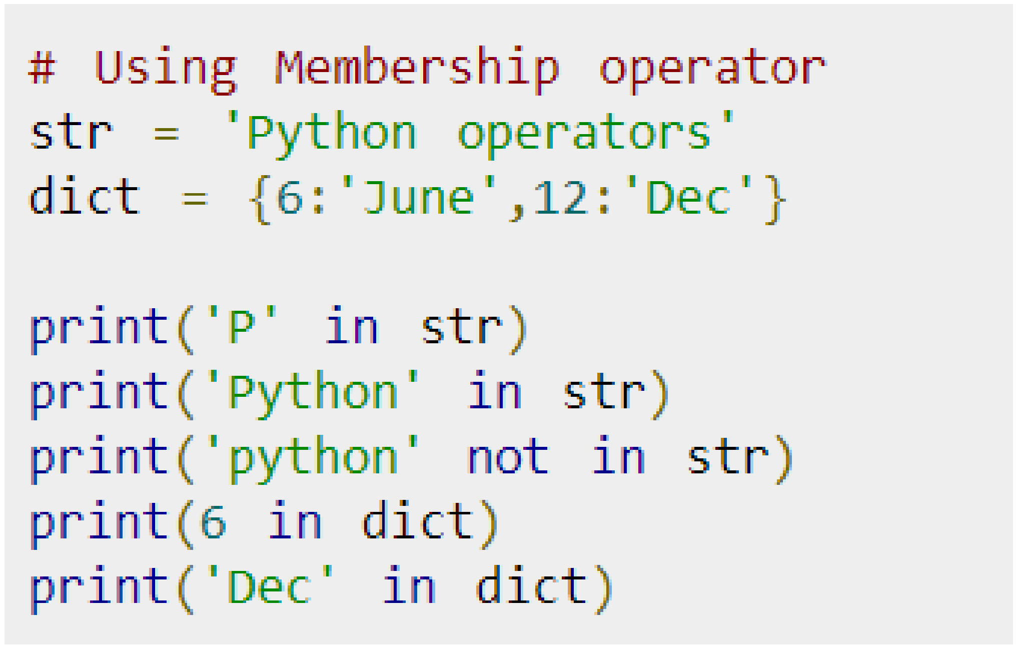
Identity Operators

Operator	Purpose	Usage
is	True – if both the operands refer to the same object, else False	a is b (True if id(a) and id(b) are the same)
is not	True – if the operands refer to different objects, else False	a is not b (True if id(a) and id(b) are different)

```
# Using 'is' identity operator
a = 7
if (type(a) is int):
 print("true")
else:
 print("false")

# Using 'is not' identity operator
b = 7.5
if (type(b) is not int):
 print("true")
else:
 print("false")
```

Membership Operators

Operator	Purpose	Usage
in	True – if the value exists in the sequence	7 in [3, 7, 9]
not in	True – if the value doesn't found in the sequence	7 not in [3, 5, 9]



```python
# Using Membership operator
str = 'Python operators'
dict = {6:'June',12:'Dec'}

print('P' in str)
print('Python' in str)
print('python' not in str)
print(6 in dict)
print('Dec' in dict)
```

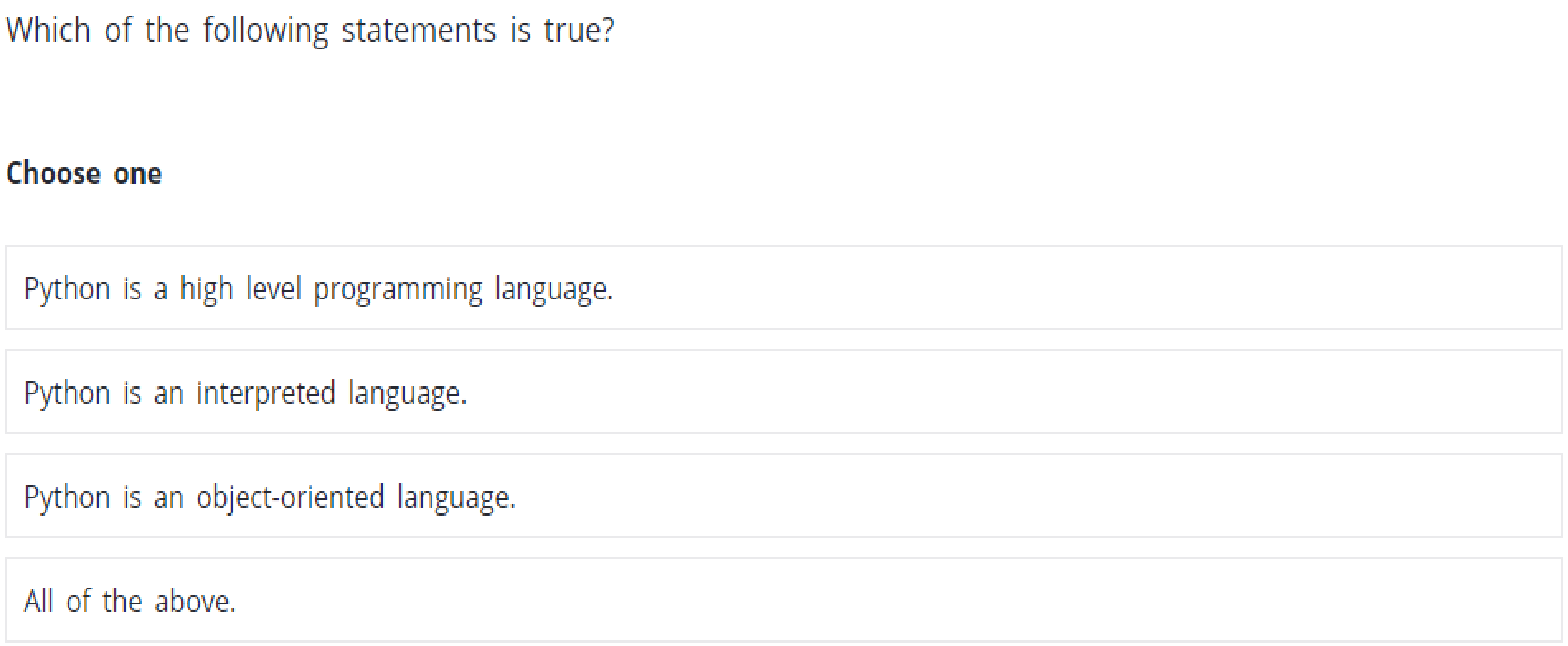
Print Statement

- print(‘hi’)
- print(5+9)
- print(‘hi’+’ hello’)
- print(1,2,3)
- print(1,2,3,sep=‘-’)
- print(‘%s hello’ % (name))

