input

- `input` : **Reads a string from user input.**
  - You can assign (store) the result of `input` into a variable.
  - Example:

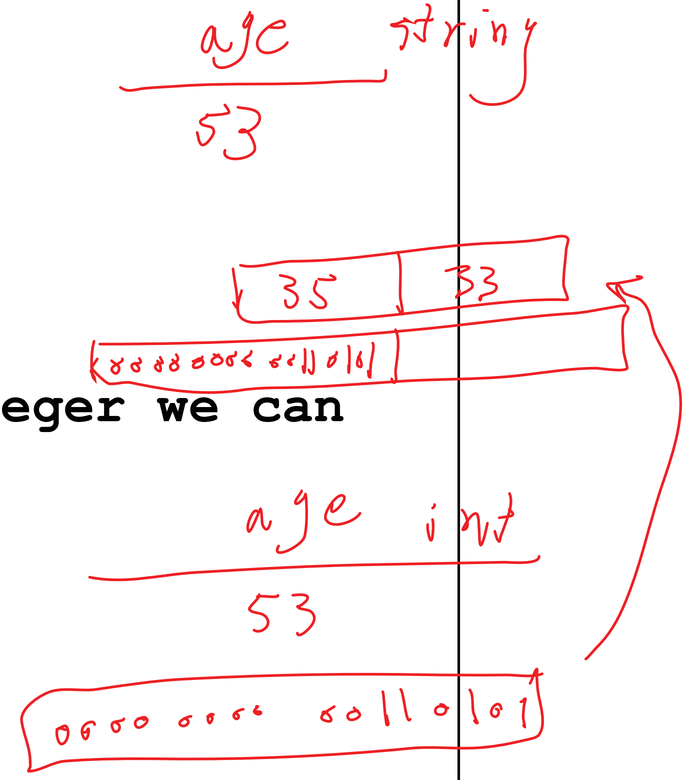
```
age = input("How old are you? ")
print ("Your age is", age)
```

Output:

```
How old are you? 53
Your age is 53
```

- **For converting a string to an integer we can use int type conversion command**

```
age = int(input("How old are you? "))
age=age+1
print ("Your age plus one is", age)
```



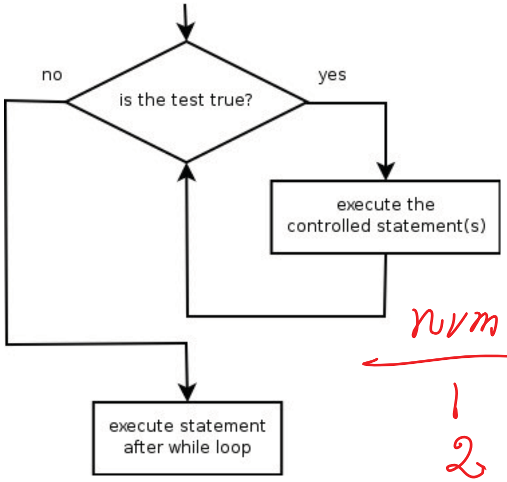
while

- **`while` loop: Executes a group of statements as long as a condition is True.**
  - good for *indefinite loops* (repeat an unknown number of times)

- **Syntax:**

```
while condition:
    statements
```

block



no

is the test true?

yes

execute the controlled statement(s)

