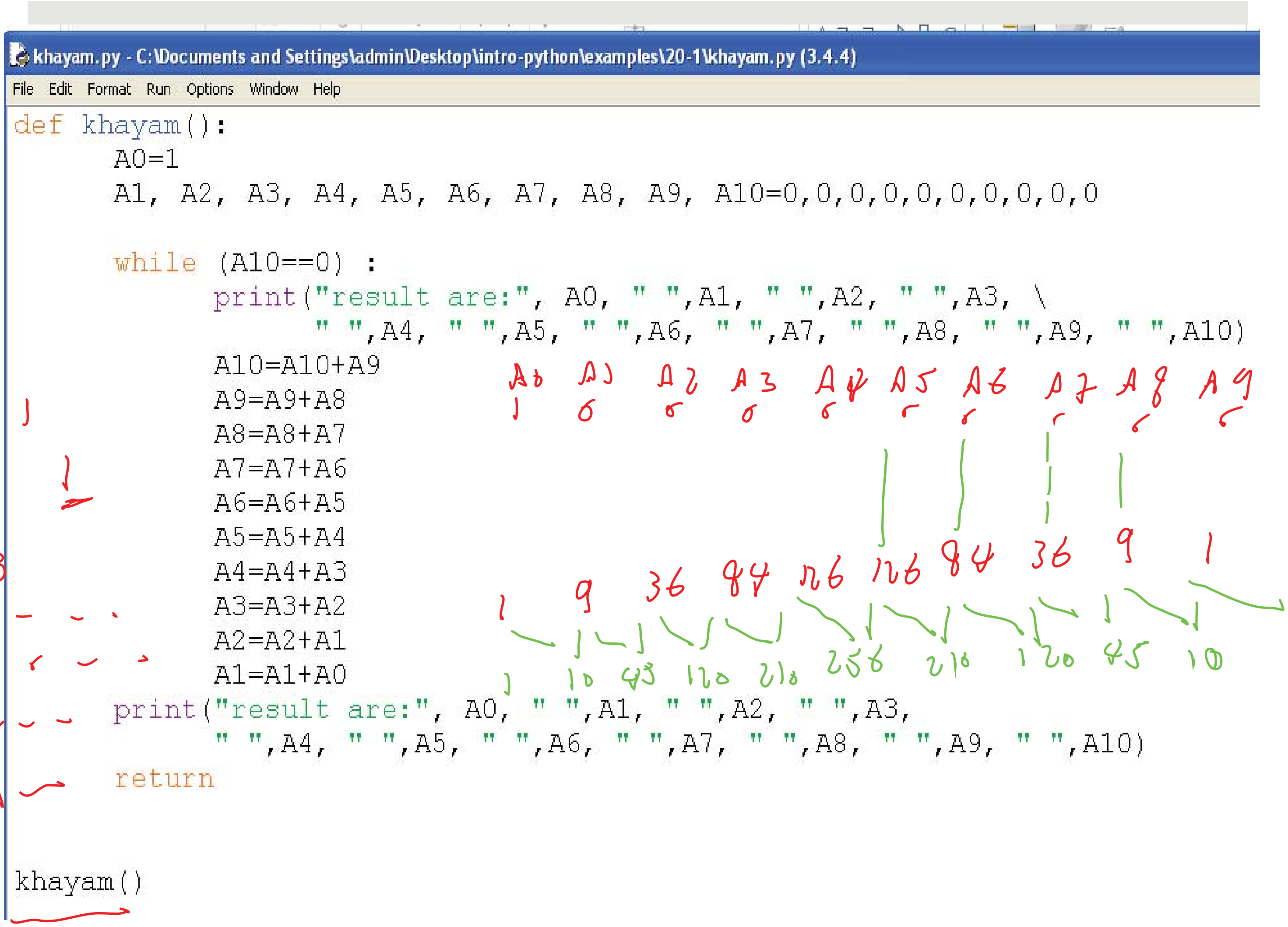


khayam.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\20-1\khayam.py (3.4.4)
File Edit Format Run Options Window Help

