



Python Programming

Basics of Python Programming

Outline

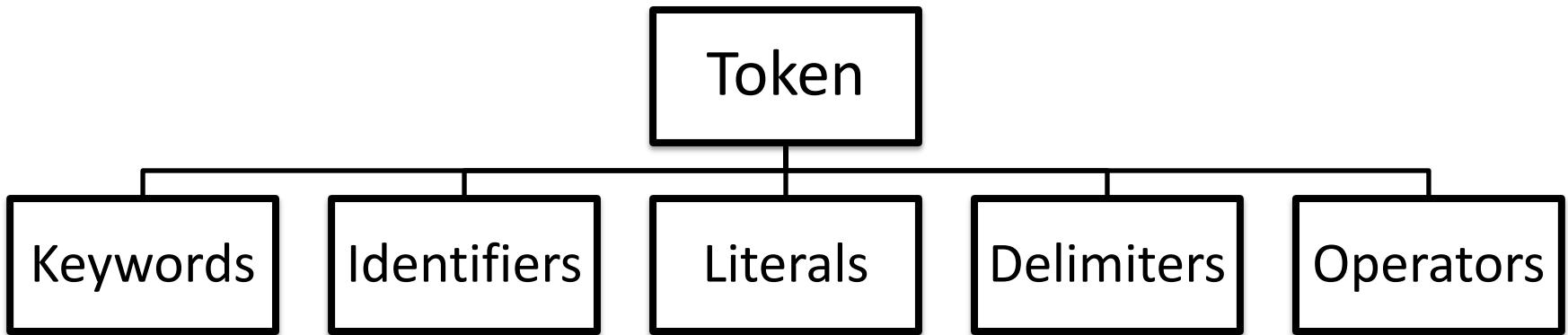
- Python Character Set
- Token
- Python Core Data Type
- Assigning Value to Variable
- print() function
- input() function
- eval() function
- Formatting Number and Strings



Python Character Set

- Any program written in python contains words or statements which follow a sequence of characters. When these characters are submitted to python interpreter, they are interpreted or identified in various contexts such as characters, identifiers, constants etc.
- Python uses following character set:
- **Letters:** lowercase and uppercase letters
- **Digits:** 0 1 2 3 4 5 6 7 8 9
- **White Space Characters :**
Space (), Horizontal tab (\t), Form feed (\f) ,Vertical tab (\v), New-line character (\n) etc.
- **Special Symbols:**
! # % ^ & * () - _ = + ~ ' " : ; ? / | \ { } [] , . < > \$

Token



Python interpreter reads code as a sequence of characters and translates them into a sequence of tokens, classifying each by its lexical category, this operation is called “tokenization”.

Token

- **Keyword**
- Keywords are the reserved words in Python. Keywords cannot be used as variable name, function name or any other identifier.
- There are 33 keywords in Python 3.4.2. This number can vary slightly in course of time.

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

Token

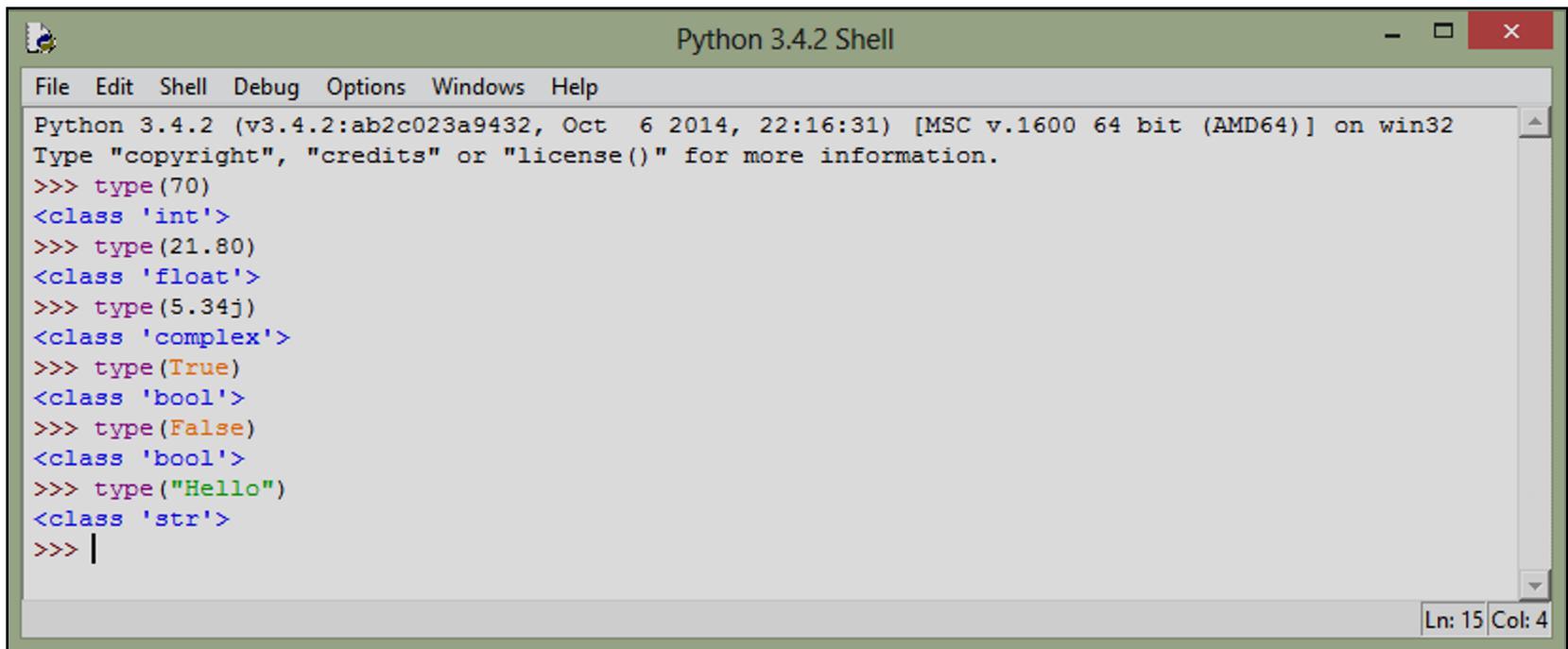
- **Identifiers**
- Identifier is the name given to entities like class, functions, variables etc. in Python. It helps differentiating one entity from another.
- **Rules for writing identifiers**
- Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore (_). Names like myClass, var_1 and print_this_to_screen, all are valid example.
- An identifier cannot start with a digit. variable is invalid, but variable1 is perfectly fine.
- Keywords cannot be used as identifiers.
- You cannot use special symbols like !, @, #, \$, % etc. as an identifier.
- Identifier can be of any length.

Token

- **Literals**
- Literals are numbers or strings or characters that appear directly in a program. A list of some literals in python is as follows:
- 70 # Integer Literal
- 21.80 # Floating Point Literal
- 5.34j # Complex Literals
- True/ False # Boolean Literals
- "Hello" # String Literals

Token

- **Type of Literals**
- To know the exact type of any value, python offers an in built method called `type`.
- The syntax to know the type of any value is `type(value)`



Python 3.4.2 Shell

```
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.

>>> type(70)
<class 'int'>
>>> type(21.80)
<class 'float'>
>>> type(5.34j)
<class 'complex'>
>>> type(True)
<class 'bool'>
>>> type(False)
<class 'bool'>
>>> type("Hello")
<class 'str'>
>>> |
```

Ln: 15 Col: 4

Token

- **Delimiters**
- Delimiters are symbols that perform three special roles in Python: grouping, punctuation, and assignment/binding of objects to names.

Python's Delimiters															
()	[]	{	}	grouping									
.	,	:	;	@	punctuation										
=	+=	-=	*=	/=	//=	%=	**=	arithmetic assignment/binding							
&=	=	^=	<<=	>>=	bit-wise assignment/binding										

Token

- **Operators**
- Operators are particular symbols which operate on some values and produce an output. The values are known as Operands. Operators compute a result based on the value(s) of their operands: e.g., + is the addition operator.

Python's Operators							
+ - * / // % **	arithmetic operators						
== != < > <= >=	relational operators						
and not or	logical operators						
& ~ ^ << >>	bit-wise operators						

Python Core Data Type

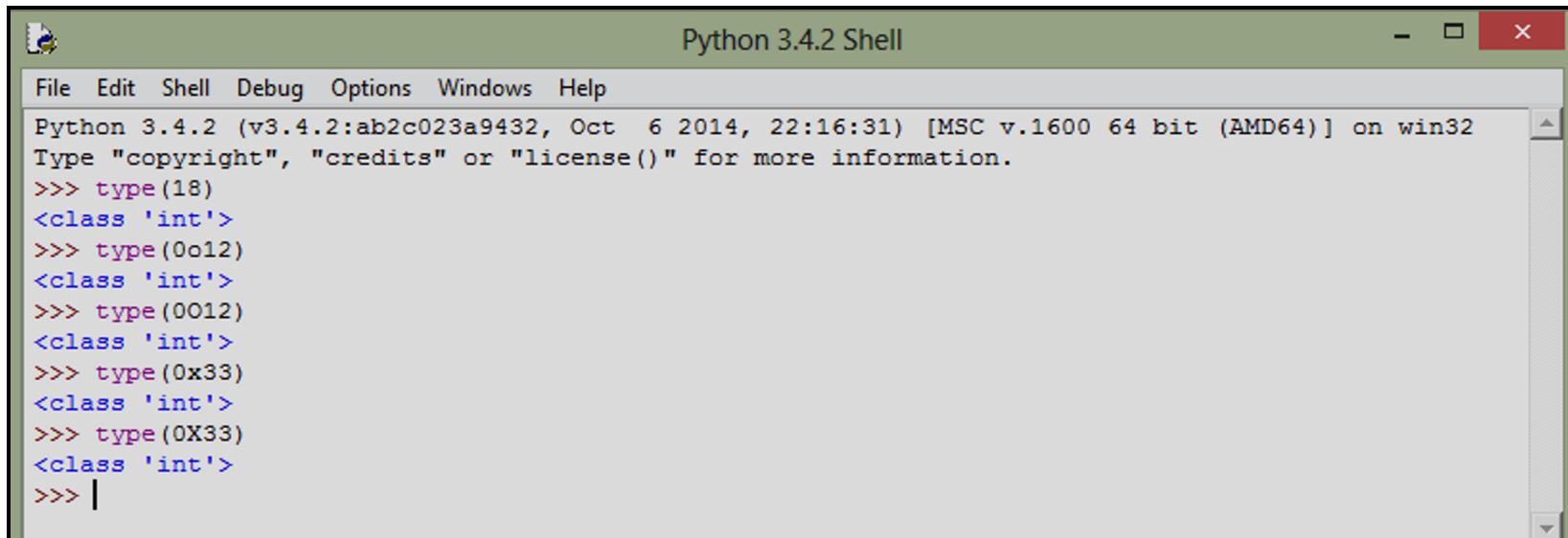
- In all the programming languages programmers would work with data, so data types is one of the crucial ingredient of any programming language.
- In Python, all the data types are represented in the form of objects; either built-in objects that Python provides, or objects that user creates using Python classes.
- The core data types provided by Python are:
- **Numeric types**
- Integers, floating point numbers and complex numbers falls under Python numbers category. They are defined as `int`, `float` and `complex` class in Python.
- You can use the `type()` function to know which class a variable or a value belongs.