execute statement after while loop

- **Example:**

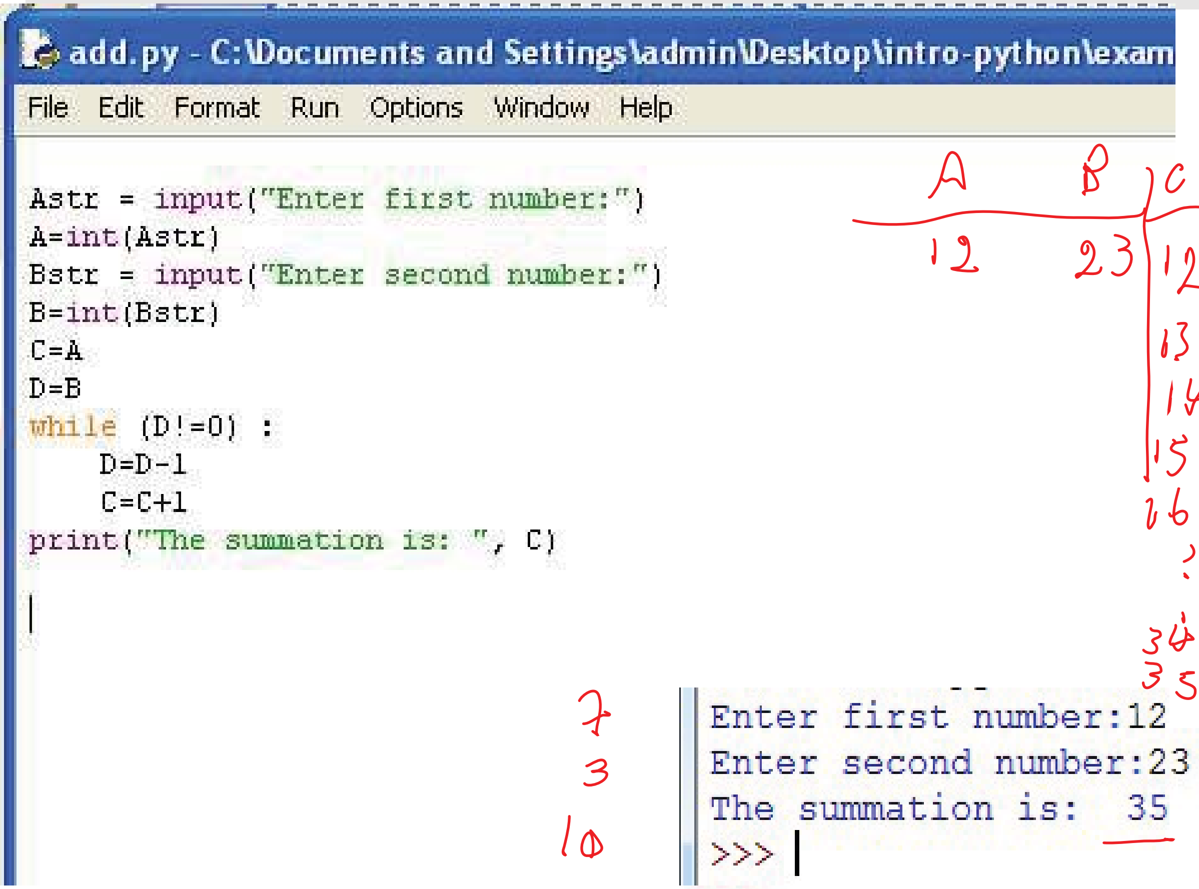
```
number = 1
while number < 200:
    print (number)
    number = number * 2
```

number
1
2
4
8
16
32
64
128
256

- **Output:**

```
1 2 4 8 16 32 64 128
```

Add Example


add.py - C:\Documents and Settings\admin\Desktop\intro-python\exam
File Edit Format Run Options Window Help

```
Astr = input("Enter first number:")
A=int(Astr)
Bstr = input("Enter second number:")
B=int(Bstr)
C=A
D=B
while (D!=0) :
     D=D-1
     C=C+1
print("The summation is: ", C)
```

```
Enter first number:12
Enter second number:23
The summation is:  35
>>>
```