%s	Used to print String
%d	Used to print integers
%f	Used to print floating point integers

Quiz

Quiz 1

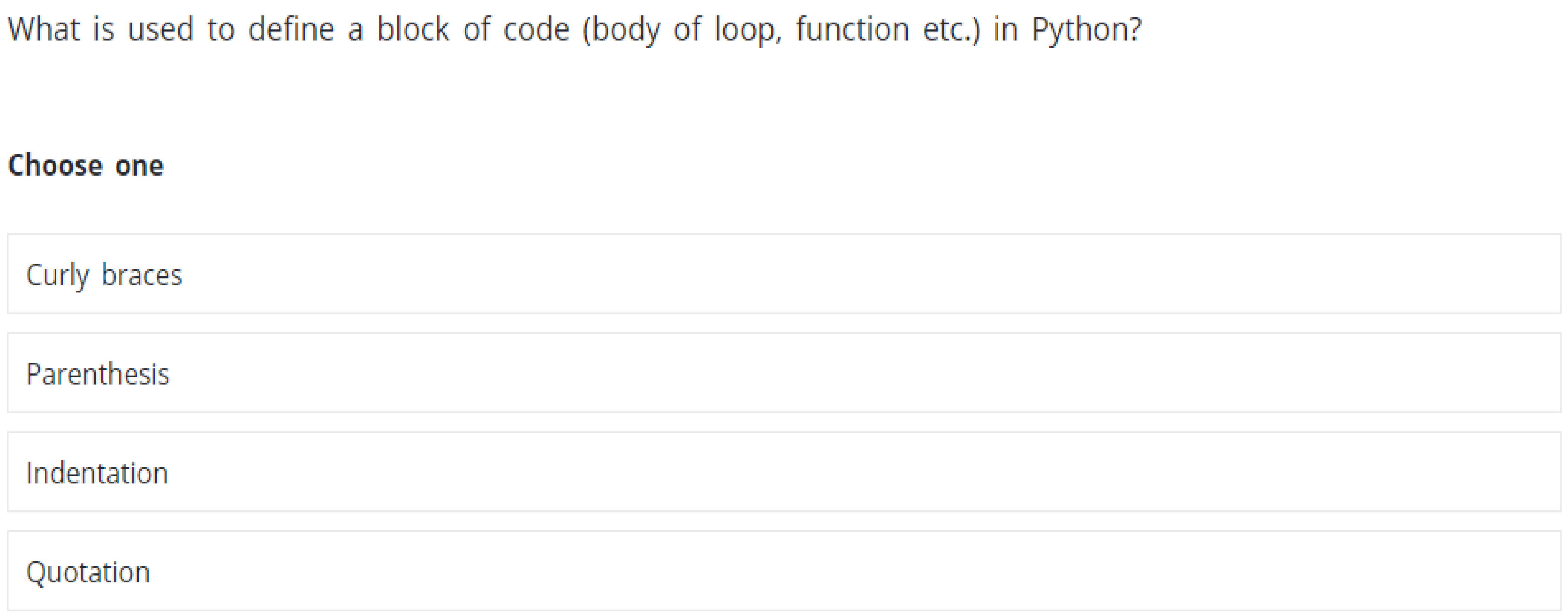


Which of the following statements is true?

Choose one

- Python is a high level programming language.
- Python is an interpreted language.
- Python is an object-oriented language.
- All of the above.

Quiz 2



What is used to define a block of code (body of loop, function etc.) in Python?