Python Core Data Type

- **Integer**
- Type : int
- It includes following types of integer specifications:
 - Normal integers i.e 18
 - Octal literals (base 8)
 - an octal number can be defined with “0o”(Zero and alphabet o in upper or lower case) as a prefix i.e 0o12 or 0012
 - Hexadecimal literals (base 16)
 - Hexadecimal literals have to be prefixed either by “0x” or “0X”.
 - i.e. 0x33 or 0X33

Python Core Data Type

- **Integer**
- Type : int
- Use the **type()** function to know which class a variable or a value belongs



Python 3.4.2 Shell

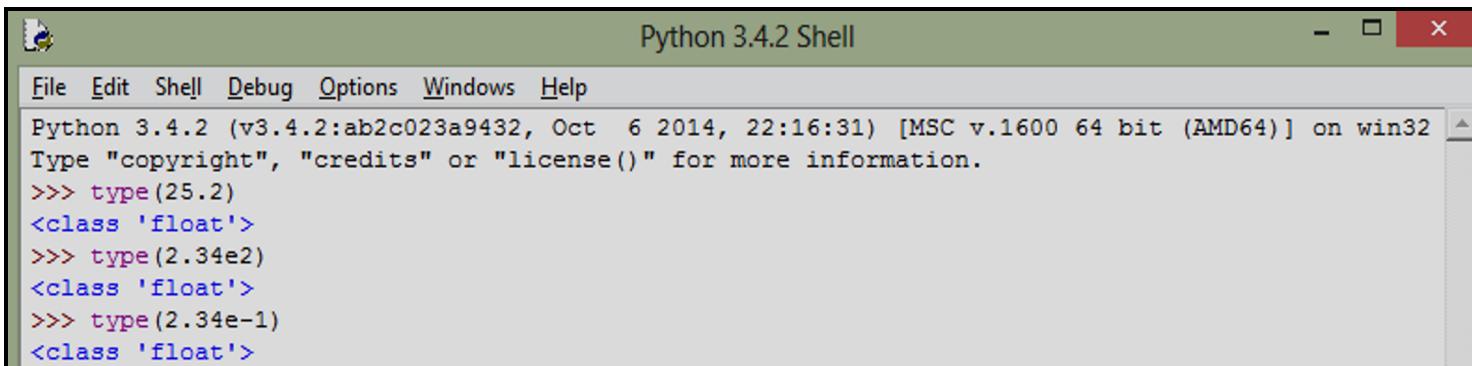
```
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.

>>> type(18)
<class 'int'>
>>> type(0o12)
<class 'int'>
>>> type(0012)
<class 'int'>
>>> type(0x33)
<class 'int'>
>>> type(0X33)
<class 'int'>
>>> |
```

Python Core Data Type

- **Floating-point numbers**
- e.g. 25.2
- Type : float
- Use the `type()` function to know which class a variable or a value belongs

Floating Point Number	Notation	Meaning
234.0	2.34e2	2.34×10^2
0.234	2.34e-1	2.34×10^{-1}



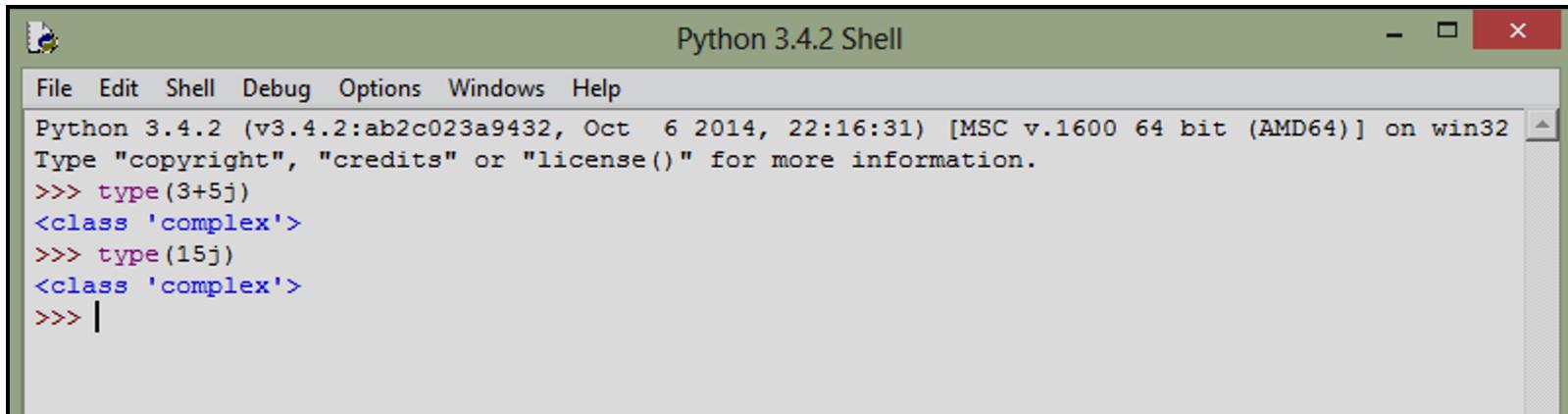
The screenshot shows the Python 3.4.2 Shell window. The title bar reads "Python 3.4.2 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Windows, and Help. The main window displays the Python version information and a few examples of using the `type()` function on different floating-point values:

```
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.

>>> type(25.2)
<class 'float'>
>>> type(2.34e2)
<class 'float'>
>>> type(2.34e-1)
<class 'float'>
```

Python Core Data Type

- **Complex numbers**
- Complex numbers are written as $<\text{real part}> + <\text{imaginary part}>j$
- i.e. $3+5j$
- i.e. $15j$
- Type: complex



The screenshot shows the Python 3.4.2 Shell window. The title bar reads "Python 3.4.2 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Windows, and Help. The main window displays the following text:

```
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.

>>> type(3+5j)
<class 'complex'>
>>> type(15j)
<class 'complex'>
>>> |
```

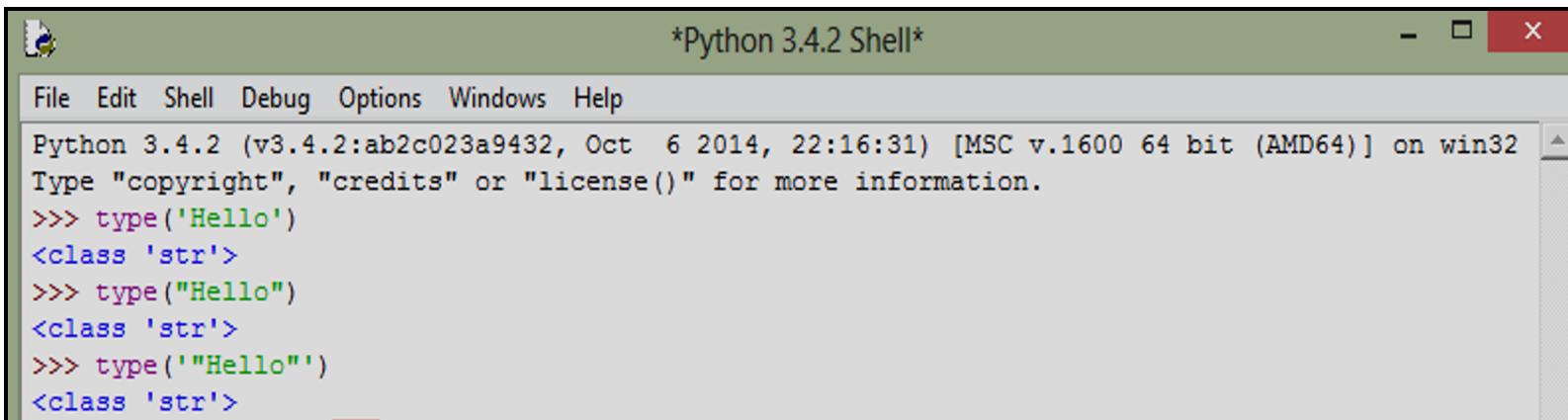


Python Core Data Type

- **Strings**
- Another important data type besides numbers are strings.
- Wrapped with the single-quote (') character:
'This is a string with single quotes'
- Wrapped with the double-quote (") character:
"This is a string with double quotes"
- Wrapped with three characters, using either single-quote or double-quote:
'''A String in triple quotes can extend over multiple lines like this one, and
can contain 'single' and "double" quotes.'''