A	B	C	D
12	23	12	23
		13	22
		14	21
		15	20
		16	19
		⋮	
		34	1
		35	0

A + B

A + 1

A − 1

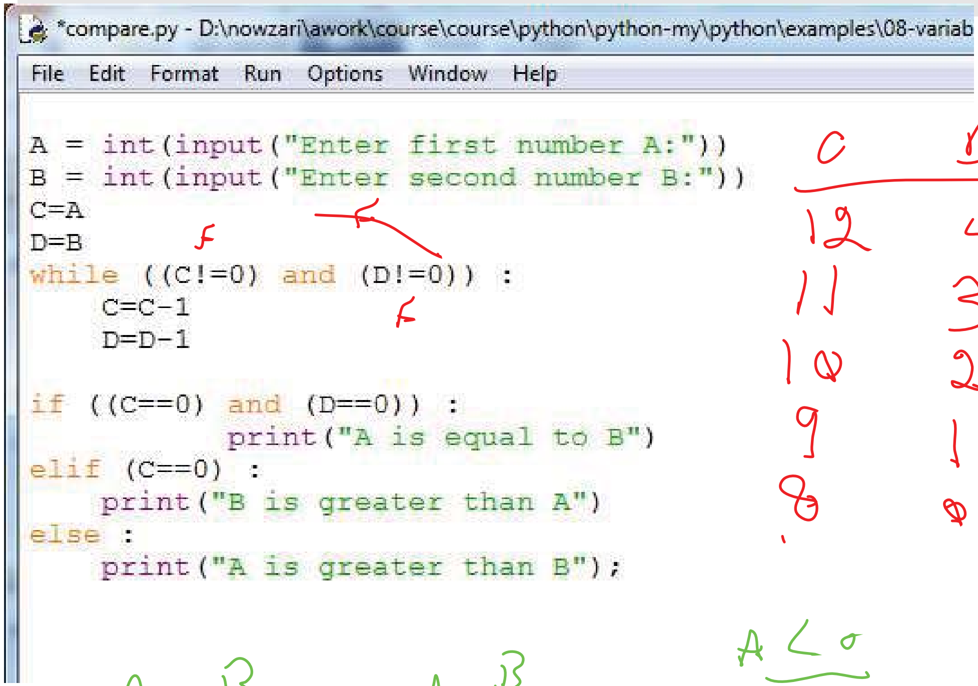
2 < A+B

7

3

10

Compare Example


*compare.py - D:\nowzari\awork\course\course\python\python-my\python\examples\08-variab
File Edit Format Run Options Window Help

```
A = int(input("Enter first number A:"))
B = int(input("Enter second number B:"))
C=A
D=B
while ((C!=0) and (D!=0)) :
    C=C-1
    D=D-1

if ((C==0) and (D==0)) :
            print("A is equal to B")
elif (C==0) :
    print("B is greater than A")
else :
    print("A is greater than B");
```

F F F

C	D
12	4
11	3
10	2
9	1
8	0

A	B
12	4
4	8
6	6
2	2

C	D
2	2
1	1
0	0

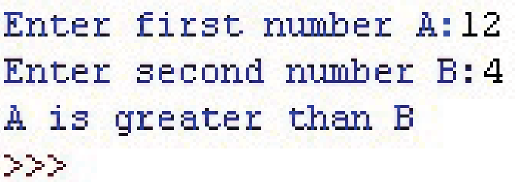
>
>5
<
<3

5 3
1 3

A	B
12	-4

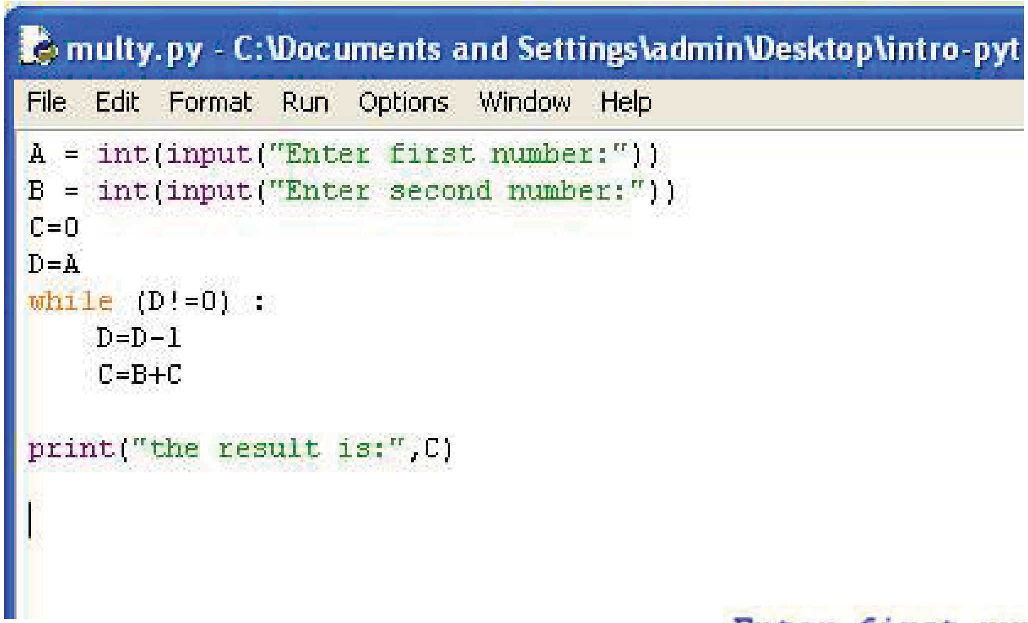
A	B
-8	-5

A < 0



```
Enter first number A:12
Enter second number B:4
A is greater than B
>>>
```

Multiply Example


multy.py - C:\Documents and Settings\admin\Desktop\intro-pyt
File Edit Format Run Options Window Help