Choose one

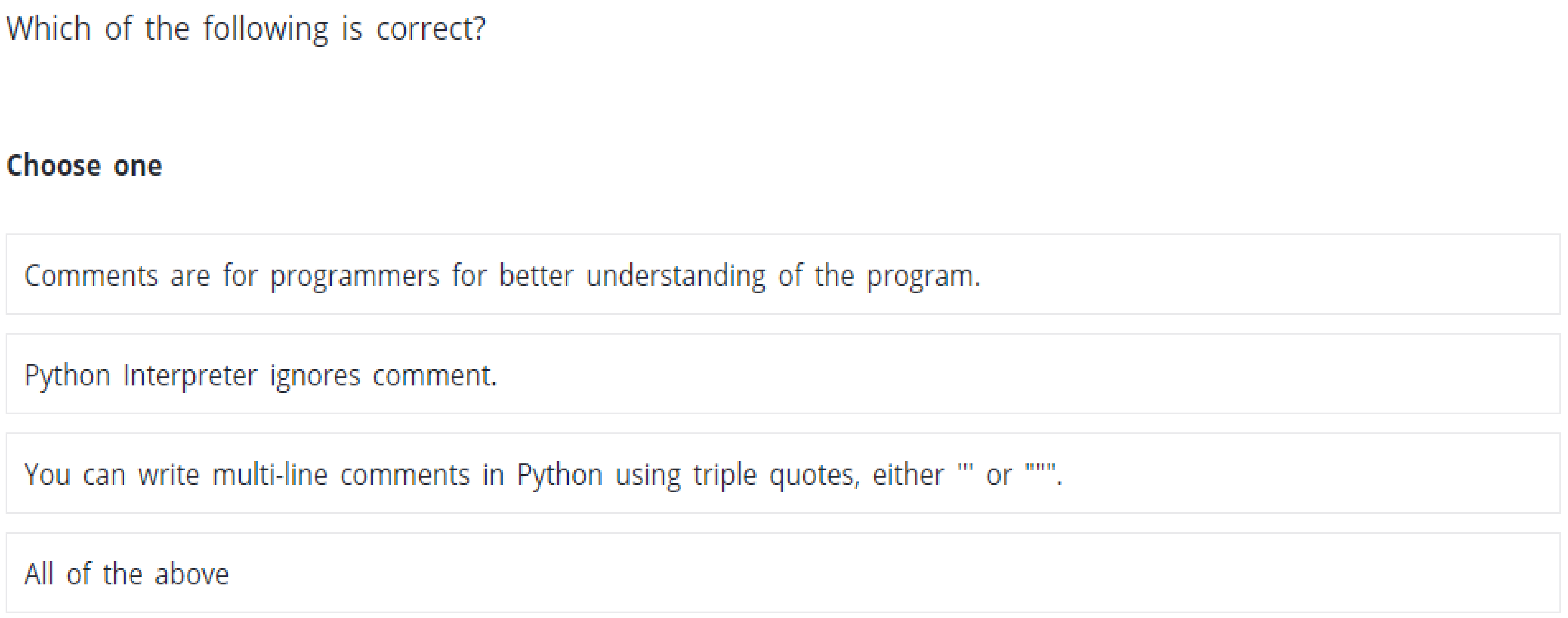
Curly braces

Parenthesis

Indentation

Quotation

Quiz 3

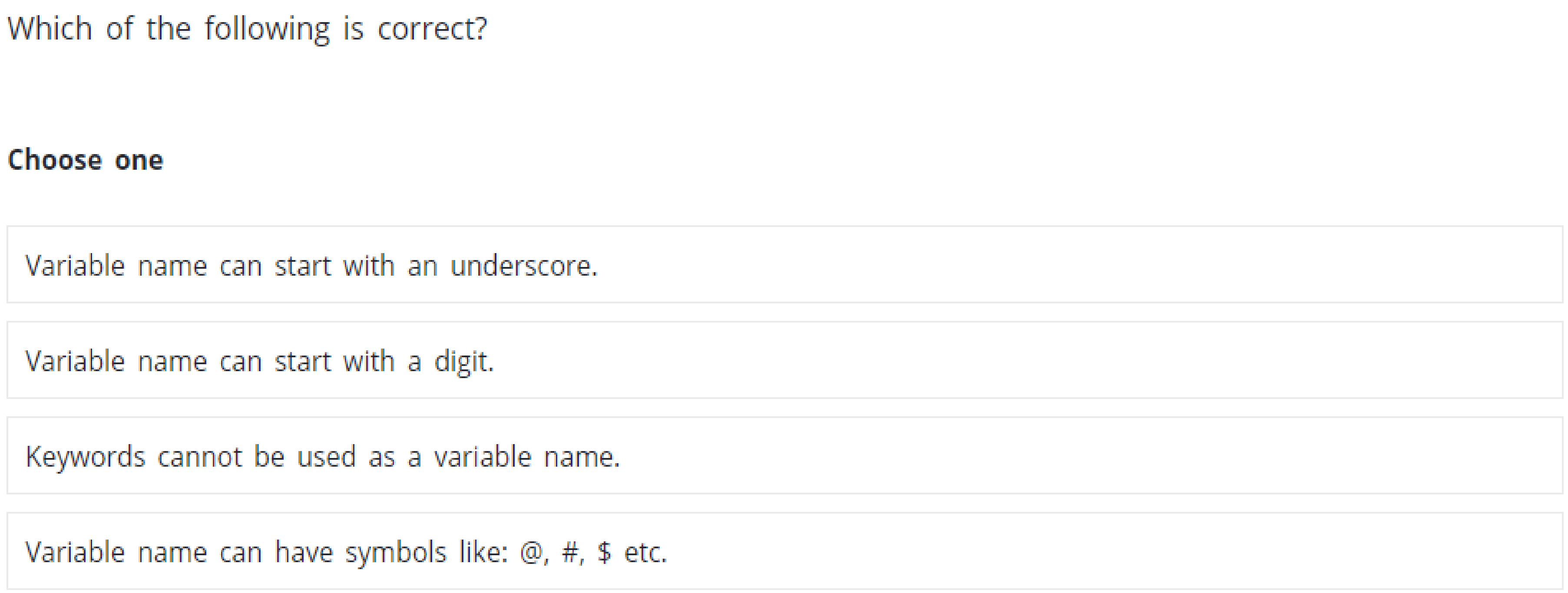


Which of the following is correct?

Choose one

- Comments are for programmers for better understanding of the program.
- Python Interpreter ignores comment.
- You can write multi-line comments in Python using triple quotes, either ''' or """.
- All of the above

Quiz 4

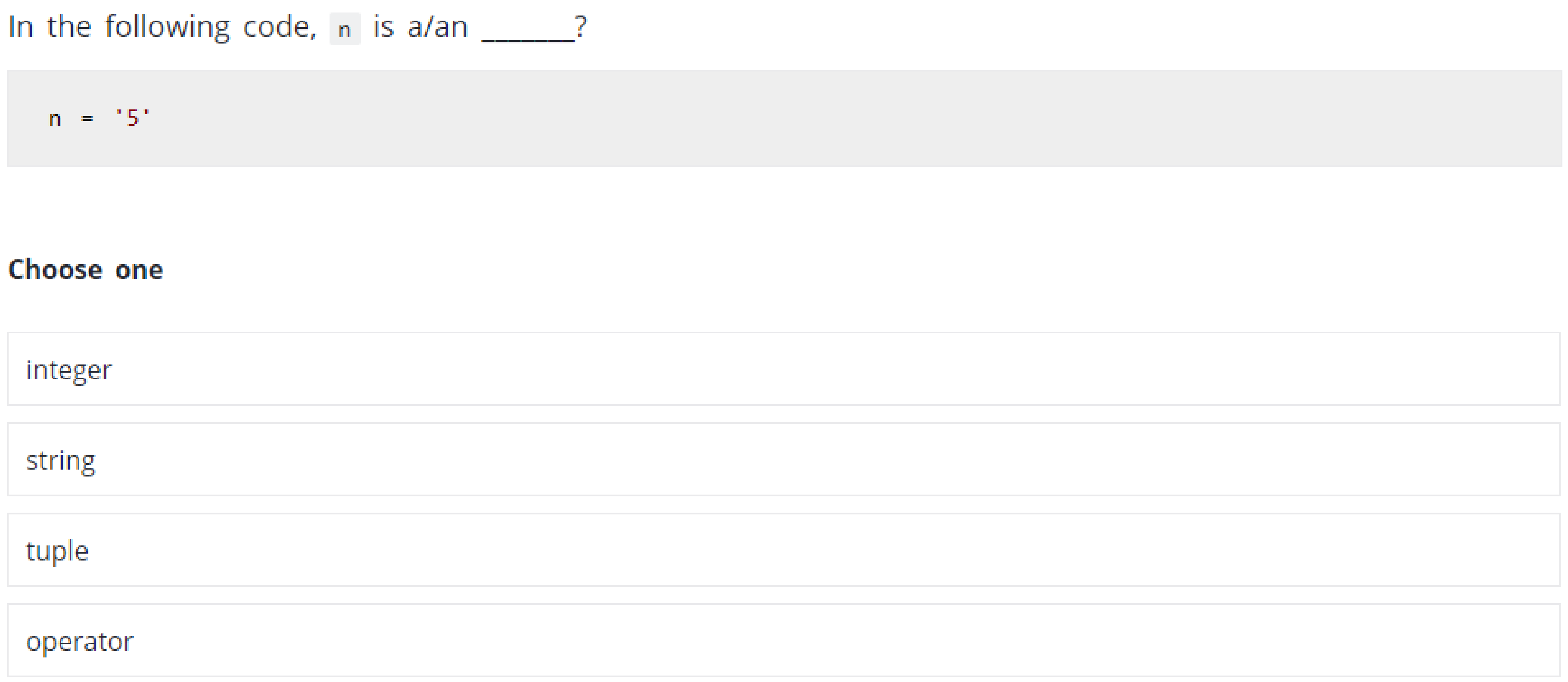


Which of the following is correct?