Python Core Data Type

- Type : str
- There exists no character type in Python. A character is simply a string of size one.
- Use the `type()` function to know which class a variable or a value belongs

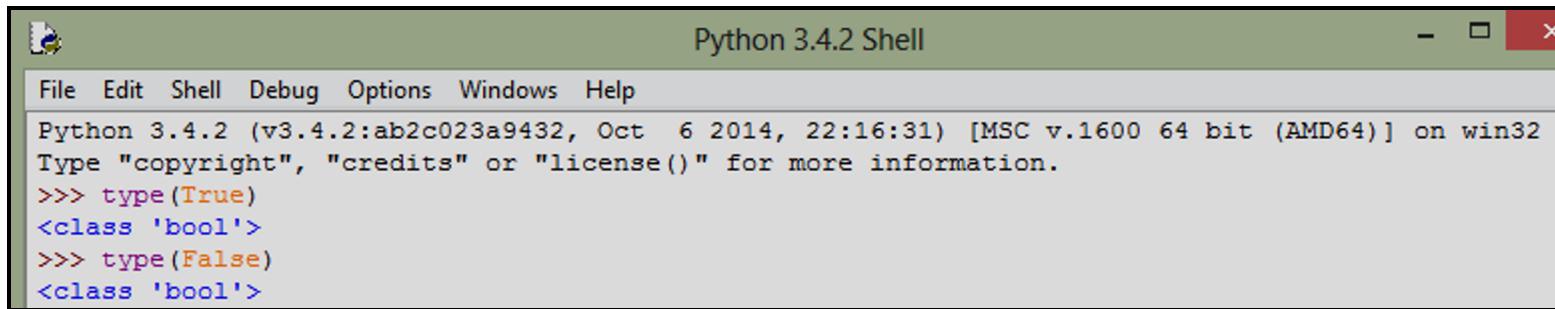


The screenshot shows a window titled "Python 3.4.2 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Windows, and Help. The main window displays Python 3.4.2 (v3.4.2:ab2c023a9432, Oct 6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on win32. It shows the following code and output:

```
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct 6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> type('Hello')
<class 'str'>
>>> type("Hello")
<class 'str'>
>>> type('"Hello"')
<class 'str'>
```

Python Core Data Type

- **Boolean**
- Type: bool for Boolean values
- i.e. True (represented as 1 internally)
- i.e. False (represented as 0 internally)



Python 3.4.2 Shell

```
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> type(True)
<class 'bool'>
>>> type(False)
<class 'bool'>
```

- In addition, Python supports a number of types that represent a collection of values -including strings, lists and dictionaries.



Assigning Value to Variable

- Python is dynamically typed meaning that no pre-declaration of a variable or its type is necessary. The type (and value) are initialized on assignment. Assignments are performed using the equal(=) sign.
- A variable is introduced by assigning a value to it.
- Example: x=20
- A variable that has been assigned a value of a given type may later be assigned a value of a different type.
- Example: x=30.42
- A variable may be assigned and reassigned as often as necessary. The type of a variable will change if it is reassigned an expression of a different type.
- Python is case-sensitive, meaning that the identifier "cAsE" is different from "CaSe."

Assigning Value to Variable

- Python also supports *augmented assignment, statements that both refer to and assign values to variables*. You can take the following expression
- $n = n * 10$ and use this shortcut instead: $n *= 10$
- **Multiple Assignment**
- Python allows you to assign a single value or multiple values to several variables simultaneously.

```
>>> x, y = 2, 3
```

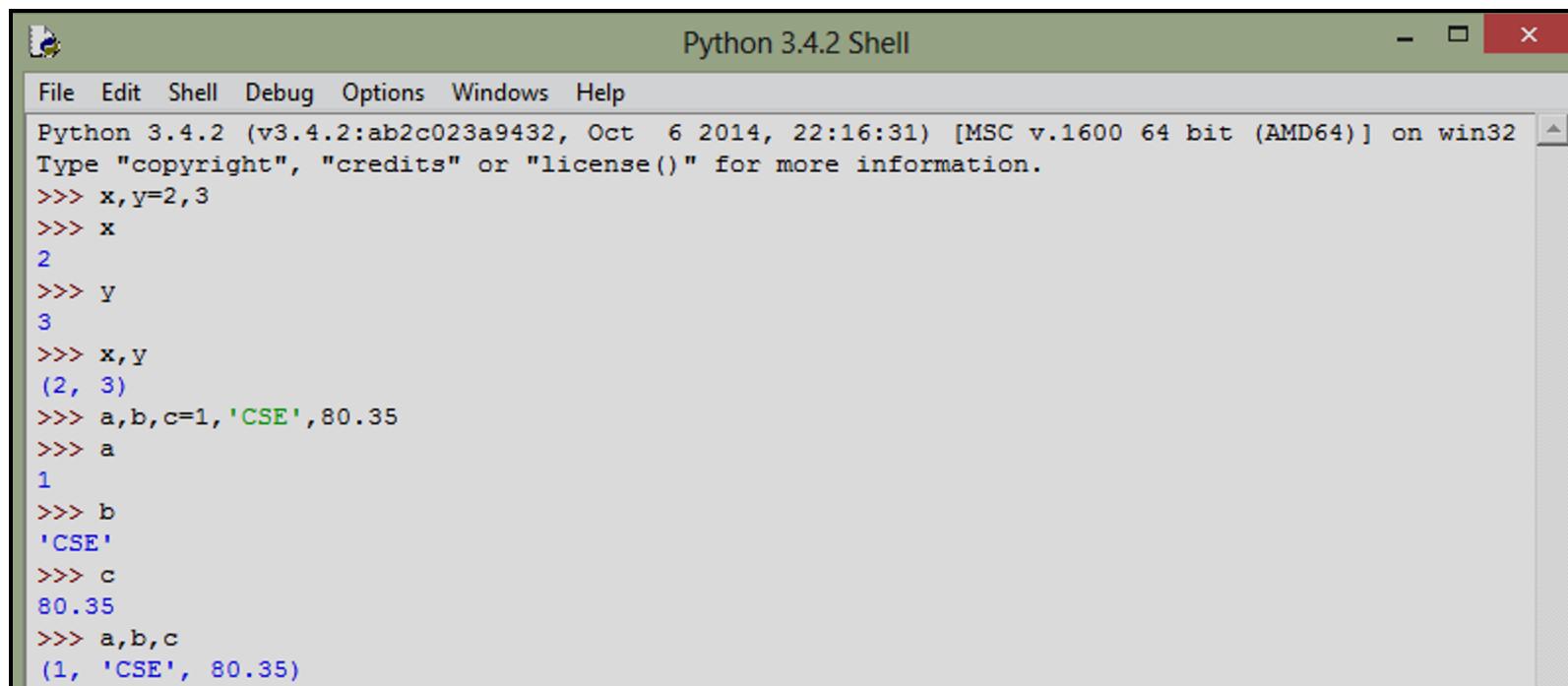
```
>>> x
```

```
2
```

```
>>> y
```

```
3
```

Assigning Value to Variable



Python 3.4.2 Shell

```
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.

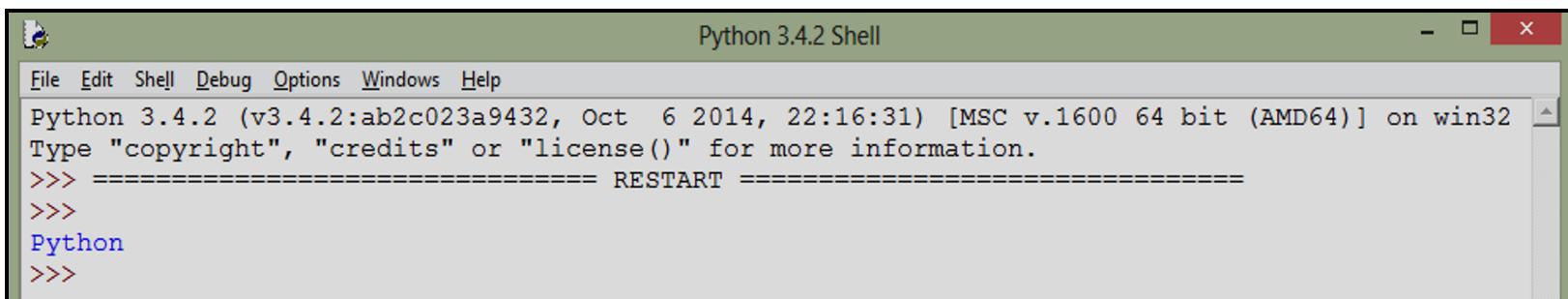
>>> x,y=2,3
>>> x
2
>>> y
3
>>> x,y
(2, 3)
>>> a,b,c=1,'CSE',80.35
>>> a
1
>>> b
'CSE'
>>> c
80.35
>>> a,b,c
(1, 'CSE', 80.35)
```

Comments

- Comments are the non-executable statements in a program. They are just added to describe the statements in the program code. Comments make the program easily readable and understandable by the programmer as well as other users who are seeing the code. The interpreter simply ignores the comments.
- In Python, a hash sign (#) and triple quotation marks is used for comments.

```
#this is python comment
print("Python")
#End of program
```

```
'''This is python comment
Comments make the program easily readable'''
print("Python")
#End of program
```

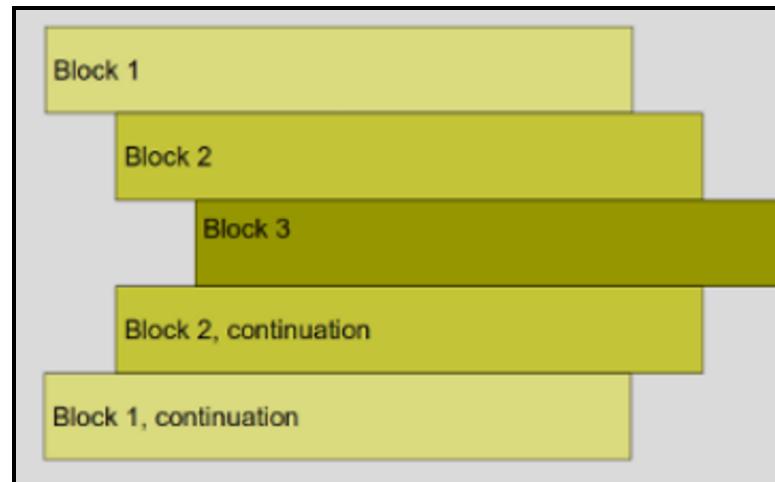


A screenshot of the Python 3.4.2 Shell window. The title bar reads "Python 3.4.2 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Windows, and Help. The main window displays the following text:

```
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Python
>>>
```