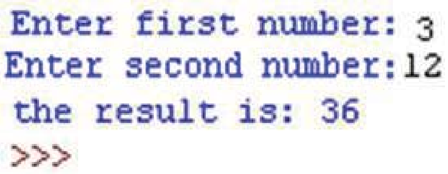
```
A = int(input("Enter first number:"))
B = int(input("Enter second number:"))
C=0
D=A
while (D!=0) :
    D=D-1
    C=B+C

print("the result is:",C)
```

A	B	C	D
3	12	0	3
		12	2
		24	1
		36	0

A * B

+

A	B
3	12

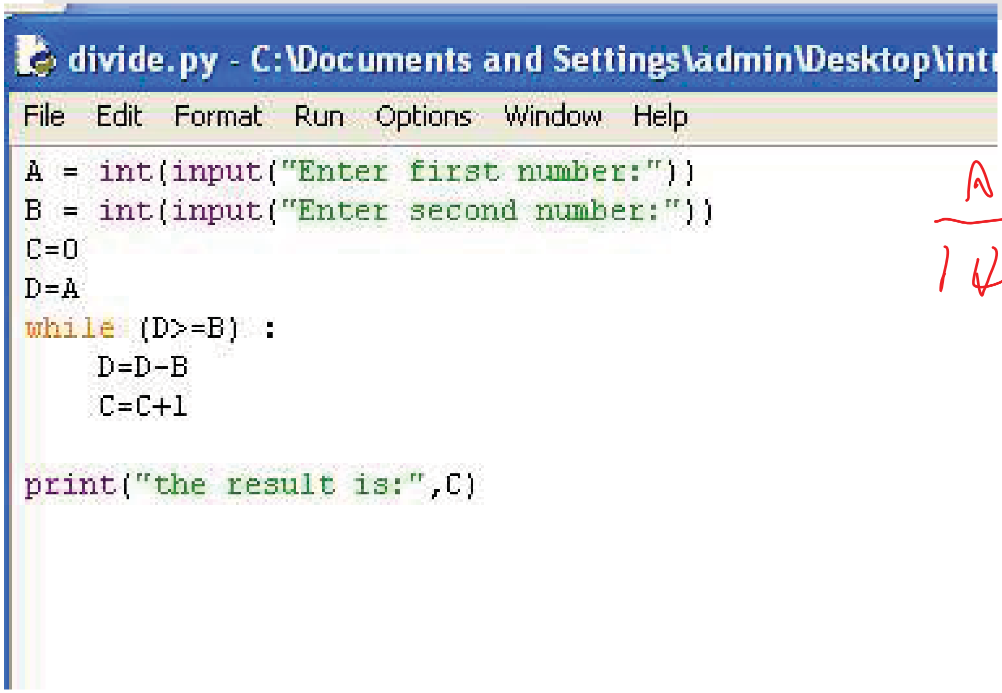


```
Enter first number: 3
Enter second number:12
the result is: 36
>>>
```

4
2
8

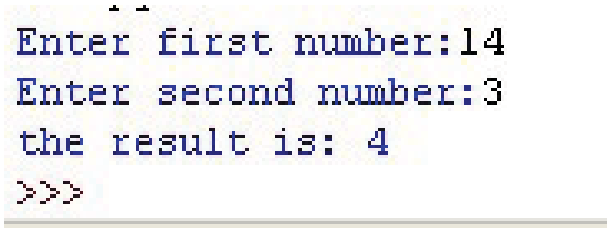
Divide Example

%



```
A = int(input("Enter first number:"))
B = int(input("Enter second number:"))
C=0
D=A
while (D>=B) :
     D=D-B
     C=C+1

print("the result is:",C)
```

A	B	C	D
14	3	0	14
		1	11
		2	8
		3	5
		4	2



```
Enter first number:14
Enter second number:3
the result is: 4
>>>
```

A//B

÷

14⌊3 4
=

12 3 | 1
9 | 2
6 | 3
3 | 4
0

Literals

- **Constants which is used in the text of the programs**

A = 3

B = "Test"