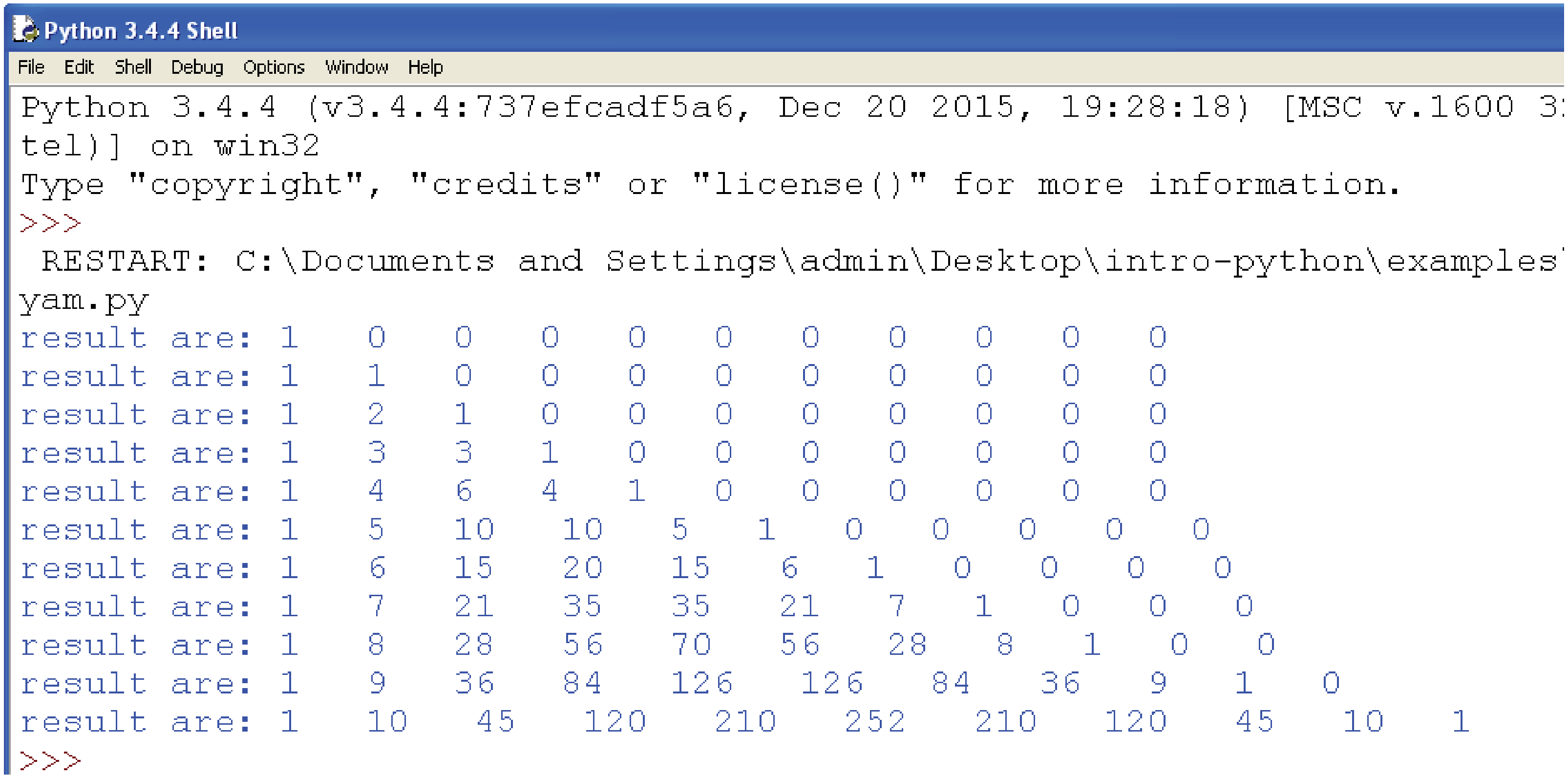
```python
def khayam():
        A0=1
        A1, A2, A3, A4, A5, A6, A7, A8, A9, A10=0,0,0,0,0,0,0,0,0,0

        while (A10==0) :
                print("result are:", A0, " ",A1, " ",A2, " ",A3, \
                        " ",A4, " ",A5, " ",A6, " ",A7, " ",A8, " ",A9, " ",A10)
                A10=A10+A9
                A9=A9+A8
                A8=A8+A7
                A7=A7+A6
                A6=A6+A5
                A5=A5+A4
                A4=A4+A3
                A3=A3+A2
                A2=A2+A1
                A1=A1+A0
        print("result are:", A0, " ",A1, " ",A2, " ",A3,
                " ",A4, " ",A5, " ",A6, " ",A7, " ",A8, " ",A9, " ",A10)
        return

khayam()
```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1