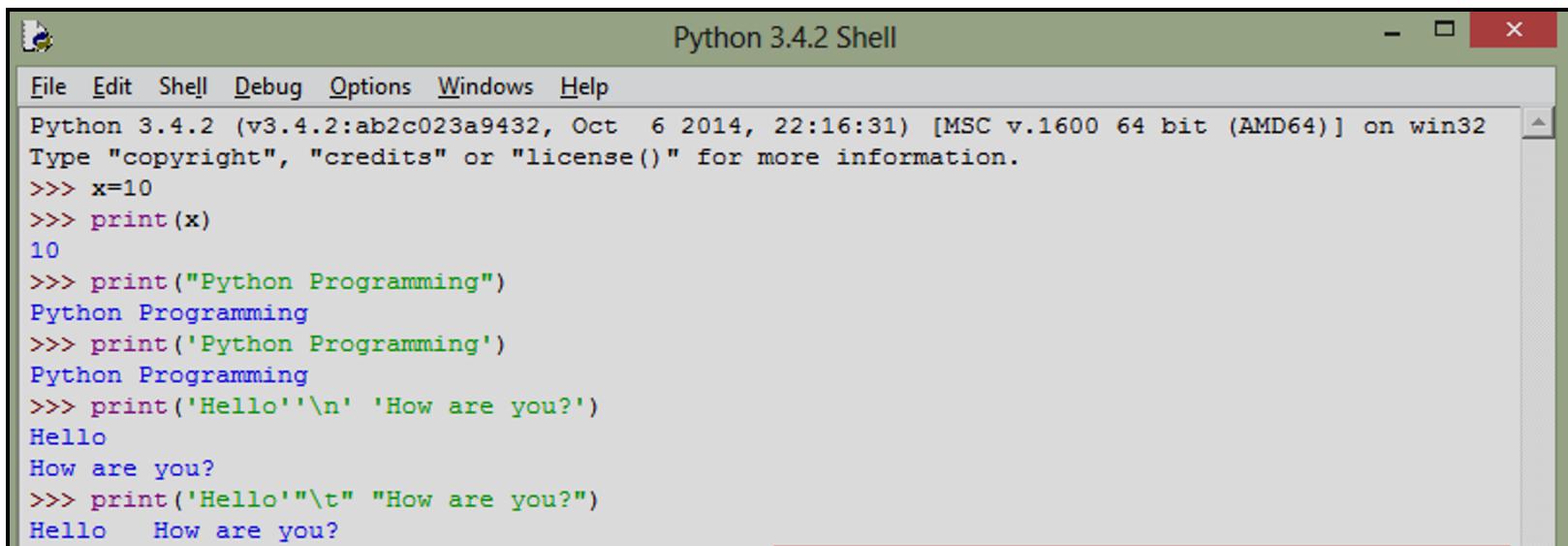
Indenting Code

- Python programs get structured through indentation, i.e. code blocks are defined by their indentation. This principle makes it easier to read and understand other people's Python code.
- All statements with the same distance to the right belong to the same block of code, i.e. the statements within a block line up vertically. The block ends at a line less indented or the end of the file. If a block has to be more deeply nested, it is simply indented further to the right.



Program Output, print() function

- The programmer can print the output of a value by using the **print** function. The simplest form for using this function looks like the following:
- **print(argument)**
- The argument can be a value of any type int, str, float etc. It can also be a value stored in a variable.



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> x=10
>>> print(x)
10
>>> print("Python Programming")
Python Programming
>>> print('Python Programming')
Python Programming
>>> print('Hello'\n' How are you?')
Hello
How are you?
>>> print('Hello'\t" "How are you?")
Hello      How are you?
```



Program Input, `input()`function

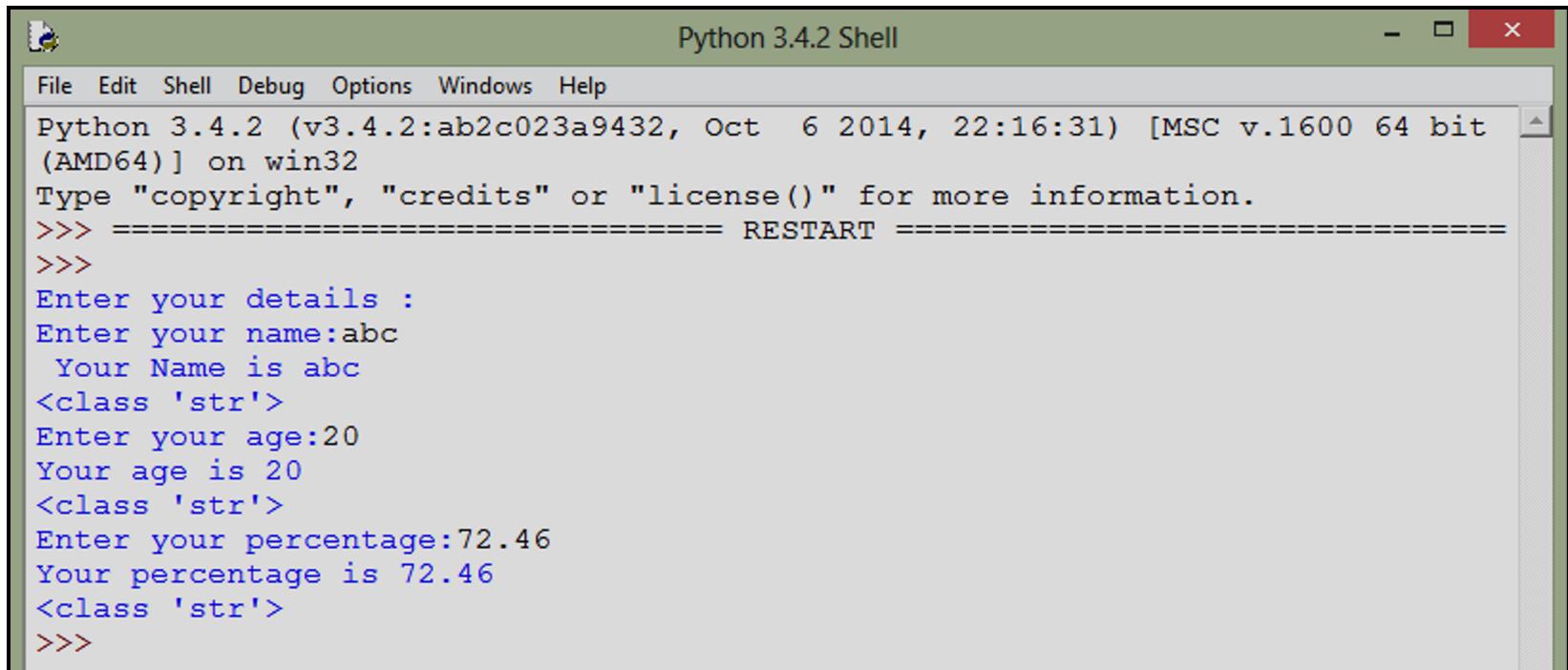
- The input function is a simple way for your program to get information from people using your program.
- The basic structure is
- `Variable_name = input(message to user)`
- **Or**
- `Variable_name = input('string')`
- The above works for getting text from the user. i.e. `input()` function produces only string. A programmer can make use of any type to convert the string into a specific type.
- i.e. `x= int(input())`
- i.e. `y= float(input())`



Program Input, input()function

```
#Python Program  
print('Enter your details :')  
name=input("Enter your name:")  
print(" Your Name is",name)  
print(type(name))  
age=input("Enter your age:")  
print("Your age is",age)  
print(type(age))  
per=input("Enter your percentage:")  
print("Your percentage is",per)  
print(type(per))  
#End of Program
```

Program Input, input()function



Python 3.4.2 Shell

```
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit
(AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Enter your details :
Enter your name:abc
Your Name is abc
<class 'str'>
Enter your age:20
Your age is 20
<class 'str'>
Enter your percentage:72.46
Your percentage is 72.46
<class 'str'>
>>>
```



Program Input, input()function