C = 3.14

Numeric Literals

- You can refer to numeric values using integers, floating point numbers, scientific notation, hexadecimal notation, octal, and complex numbers: Python integers can be an size. Integers larger than 2147483647 are called "long" integers because they can't be stored in 32 bits. (The distinction between integers and longs is slowly disappearing.)
- 123   # an integer
- 1.23   # a floating point number
- -1.23   # a negative floating point number
- 1.23E45 # scientific notation
- 0x7b   # hexadecimal notation (decimal 123)
- 0173   # octal notation (decimal 123)
- 12+3*j   # complex number 12 + 3i (Note that Python uses "j"!)
- 147483648 # a long integer

a = 123

A = B + 1.23

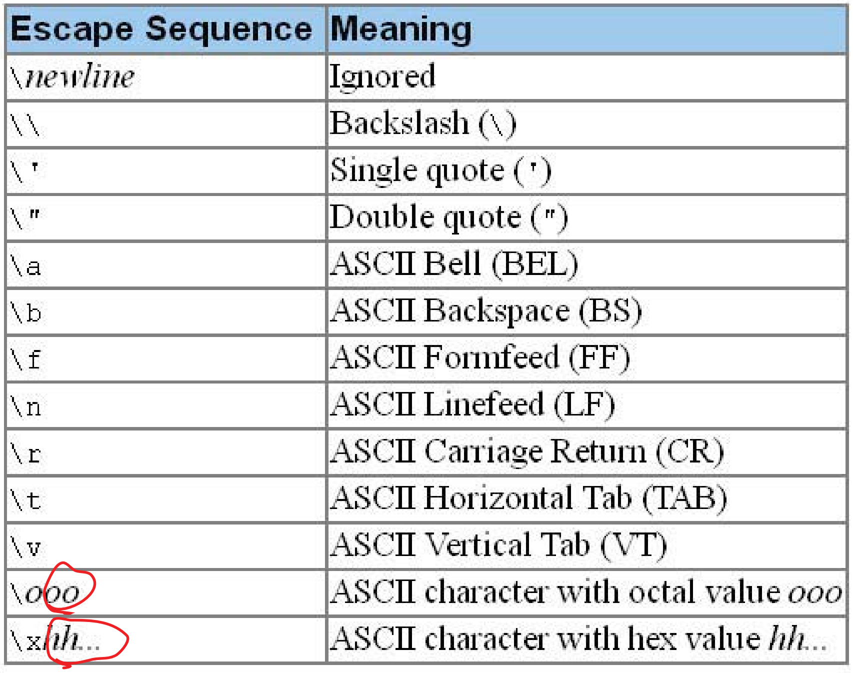
1.23×10^{45}

String Literals

Python has many different types of string literals. There are actually too many to get into, so I'll just show you a few examples

- # single quotes
  'Who said "to be or not to be"?'
- # double quotes
  "DNA goes from 5' to 3'."
- # escaped quotes
  "\"That's not fair!\" yelled my sister."
  # creates: "That's not fair!" yelled my sister
- # triple quoted strings, with single quotes
  '''This one string can go over several lines'''
- # "raw" strings, mostly used for regular expressions
  r"\"That's not fair!\" yelled my sister."
  # creates: \"That's not fair!\" yelled my sister

String Literals



Escape Sequence	Meaning
`\`*newline*	Ignored
`\\`	Backslash (`\`)
`\'`	Single quote (`'`)
`\"`	Double quote (`"`)
`\a`	ASCII Bell (BEL)
`\b`	ASCII Backspace (BS)
`\f`	ASCII Formfeed (FF)
`\n`	ASCII Linefeed (LF)
`\r`	ASCII Carriage Return (CR)
`\t`	ASCII Horizontal Tab (TAB)
`\v`	ASCII Vertical Tab (VT)
`\`*ooo*	ASCII character with octal value *ooo*
`\x`*hh...*	ASCII character with hex value *hh...*

3 8

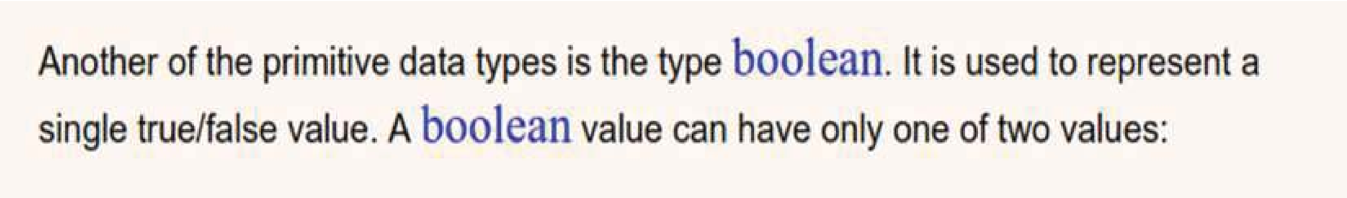
3

8

\073

\x7A

Boolean Literals



Another of the primitive data types is the type boolean. It is used to represent a single true/false value. A boolean value can have only one of two values:

True
False

A = True

A
1
0

A = False

print(A)

End