A0 A1 A2 A3
1 0 0 0
1 1 0 0
1 2 1
1 3 3 1

A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10
1 0 0 0 0 0 0 0 0 0 0
1 9 36 84 126 126 84 36 9 1 0
1 10 45 120 210 252 210 120 45 10 1

Khayyam triangle



```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:18) [MSC v.1600 3
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
 RESTART: C:\Documents and Settings\admin\Desktop\intro-python\examples
yam.py
result are: 1   0   0   0   0   0   0   0   0   0   0
result are: 1   1   0   0   0   0   0   0   0   0   0
result are: 1   2   1   0   0   0   0   0   0   0   0
result are: 1   3   3   1   0   0   0   0   0   0   0
result are: 1   4   6   4   1   0   0   0   0   0   0
result are: 1   5   10   10   5   1   0   0   0   0   0
result are: 1   6   15   20   15   6   1   0   0   0   0
result are: 1   7   21   35   35   21   7   1   0   0   0
result are: 1   8   28   56   70   56   28   8   1   0   0
result are: 1   9   36   84   126   126   84   36   9   1   0
result are: 1   10   45   120   210   252   210   120   45   10   1
>>> 
```

Fibonacci numbers

0; 1 2 3 4 5 6 7 8 9 10

0; 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

f(n)=f(n-1) + f(n-2)   n>=2

f(1) = 1

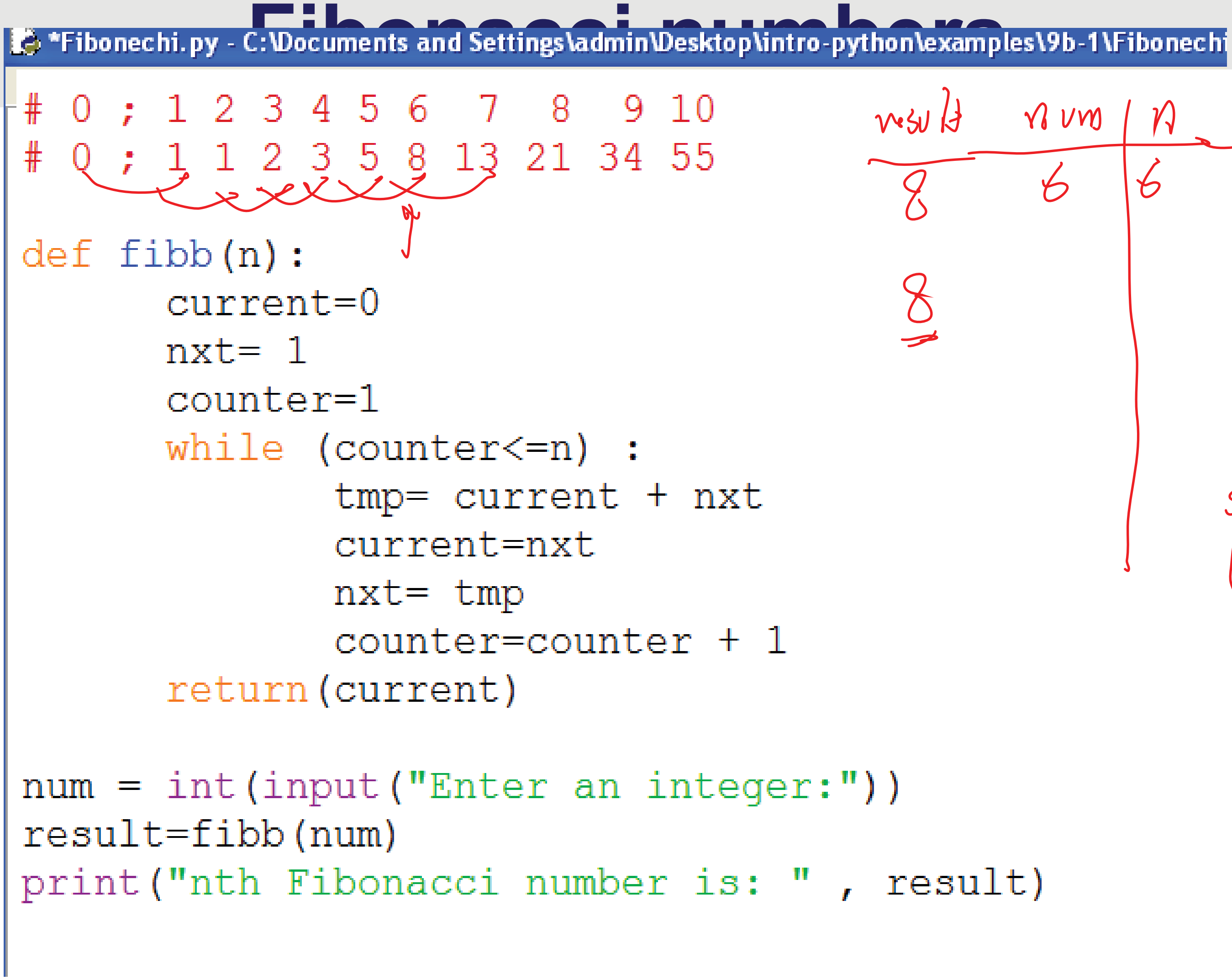
f(n) = 0 n<=0

$f(10) = f(9) + f(8)$

$= 34 + 21$

$= 55$

Fibonacci numbers

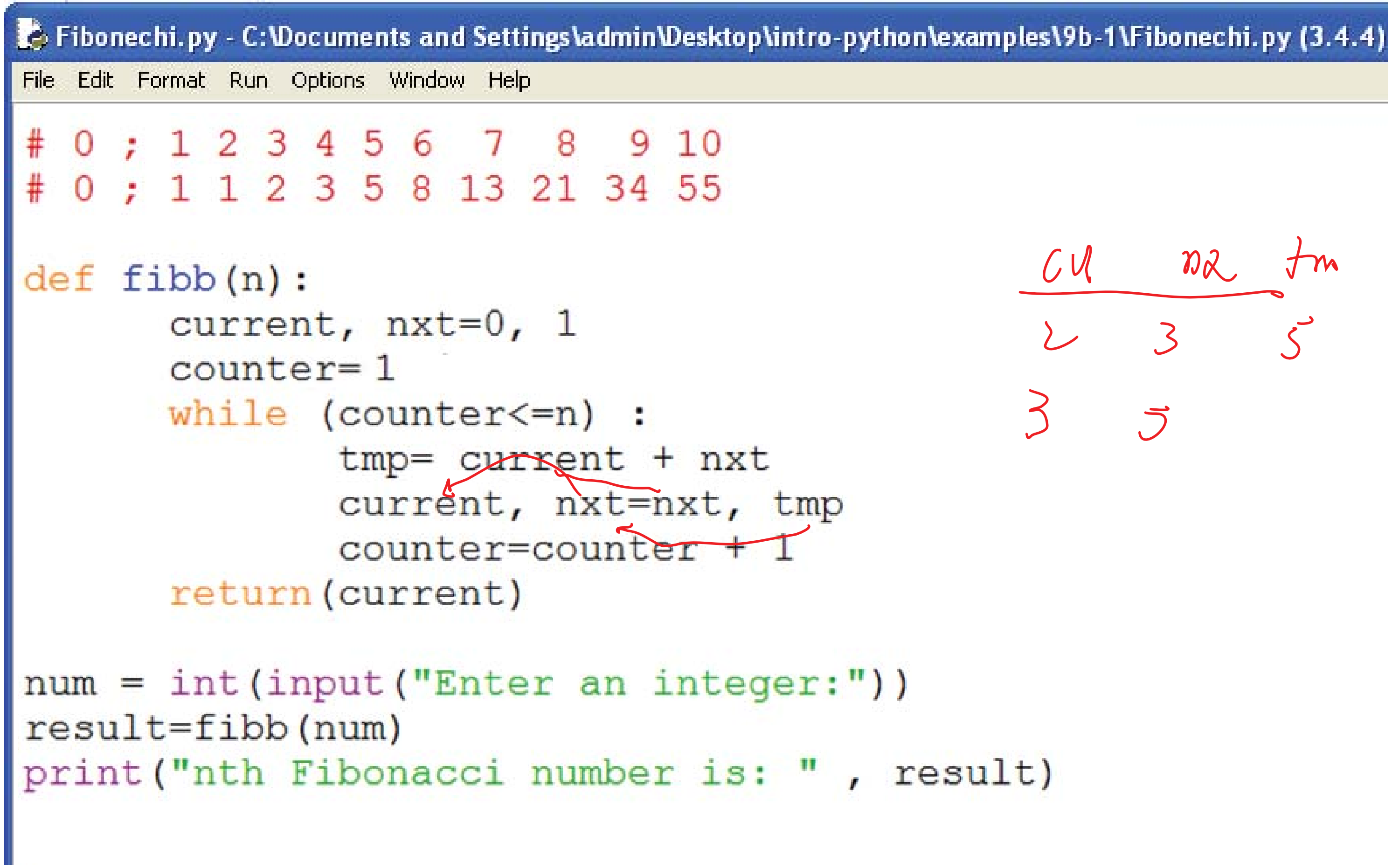


*Fibonechi.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\9b-1\Fibonechi

```
# 0 ; 1 2 3 4 5 6  7   8   9 10
# 0 ; 1 1 2 3 5 8 13 21 34 55

def fibb(n):
      current=0
      nxt= 1
      counter=1
      while (counter<=n) :
             tmp= current + nxt
             current=nxt
             nxt= tmp
             counter=counter + 1
      return(current)

num = int(input("Enter an integer:"))
result=fibb(num)
print("nth Fibonacci number is: " , result)
```

result	num	n	cu	nx	co	tm
8	6	6	0	1	1	1
			1	1	2	2
8			1	2	3	3
			2	3	4	5
			3	5	5	8
			5	8	6	13
			8	13	7	