```
#Python Program  
print('Enter your details :')  
name=input("Enter your name:")  
print(" Your Name is",name)  
print(type(name))  
age=int(input("Enter your age:"))  
print("Your age is",age)  
print(type(age))  
per=float(input("Enter your percentage:"))  
print("Your percentage is",per)  
print(type(per))
```

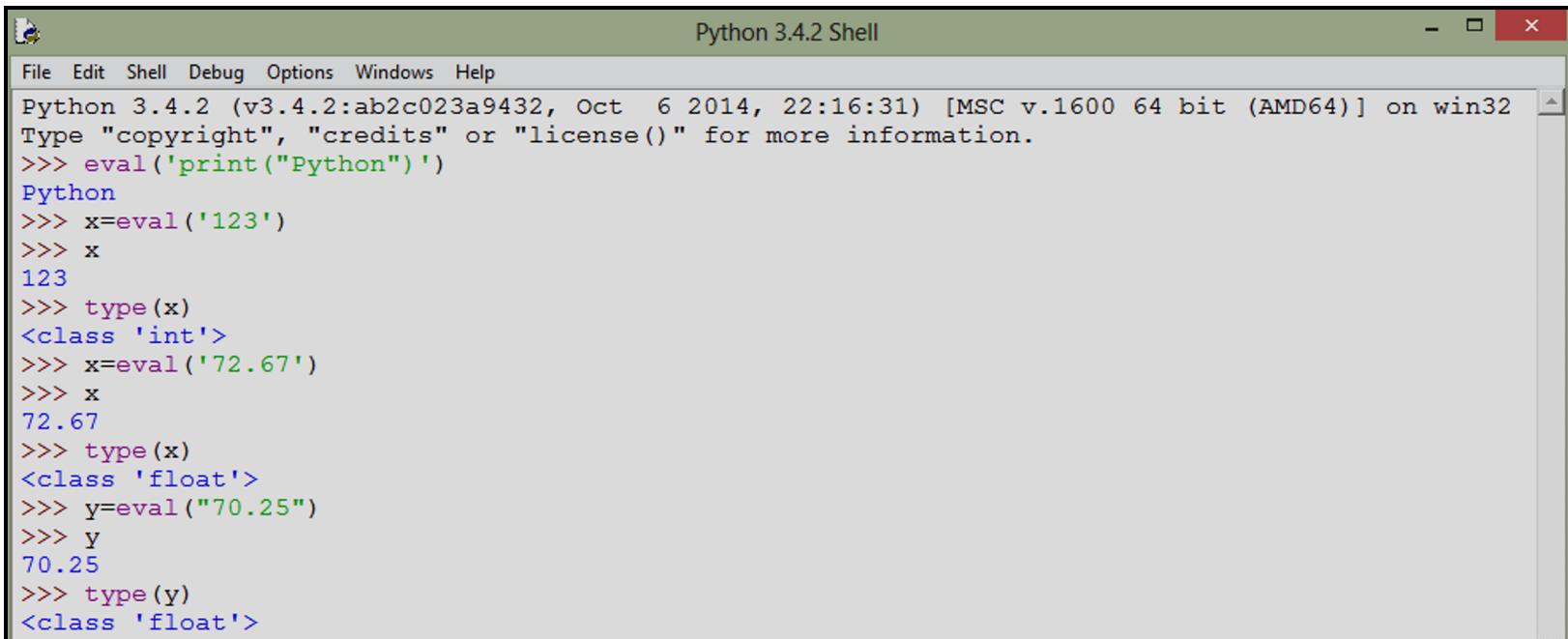
Program Input, input()function

```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct 6 2014, 22:16:31) [MSC v.1600 64 bit
(AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Enter your details :
Enter your name:abc
Your Name is abc
<class 'str'>
Enter your age:20
Your age is 20
<class 'str'>
Enter your percentage:72.46
Your percentage is 72.46
<class 'str'>
>>> ===== RESTART =====
>>>
Enter your details :
Enter your name:abc
Your Name is abc
<class 'str'>
Enter your age:20
Your age is 20
<class 'int'>
Enter your percentage:72.46
Your percentage is 72.46
<class 'float'>
>>> |
```



The eval() Function

- The full form of eval function is to evaluate. It takes a string as parameter and returns it as if it is a python expression.
- The eval function takes a string and returns it in the type it is expected.
- The eval() method returns the result evaluated from the expression.



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> eval('print("Python") ')
Python
>>> x=eval('123')
>>> x
123
>>> type(x)
<class 'int'>
>>> x=eval('72.67')
>>> x
72.67
>>> type(x)
<class 'float'>
>>> y=eval("70.25")
>>> y
70.25
>>> type(y)
<class 'float'>
```

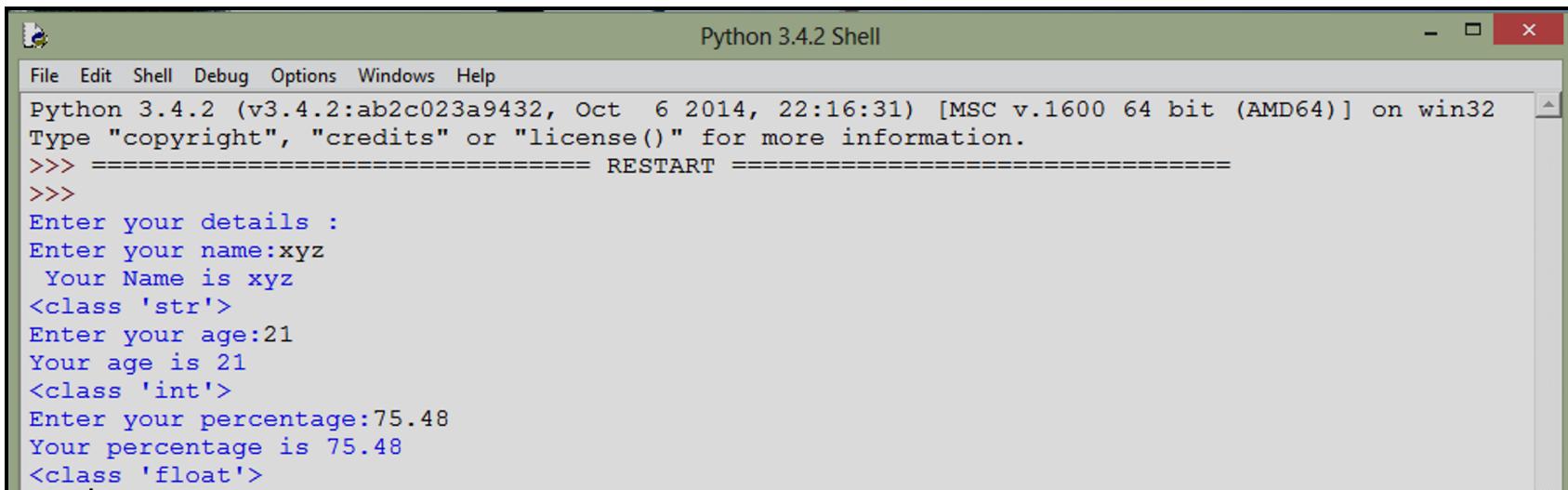
The eval() Function

- **Apply eval() to input() function**

```
#Python Program
print('Enter your details :')
name=input("Enter your name:")
print(" Your Name is",name)
print(type(name))
age=eval(input("Enter your age:"))
print("Your age is",age)
print(type(age))
per=eval(input("Enter your percentage:"))
print("Your percentage is",per)
print(type(per))
#End of Program
```

The eval() Function

Use of eval() function avoids specifying a particular type (i.e. int,float)in front of input() function.



Python 3.4.2 Shell

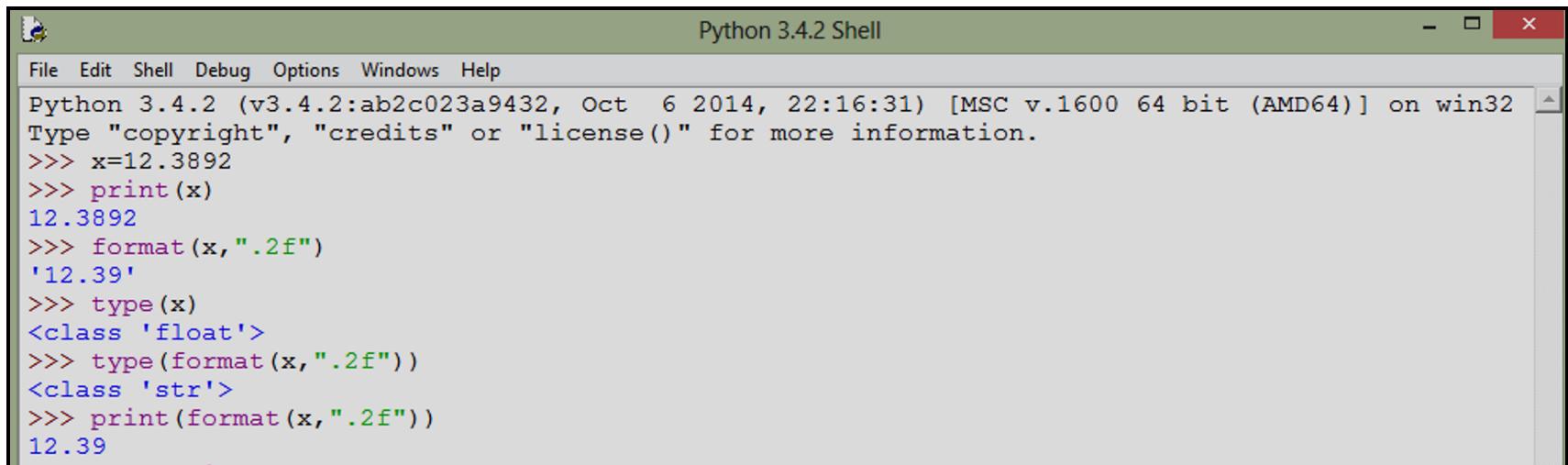
```
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Enter your details :
Enter your name:xyz
Your Name is xyz
<class 'str'>
Enter your age:21
Your age is 21
<class 'int'>
Enter your percentage:75.48
Your percentage is 75.48
<class 'float'>
```

Formatting Number and Strings

- Often it is desirable to display numbers in a certain format. You can use the format function to return a formatted string.
- The syntax to invoke this function is

format(item, format-specifier)