Choose one

- Variable name can start with an underscore.
- Variable name can start with a digit.
- Keywords cannot be used as a variable name.
- Variable name can have symbols like: @, #, $ etc.

Quiz 5



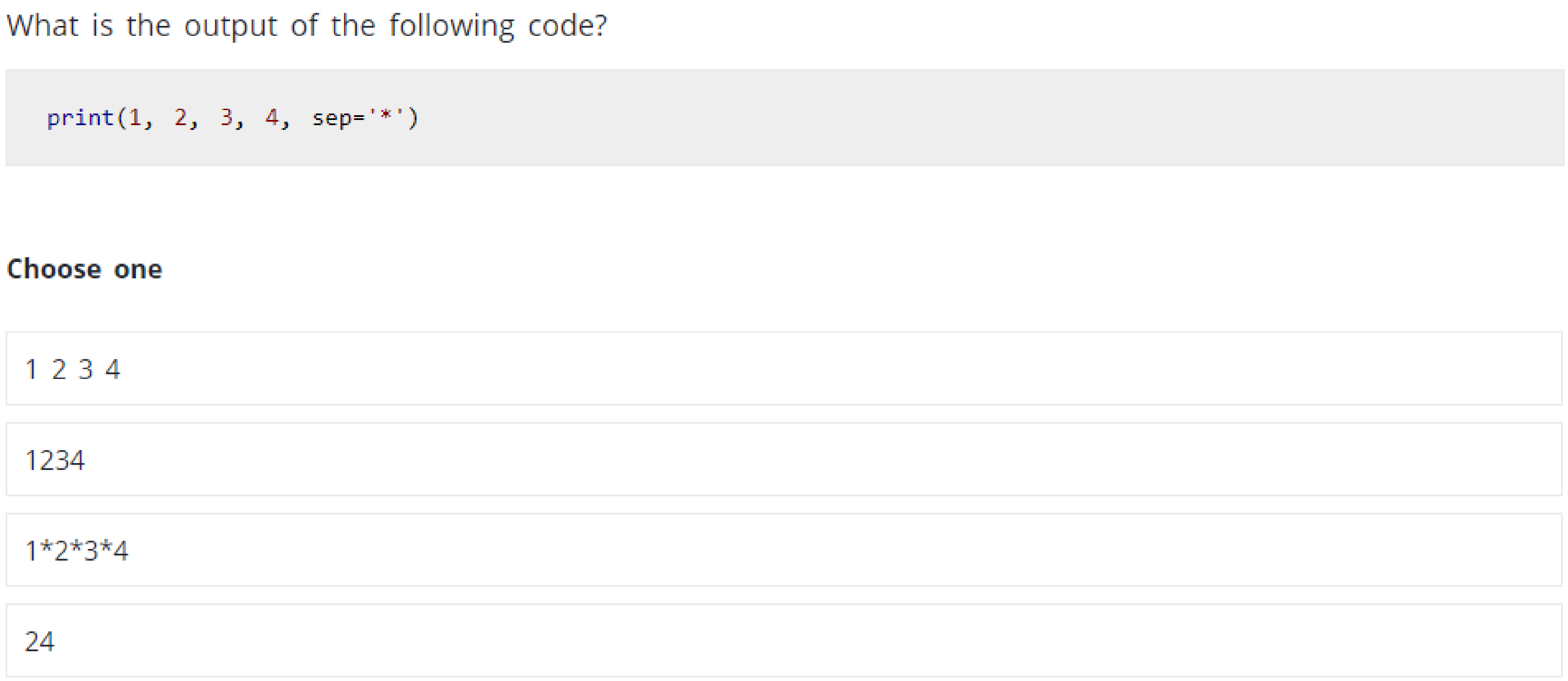
In the following code, `n` is a/an ______?

```
n = '5'
```