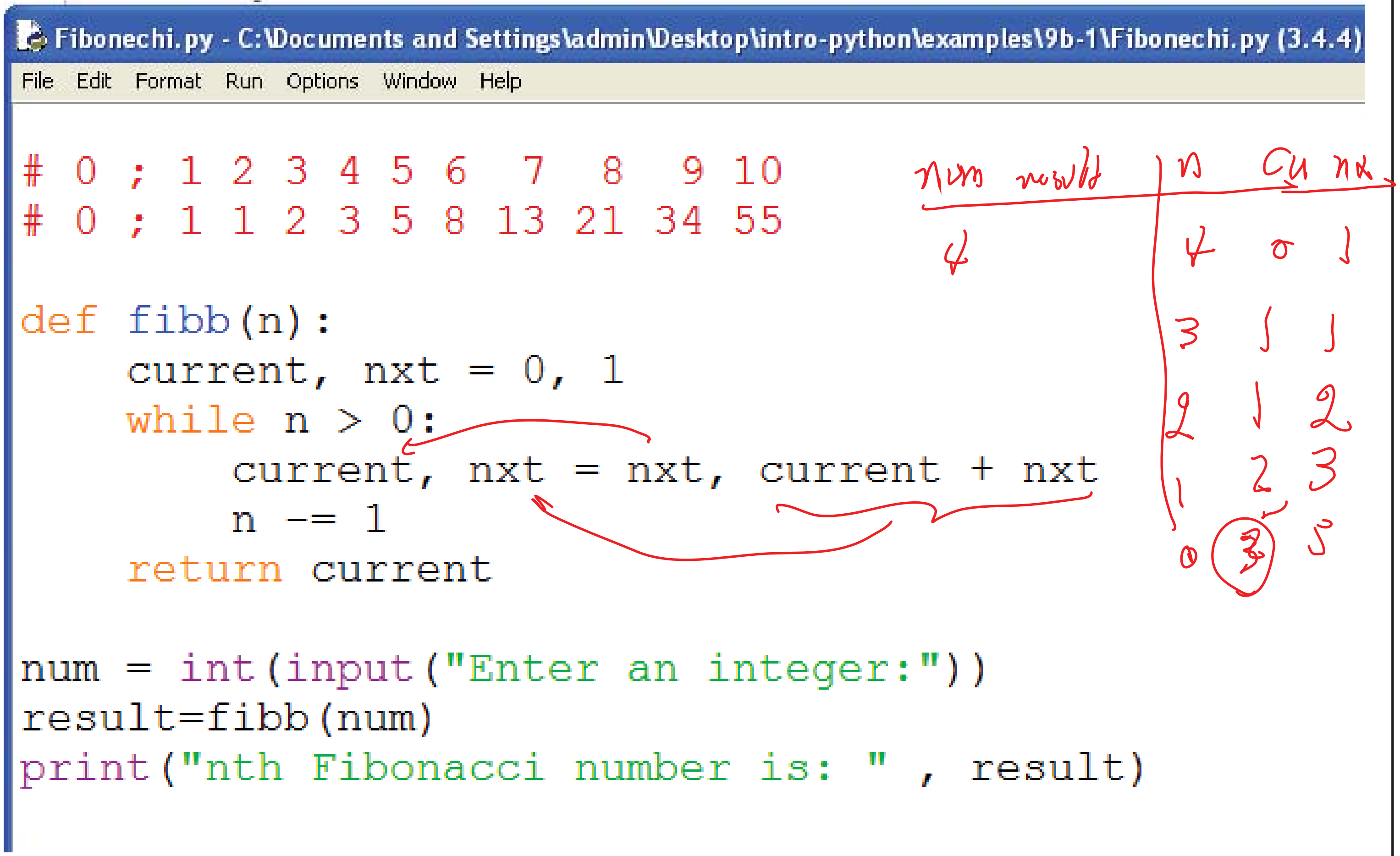
Fibonacci numbers


Fibonechi.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\9b-1\Fibonechi.py (3.4.4)
File Edit Format Run Options Window Help

```
# 0 ; 1 2 3 4 5 6  7  8  9 10
# 0 ; 1 1 2 3 5 8 13 21 34 55

def fibb(n):
	current, nxt=0, 1
	counter= 1
	while (counter<=n) :
		tmp= current + nxt
		current, nxt=nxt, tmp
		counter=counter + 1
	return(current)

num = int(input("Enter an integer:"))
result=fibb(num)
print("nth Fibonacci number is: " , result)
```

cu	nx	tm
2	3	5
3	5	

Fibonacci numbers


Fibonechi.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\9b-1\Fibonechi.py (3.4.4)
File Edit Format Run Options Window Help

```
# 0 ; 1 2 3 4 5 6  7  8  9 10
# 0 ; 1 1 2 3 5 8 13 21 34 55

def fibb(n):
    current, nxt = 0, 1
    while n > 0:
        current, nxt = nxt, current + nxt
        n -= 1
    return current

num = int(input("Enter an integer:"))
result=fibb(num)
print("nth Fibonacci number is: " , result)
```