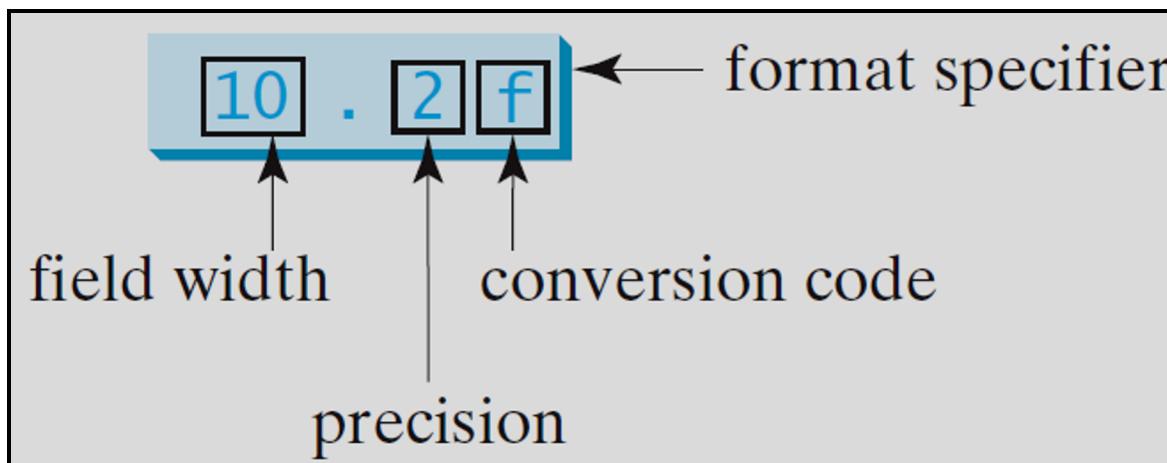
- where item is a number or a string and format-specifier is a string that specifies how the item is formatted. The function returns a string.



```
Python 3.4.2 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> x=12.3892
>>> print(x)
12.3892
>>> format(x,".2f")
'12.39'
>>> type(x)
<class 'float'>
>>> type(format(x,".2f"))
<class 'str'>
>>> print(format(x,".2f"))
12.39
```

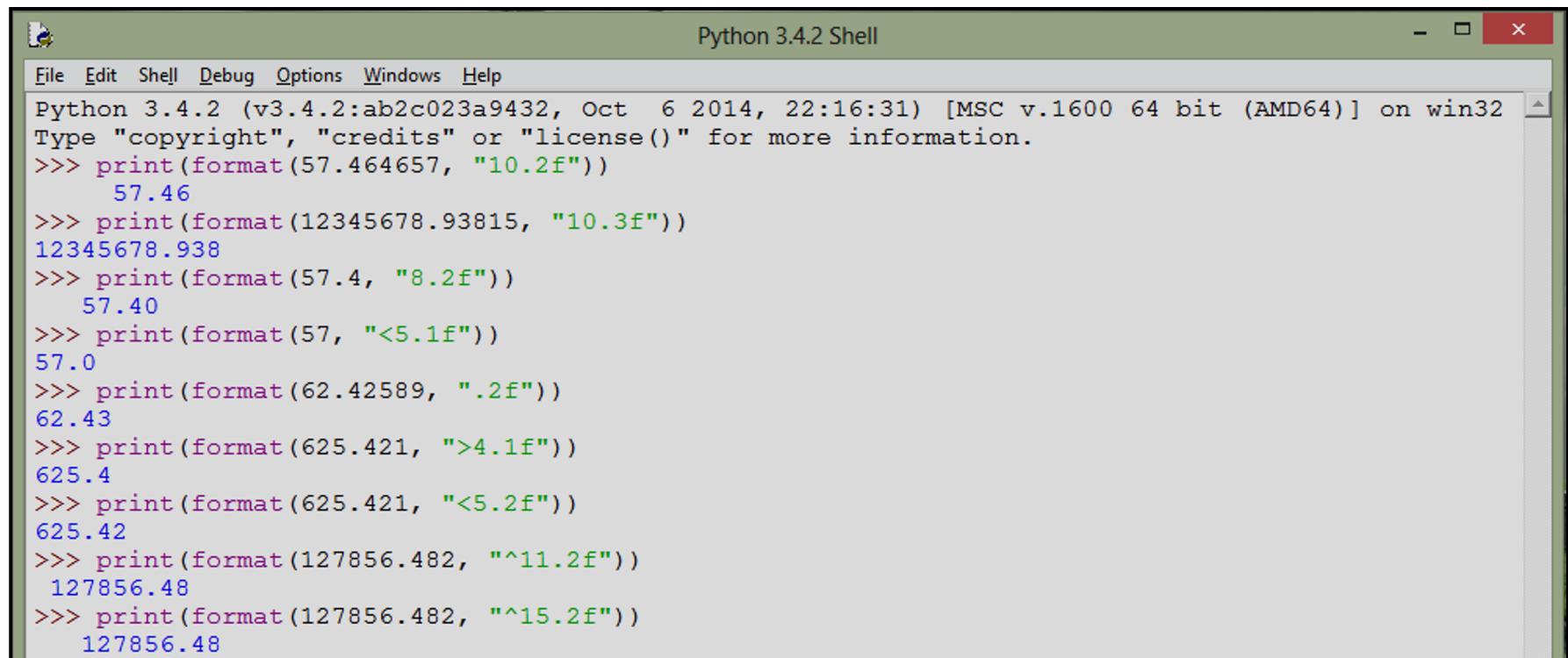
Formatting Number and Strings

- **Formatting Floating-Point Numbers**
- If the item is a float value, you can use the specifier to give the width and precision of the format in the form of **width.precisionf**. Here, width specifies the width of the resulting string, precision specifies the number of digits after the decimal point, and f is called the conversion code, which sets the formatting for floating point numbers.
- You can omit the width specifier. If so, it defaults to 0. In this case, the width is automatically set to the size needed for formatting the number.



Formatting Number and Strings

The `format("10.2f")` function formats the number into a string whose width is 10, including a decimal point and two digits after the point. If there are fewer digits before the decimal point, spaces are inserted before the number. If there are more digits before the decimal point, the number's width is automatically increased.

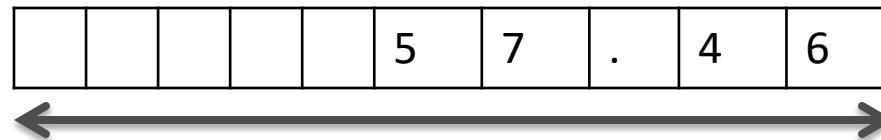


```
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.

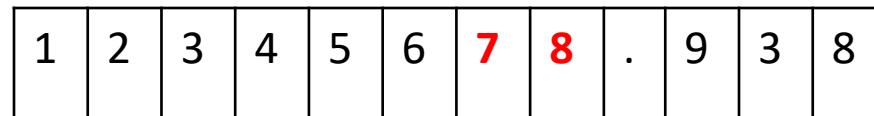
>>> print(format(57.464657, "10.2f"))
      57.46
>>> print(format(12345678.93815, "10.3f"))
12345678.938
>>> print(format(57.4, "8.2f"))
   57.40
>>> print(format(57, "<5.1f"))
  57.0
>>> print(format(62.42589, ".2f"))
 62.43
>>> print(format(625.421, ">4.1f"))
625.4
>>> print(format(625.421, "<5.2f"))
625.42
>>> print(format(127856.482, "^11.2f"))
127856.48
>>> print(format(127856.482, "^15.2f"))
127856.48
```

Formatting Number and Strings

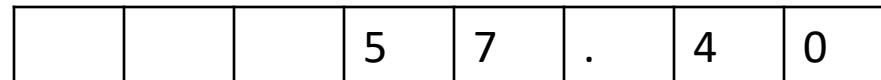
- 57.464657 (10.2f)



- 12345678.93815 (10.3f)



- 57.4 (8.2f)



where a square box () denotes a blank space.

Note that the decimal point is counted as one space.

Formatting Number and Strings

- 57 (<5.1f)
- By default, the number is right justified. You can insert < in the format specifier to specify an item to be left justified.

5	7	.	1	
---	---	---	---	--

- 625.421 (>4.1f)
- By default, the number is right justified. You can specify > in the format to specify right justification.

6	2	5	.	4
---	---	---	---	---

Formatting Number and Strings

- 127856.482 (^11.2f) (^ for center align)

	1	2	7	8	5	6	.	4	8	
--	---	---	---	---	---	---	---	---	---	--

- 127856.482 (^15.2f) (^ for center align)

			1	2	7	8	5	6	.	4	8		
--	--	--	---	---	---	---	---	---	---	---	---	--	--

Formatting Number and Strings

- **Formatting in Scientific Notation**
- If you change the conversion code from f to e, the number will be formatted in scientific notation.
- For example,
- a) 57.467657 format 10.2e