Choose one

- integer
- string
- tuple
- operator

Quiz 6



What is the output of the following code?

```
print(1, 2, 3, 4, sep='*')
```

Choose one

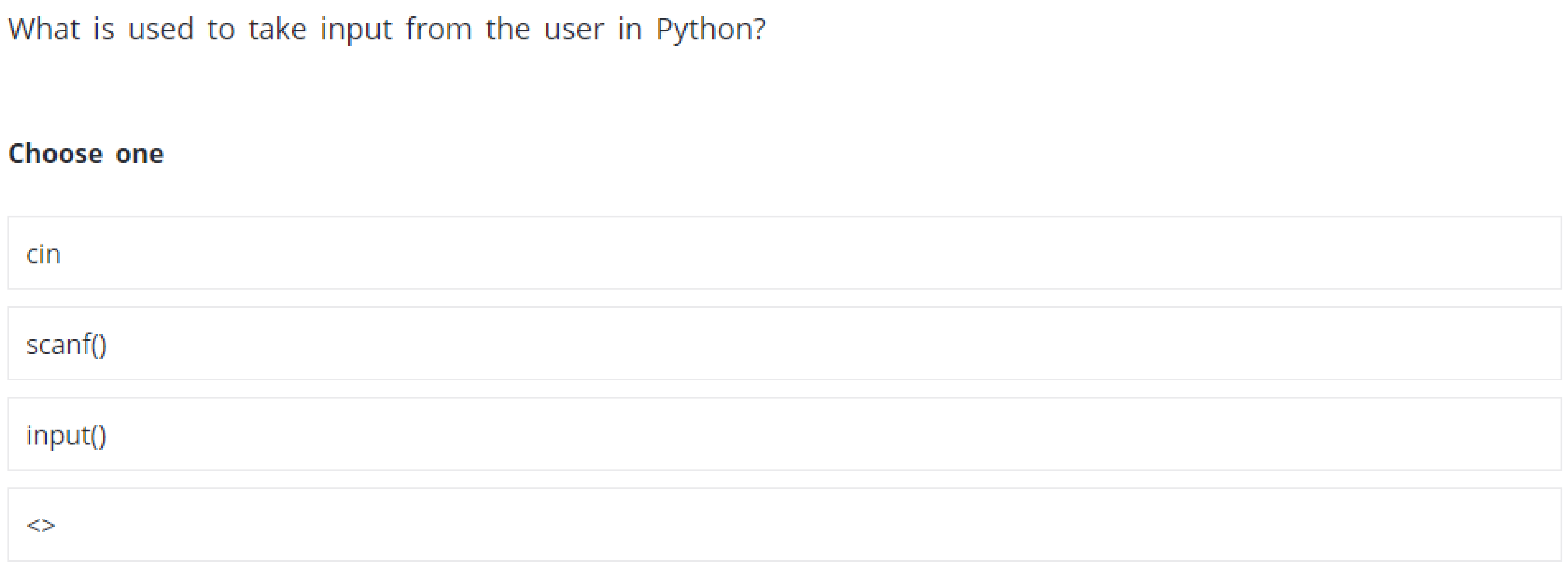
1 2 3 4

1234

1*2*3*4

24

Quiz 7

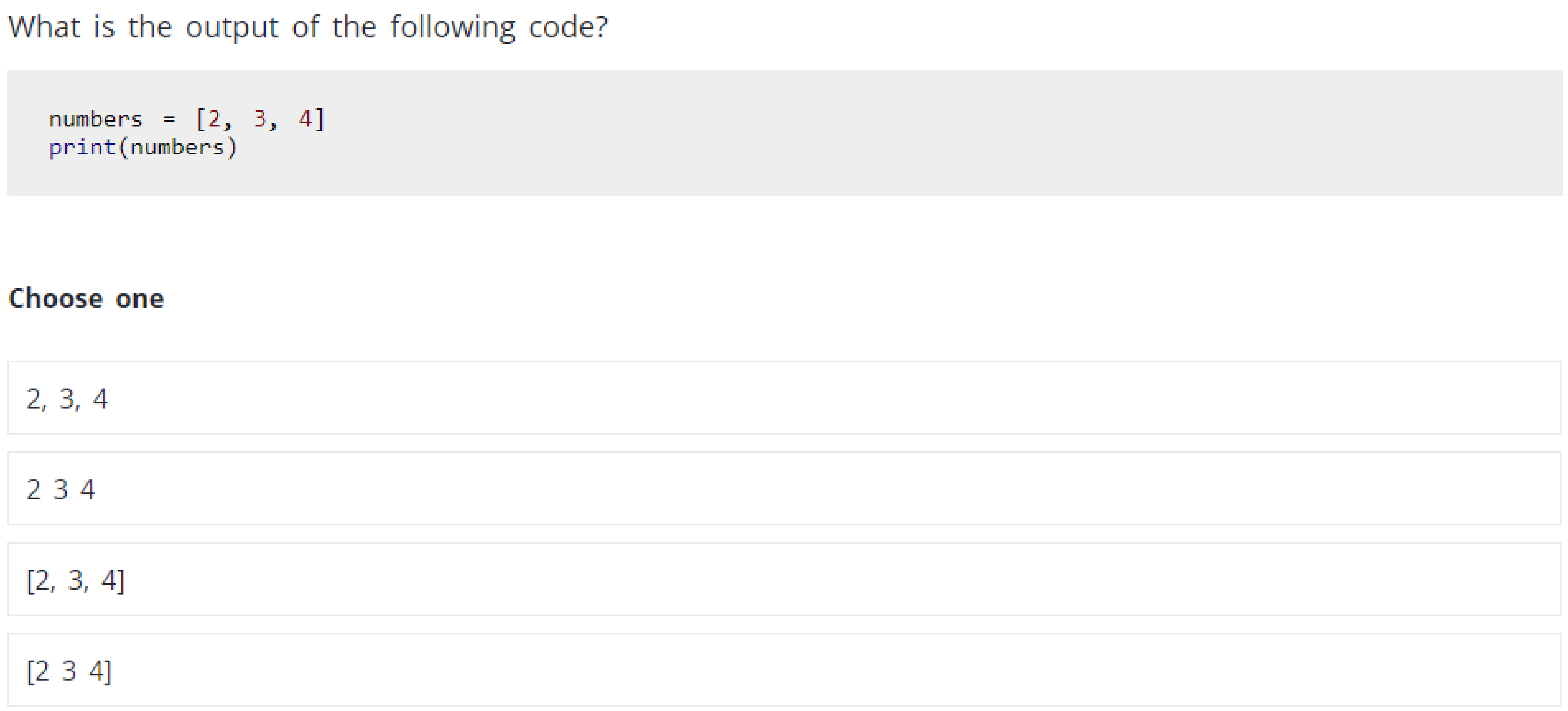


What is used to take input from the user in Python?

Choose one

- cin
- scanf()
- input()
- <>

Quiz 8



What is the output of the following code?

```
numbers = [2, 3, 4]
print(numbers)
```