num	result	n	cu	nx
4		4	0	1
		3	1	1
		2	1	2
		1	2	3
		0	3	5

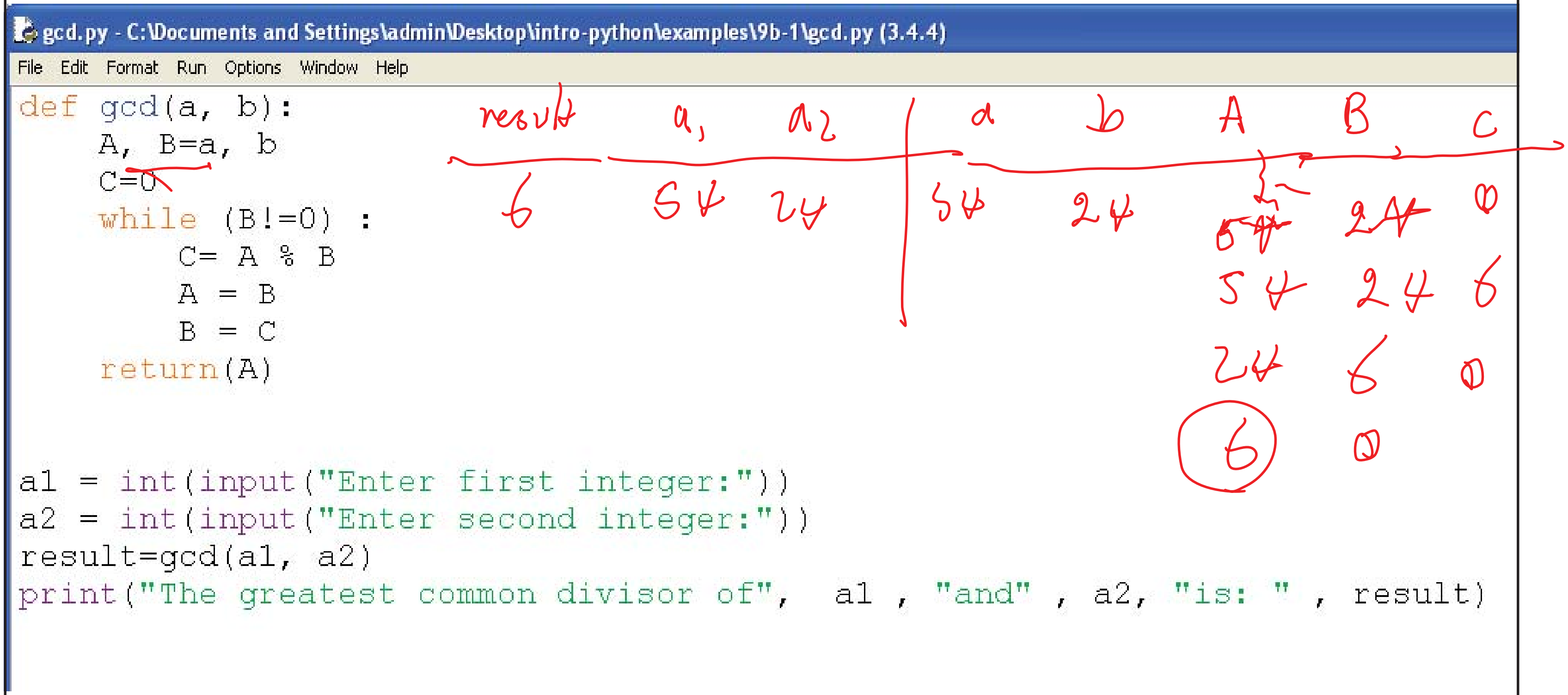
greatest common divisor

- Gcd(54, 24) = 6

a b

a	b	c
54	24	6
24	6	0
6	0	
=	=	

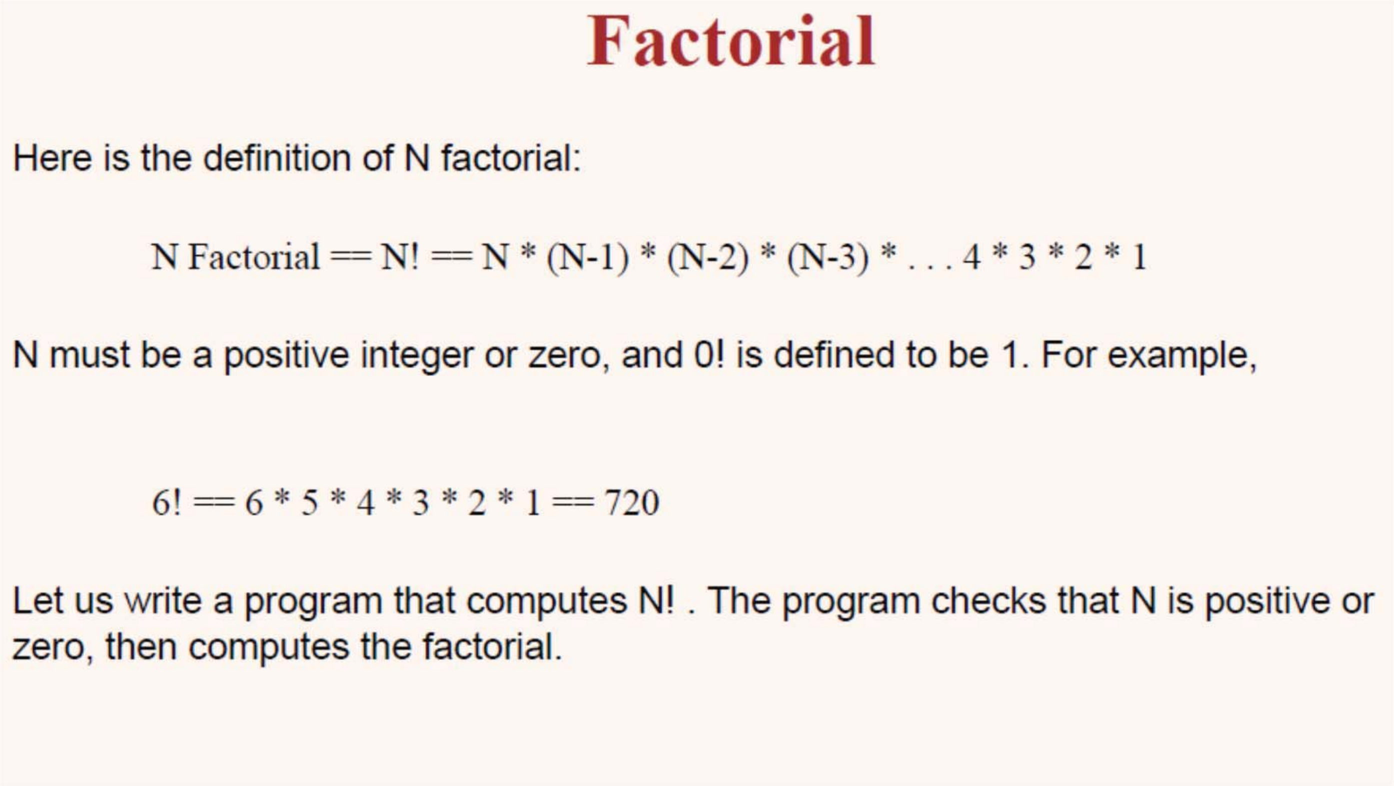
greatest common divisor


gcd.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\9b-1\gcd.py (3.4.4)
File Edit Format Run Options Window Help

```
def gcd(a, b):
    A, B=a, b
    C=0
    while (B!=0) :
        C= A % B
        A = B
        B = C
    return(A)


a1 = int(input("Enter first integer:"))
a2 = int(input("Enter second integer:"))
result=gcd(a1, a2)
print("The greatest common divisor of",  a1 , "and" , a2, "is: " , result)
```