5.75×10^1

i.e. 5.75×10^1

		5	.	7	5	e	+	0	1
--	--	---	---	---	---	---	---	---	---

- b) 2173.0 format 8.3e

2.173×10^3

2.173×10^3

2	.	1	7	3	e	+	0	3
---	---	---	---	---	---	---	---	---

Formatting Number and Strings

- **Formatting in Scientific Notation**

- c) 0.000716 format 9.2e

7.16×10^{-4}

7.16×10^{-4}

	7	.	1	6	e	-	0	4
--	---	---	---	---	---	---	---	---

- d) 0.00051 format 7.1e

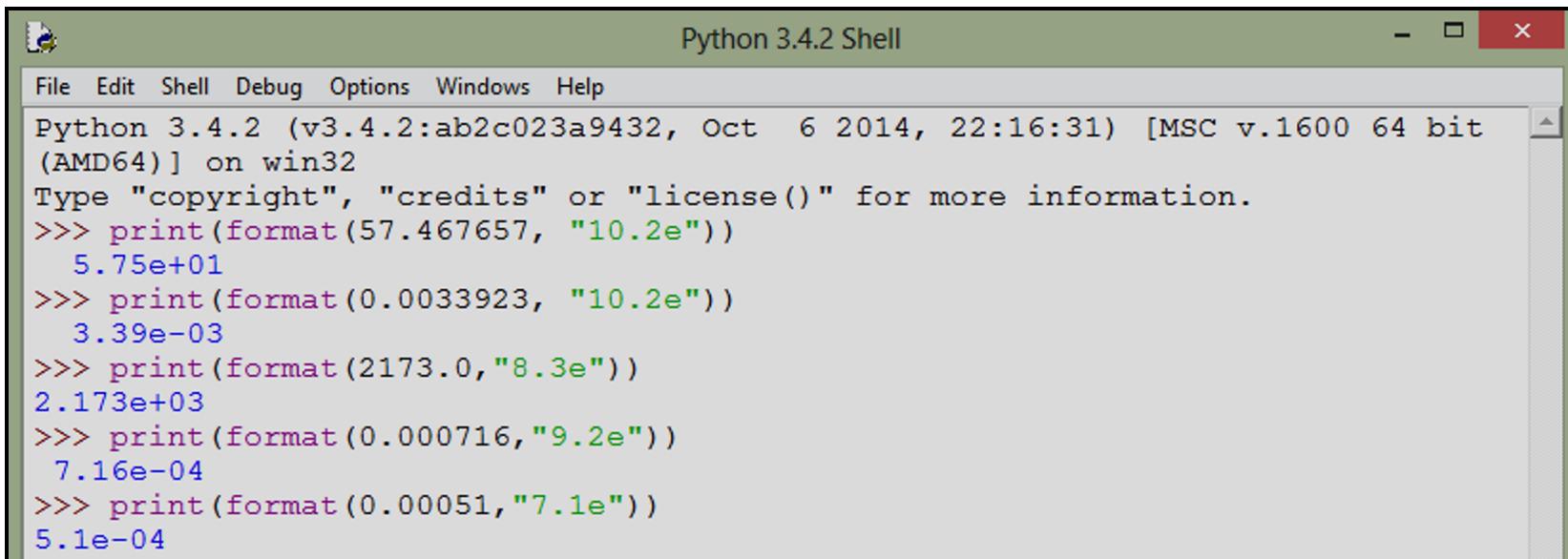
5.1×10^{-4}

5.1×10^{-4}

5	.	1	e	-	0	4
---	---	---	---	---	---	---

Formatting Number and Strings

- **Formatting in Scientific Notation**



Python 3.4.2 Shell

```
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit
(AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print(format(57.467657, "10.2e"))
5.75e+01
>>> print(format(0.0033923, "10.2e"))
3.39e-03
>>> print(format(2173.0,"8.3e"))
2.173e+03
>>> print(format(0.000716,"9.2e"))
7.16e-04
>>> print(format(0.00051,"7.1e"))
5.1e-04
```

Formatting Number and Strings

- **Formatting as a Percentage**
- You can use the conversion code % to format a number as a percentage.
- The format causes the number to be multiplied by 100 and displayed with a % sign following it. The total width includes the % sign counted as one space.
- a) 0.0033923, "10.2%"

0.34%

					0	.	3	4	%
--	--	--	--	--	---	---	---	---	---

- b) 0.0033923, "6.3%"

0.339%

0	.	3	3	9	%
---	---	---	---	---	---

- c) 7.4, "4.2%"

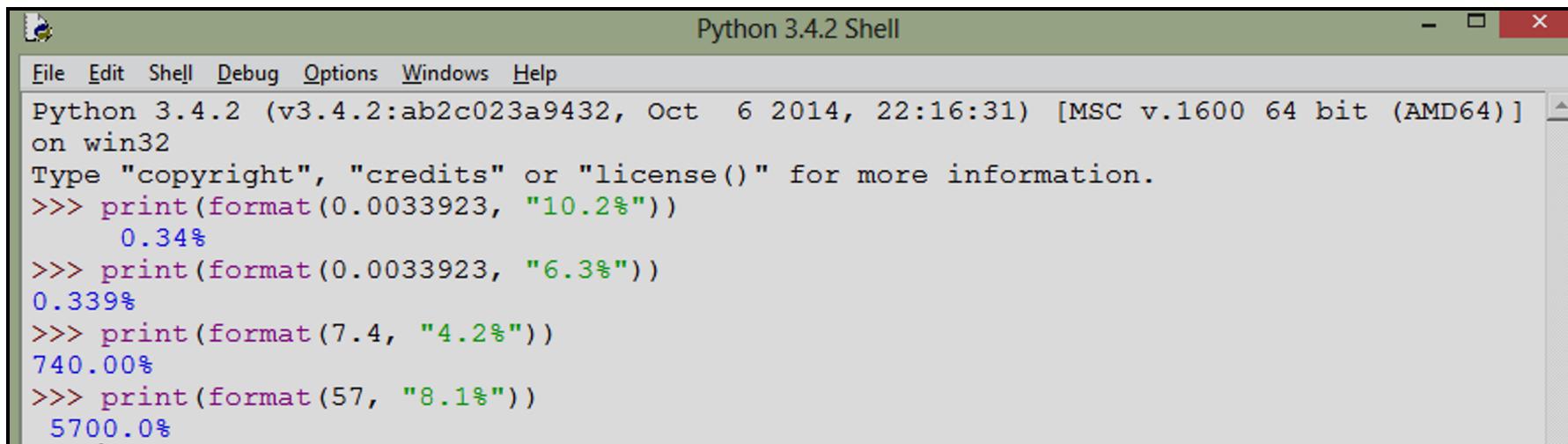
740.00%

7	4	0	.	0	0	%
---	---	---	---	---	---	---

Formatting Number and Strings

- **Formatting as a Percentage**
- 57, "8.1%"
- 5700.0%

	5	7	0	0	.	0	%
--	---	---	---	---	---	---	---



Python 3.4.2 Shell

```
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> print(format(0.0033923, "10.2%"))
    0.34%
>>> print(format(0.0033923, "6.3%"))
0.339%
>>> print(format(7.4, "4.2%"))
740.00%
>>> print(format(57, "8.1%"))
5700.0%
```

Formatting Number and Strings

- **Formatting Integers**
- The conversion codes d, x, o, and b can be used to format an integer in decimal, hexadecimal, octal, or binary. You can specify a width for the conversion.
- **Decimal to Binary Conversion**
- e.g. Convert 256 to binary

DIVISION	RESULT	REMAINDER
256/2	128	0
128/2	64	0
64/2	32	0
32/2	16	0
16/2	8	0
8/2	4	0
4/2	2	0
2/2	1	0
1/2	0	1
Answer		100000000

Formatting Number and Strings

- **Decimal to Binary Conversion**

- 256,"8b"

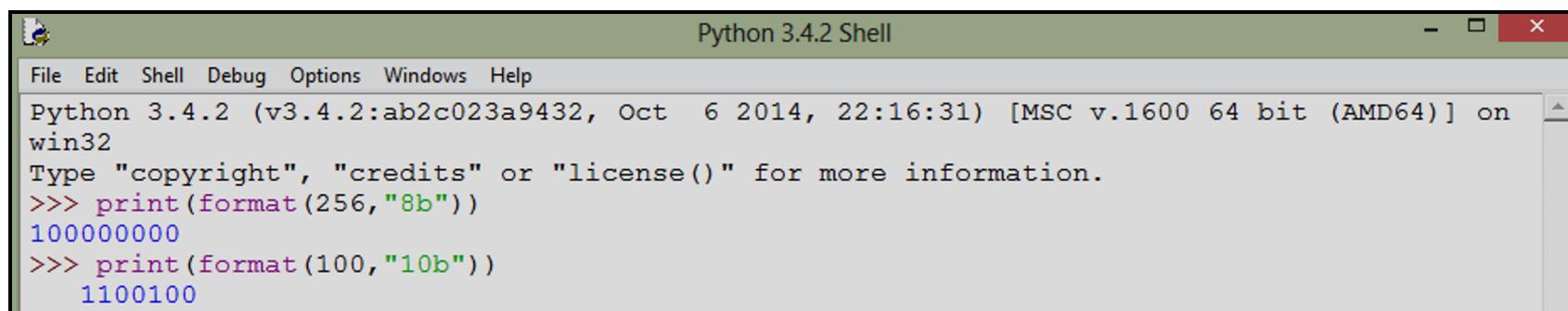
1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---

- 100000000

- 100,"10b"

- 1100100

			1	1	0	0	1	0	0
--	--	--	---	---	---	---	---	---	---



Python 3.4.2 Shell

```
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on
win32
Type "copyright", "credits" or "license()" for more information.
>>> print(format(256,"8b"))
100000000
>>> print(format(100,"10b"))
1100100
```