Choose one

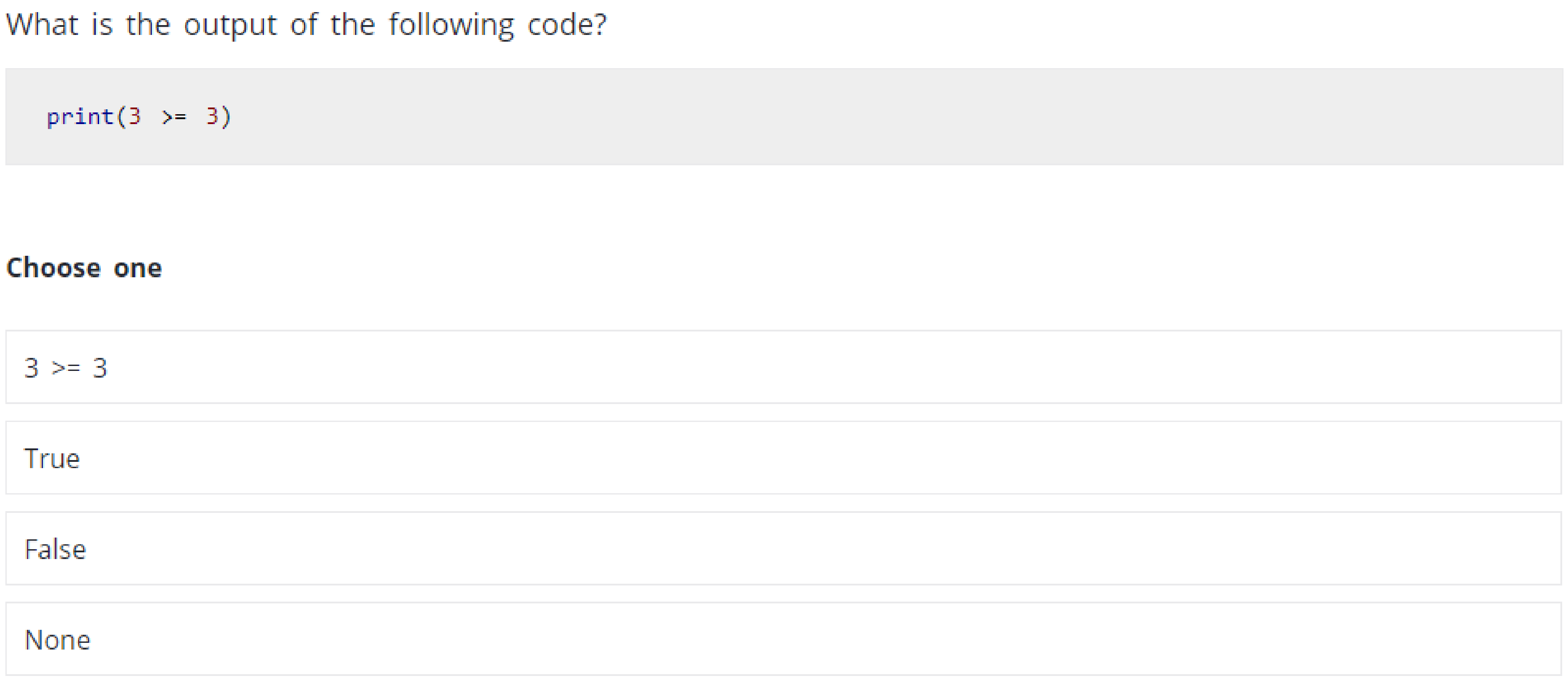
2, 3, 4

2 3 4

[2, 3, 4]

[2 3 4]

Quiz 9



What is the output of the following code?

```
print(3 >= 3)
```

Choose one

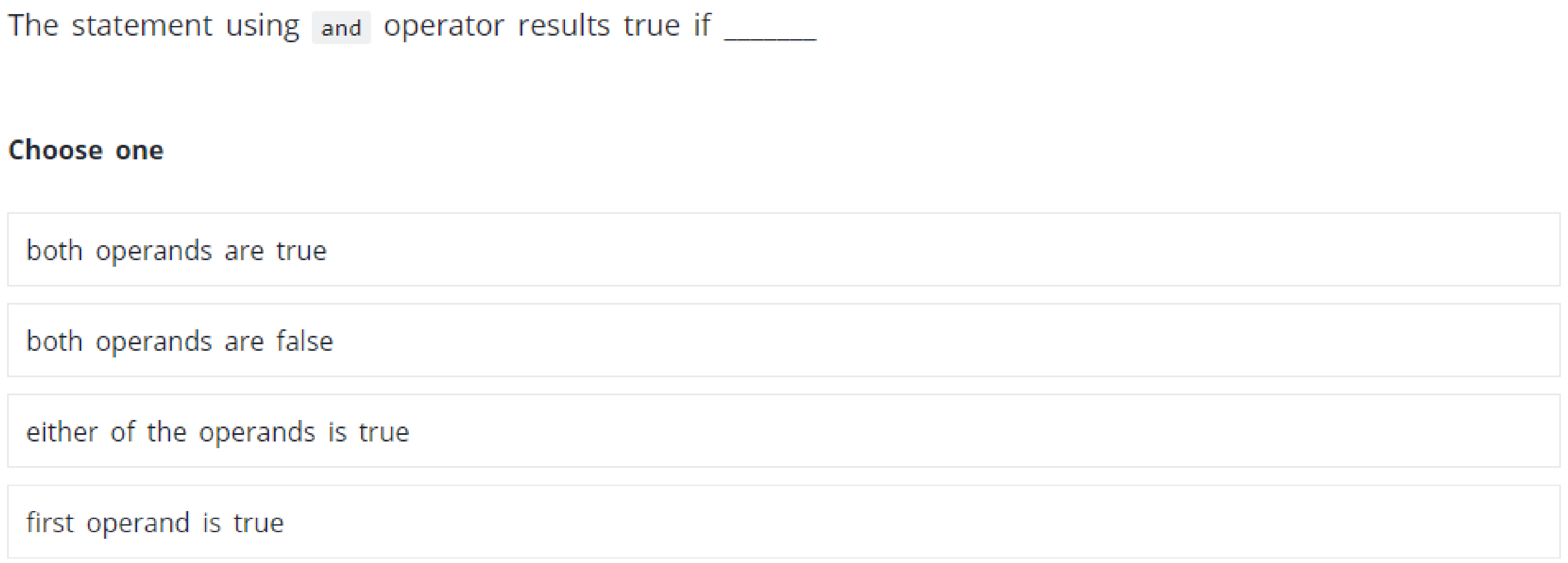
3 >= 3

True

False

None

Quiz 10

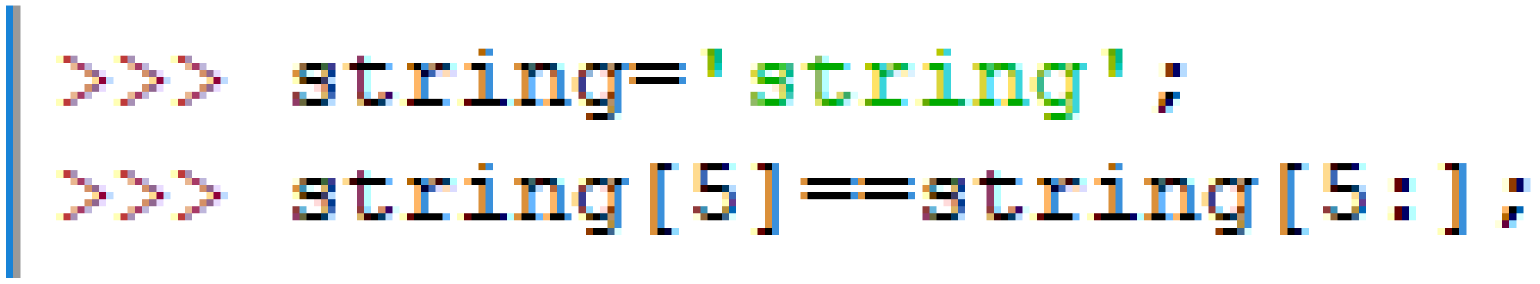


The statement using `and` operator results true if ______

Choose one

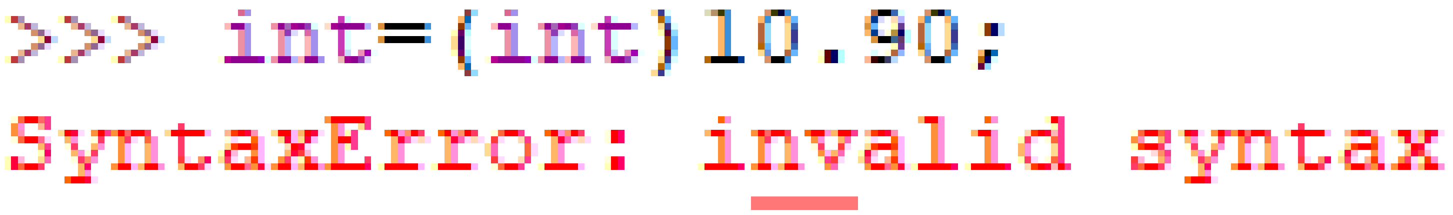
- both operands are true
- both operands are false
- either of the operands is true
- first operand is true

Quiz 11: What is the Output?



```
>>> string='string';
>>> string[5]==string[5:];
```

Quiz 12: what is the mistake?