Formatting Number and Strings

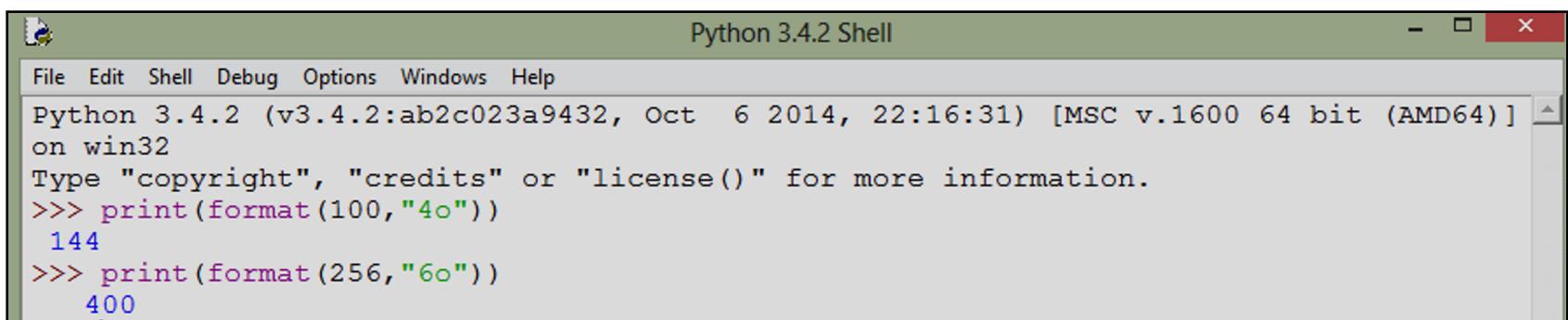
- **Decimal to Octal conversion**

- $100, "4o"$
- 144

	1	4	4
--	---	---	---

- $256, "6o"$
- 400

			4	0	0
--	--	--	---	---	---



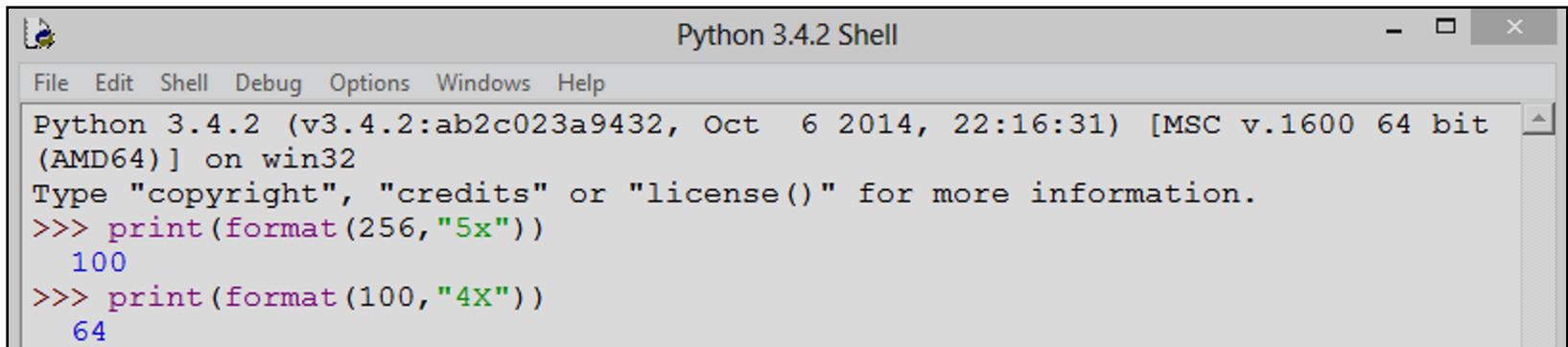
Python 3.4.2 Shell

```
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> print(format(100,"4o"))
144
>>> print(format(256,"6o"))
400
```

Formatting Number and Strings

- **Formatting Integers**
- For example, **Decimal to Hexadecimal** conversion

DIVISION	RESULT	REMAINDER (HEX)
$100 / 16$	6	4
$6 / 16$	0	6
ANSWER		64

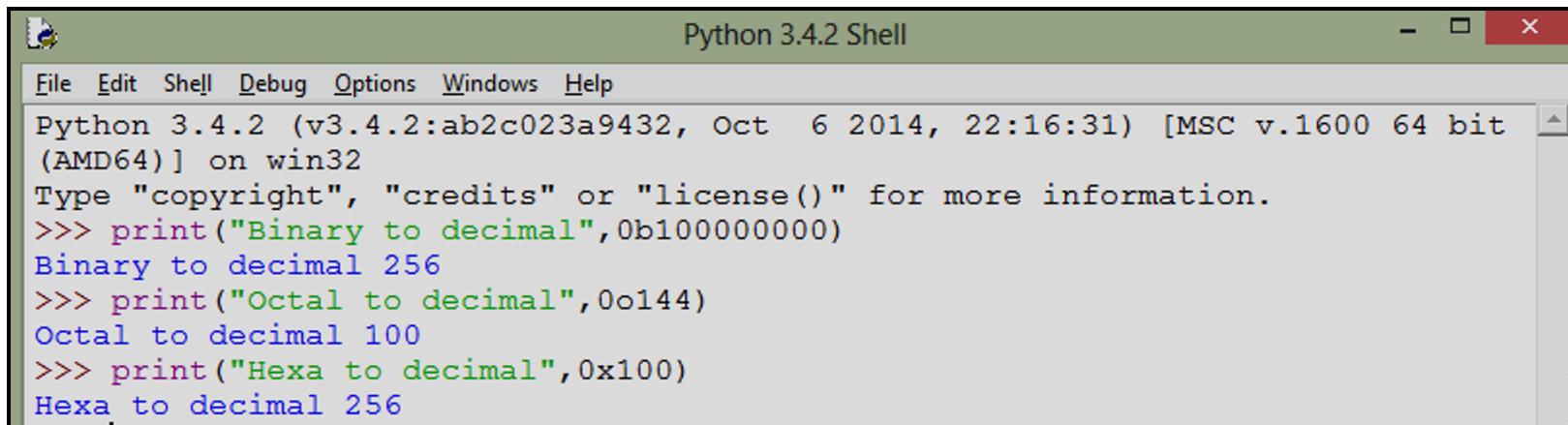


Python 3.4.2 Shell

```
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit
(AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print(format(256, "5x"))
 100
>>> print(format(100, "4X"))
 64
```

Formatting Number and Strings

- **Conversions**
- Decimal to Binary, octal and Hexadecimal conversions



Python 3.4.2 Shell

```
File Edit Shell Debug Options Windows Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit
(AMD64) ] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Binary to decimal",0b100000000)
Binary to decimal 256
>>> print("Octal to decimal",0o144)
Octal to decimal 100
>>> print("Hexa to decimal",0x100)
Hexa to decimal 256
```

Formatting Number and Strings

- **Formatting Strings**
- You can use the conversion code **s** to format a string with a specified width.
- By default, a string is left justified.
- To right-justify it, put the symbol **>** in the format specifier.
- If the string is longer than the specified width, the width is automatically increased to fit the string.
- e.g.
- Format the string :
- a) Welcome to Python, 20s

W	e		c	o	m	e	t	o	P	y	t	h	o	n			
---	---	--	---	---	---	---	---	---	---	---	---	---	---	---	--	--	--

Formatting Number and Strings

- b) Jaspreet Singh, 14s

J	a	s	p	r	e	e	t		S	i	n	g	h
---	---	---	---	---	---	---	---	--	---	---	---	---	---

- c) Jaspreet Singh, >16s (Right Justify)

		J	a	s	p	r	e	e	t		S	i	n	g	h
--	--	---	---	---	---	---	---	---	---	--	---	---	---	---	---

- d) Python and Java support portability, >25s

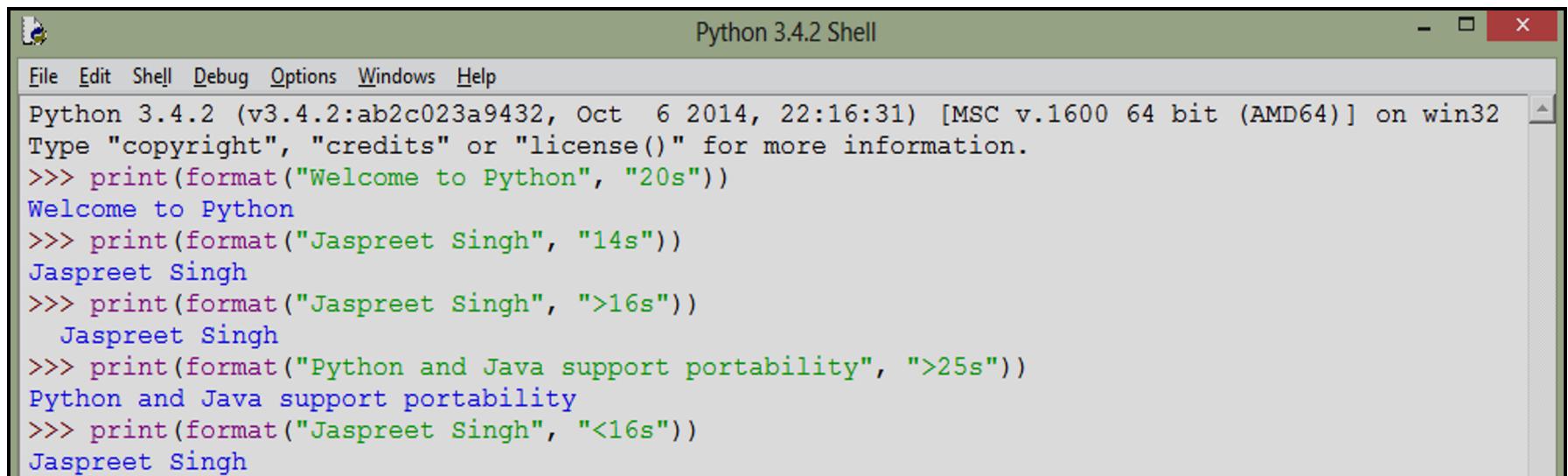
P	y	t	h	o	n		a	n	d		J	a	v	a		S	u	p	p	o	r	t		P	o	r	t	a	b	i	l	i	t	y
---	---	---	---	---	---	--	---	---	---	--	---	---	---	---	--	---	---	---	---	---	---	---	--	---	---	---	---	---	---	---	---	---	---	---

- e) Jaspreet Singh, <16s (Left Justify)

J	a	s	p	r	e	e	t		S	i	n	g	h		
---	---	---	---	---	---	---	---	--	---	---	---	---	---	--	--

Formatting Number and Strings

Formatting Strings



The screenshot shows a Windows application window titled "Python 3.4.2 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Windows, and Help. The main window displays Python code and its output. The code demonstrates various string formatting techniques using the `format` function:

```
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.

>>> print(format("Welcome to Python", "20s"))
Welcome to Python
>>> print(format("Jaspreet Singh", "14s"))
Jaspreet Singh
>>> print(format("Jaspreet Singh", ">16s"))
   Jaspreet Singh
>>> print(format("Python and Java support portability", ">25s"))
Python and Java support portability
>>> print(format("Jaspreet Singh", "<16s"))
Jaspreet Singh
```

Formatting Number and Strings

Frequently Used Specifiers

<i>Specifier</i>	<i>Format</i>
" 10.2f "	Format the float item with width 10 and precision 2.
" 10.2e "	Format the float item in scientific notation with width 10 and precision 2.
" 5d "	Format the integer item in decimal with width 5.
" 5x "	Format the integer item in hexadecimal with width 5.
" 5o "	Format the integer item in octal with width 5.
" 5b "	Format the integer item in binary with width 5.
" 10.2% "	Format the number in decimal.
" 50s "	Format the string item with width 50.
" <10.2f "	Left-justify the formatted item.
" >10.2f "	Right-justify the formatted item.



THANKS.....