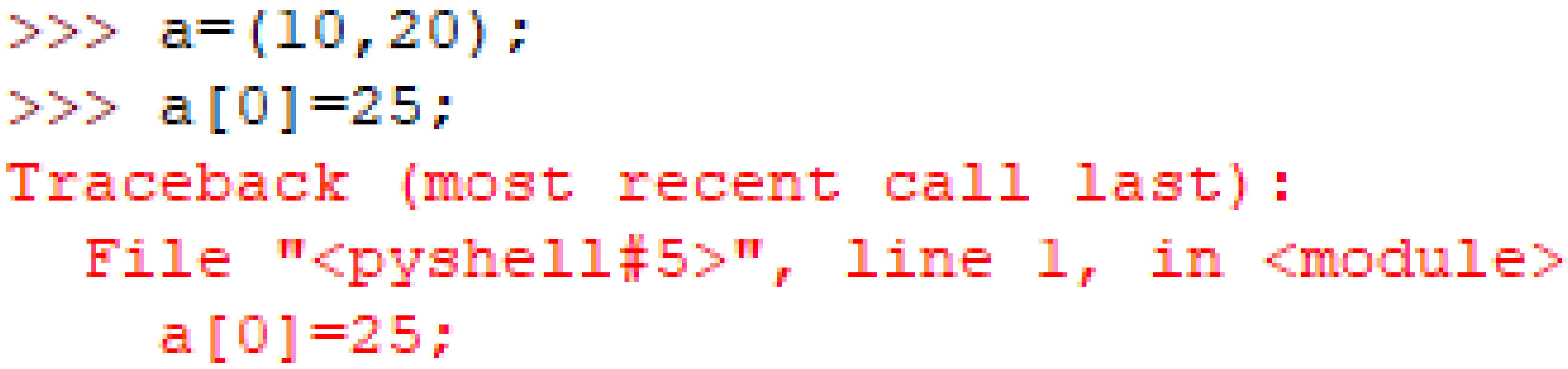


```
>>> int=(int)10.90;
SyntaxError: invalid syntax
```

Quiz 13: Which is not relevant ?

```
a = [ 1,2,3];
a = [ { “key”: “value” } ]
a = list((1,2,3))
a = { 1,2,3 }
```

Quiz 14: What is the mistake ?



```
>>> a=(10,20);
>>> a[0]=25;
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    a[0]=25;
```

Quiz 15: Which is the incorrect statement?

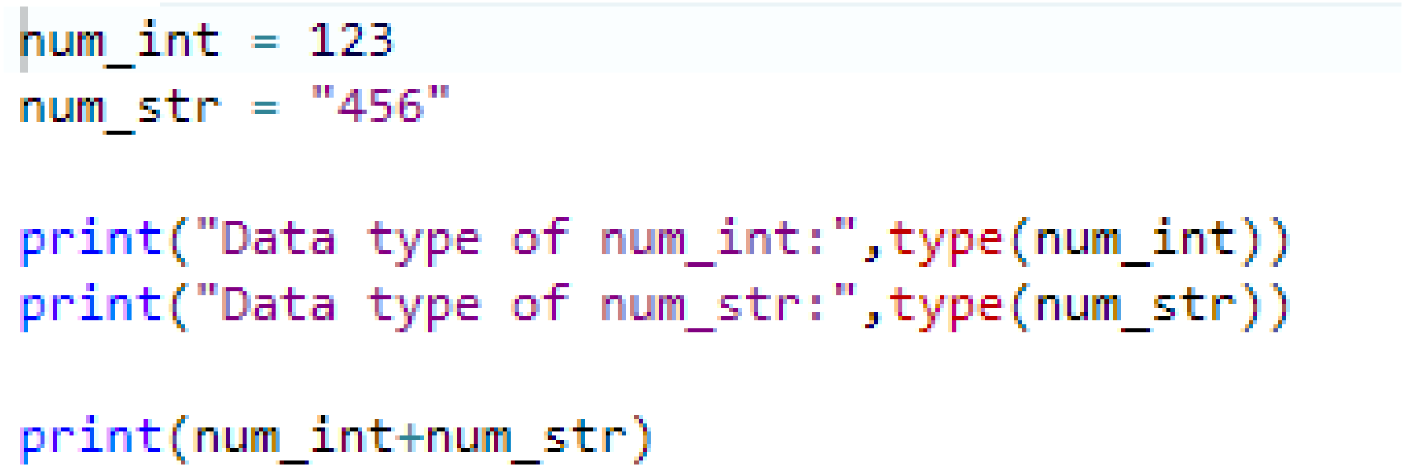
a = true and true

a = True or False

a = 1 and 1

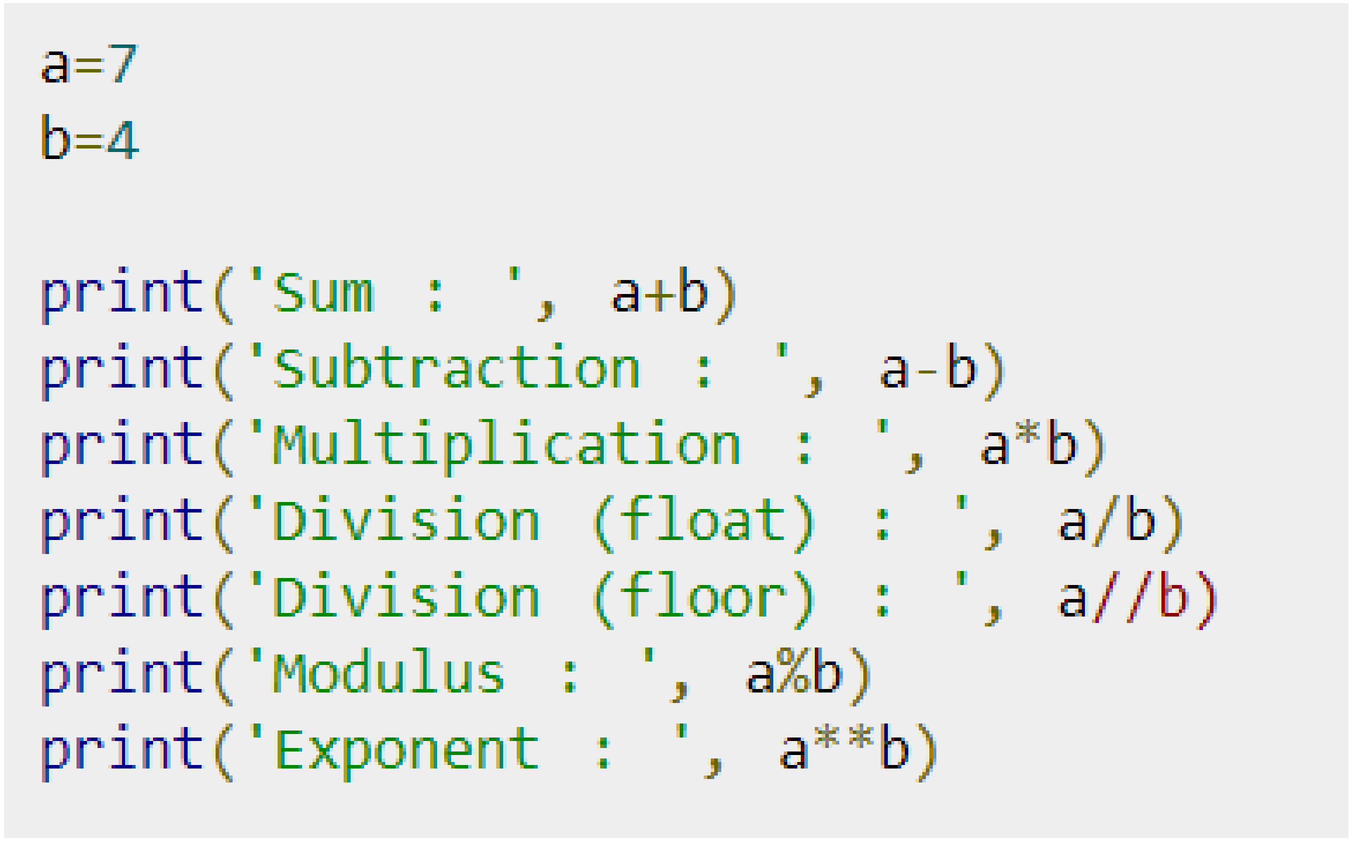
a = 0 and 1

Quiz 15: What is Output?



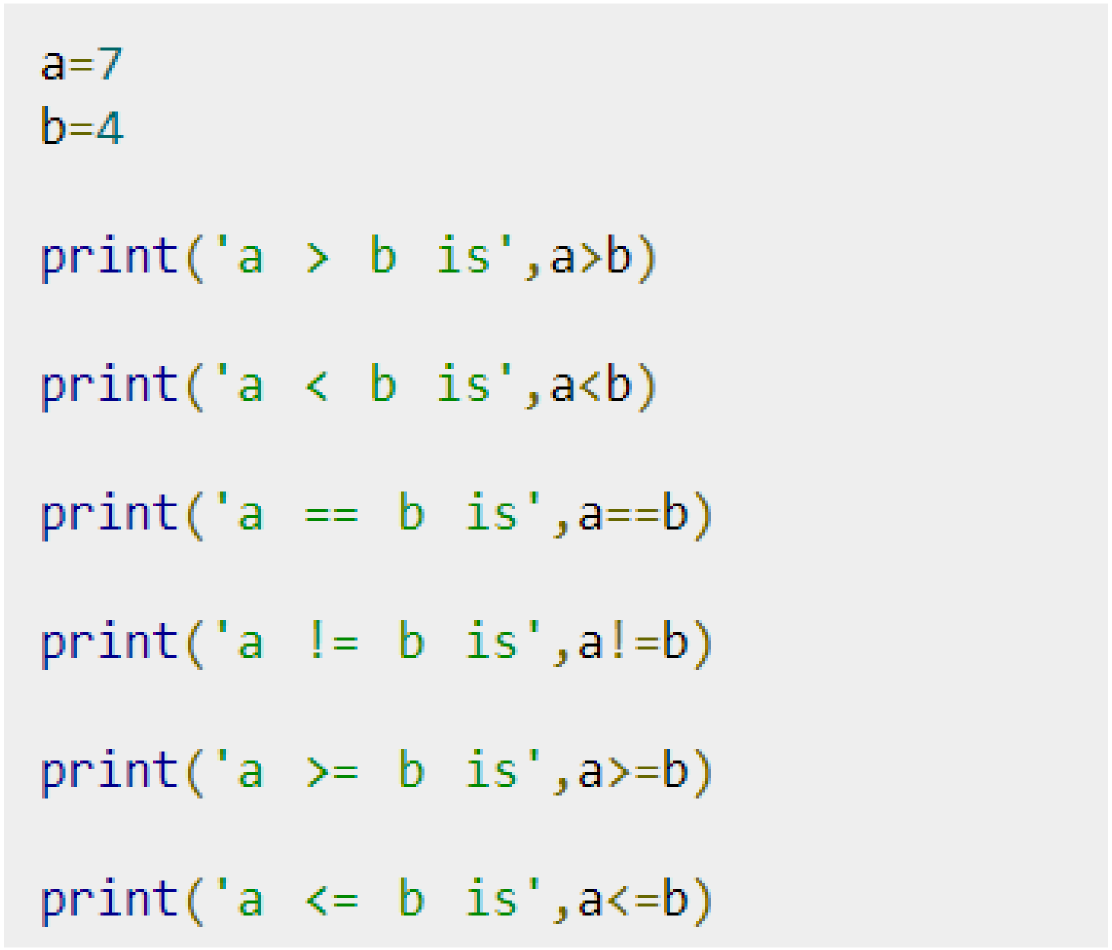
```
num_int = 123
num_str = "456"

print("Data type of num_int:",type(num_int))
print("Data type of num_str:",type(num_str))

print(num_int+num_str)
```

Quiz 16: What is the Output?



```
a=7
b=4

print('Sum : ', a+b)
print('Subtraction : ', a-b)
print('Multiplication : ', a*b)
print('Division (float) : ', a/b)
print('Division (floor) : ', a//b)
print('Modulus : ', a%b)
print('Exponent : ', a**b)
```

Quiz 17: What is the Output?



```
a=7
b=4

print('a > b is',a>b)

print('a < b is',a<b)

print('a == b is',a==b)

print('a != b is',a!=b)

print('a >= b is',a>=b)

print('a <= b is',a<=b)
```

ANY QUERY?

Ayyappan Murugesan

Phone: 7010774329

WhatsApp: 8870483557