result	a1	a2	a	b	A	B	C
6	54	24	54	24	~~54~~	~~24~~	0
					54	24	6
					24	6	0
					6	0	

Factorial



Factorial

Here is the definition of N factorial:

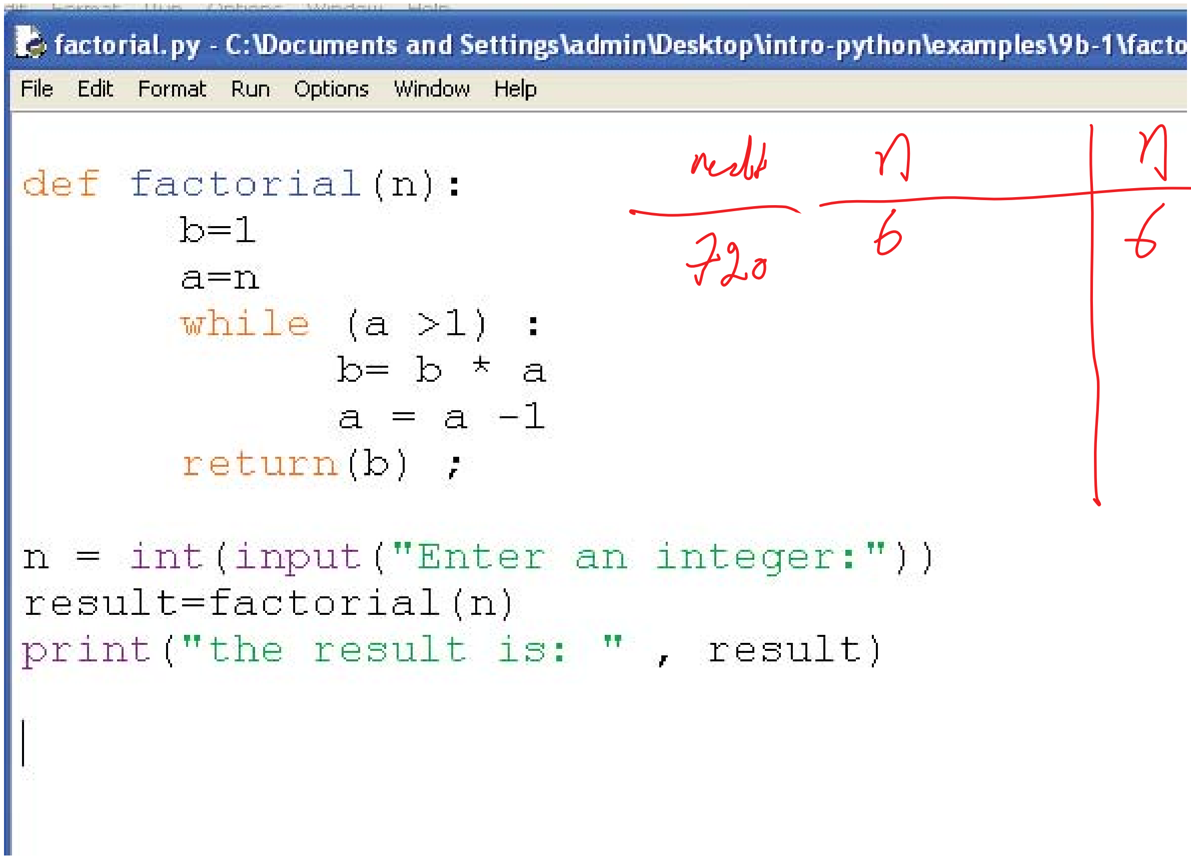
N Factorial == N! == N * (N-1) * (N-2) * (N-3) * . . . 4 * 3 * 2 * 1

N must be a positive integer or zero, and 0! is defined to be 1. For example,

6! == 6 * 5 * 4 * 3 * 2 * 1 == 720

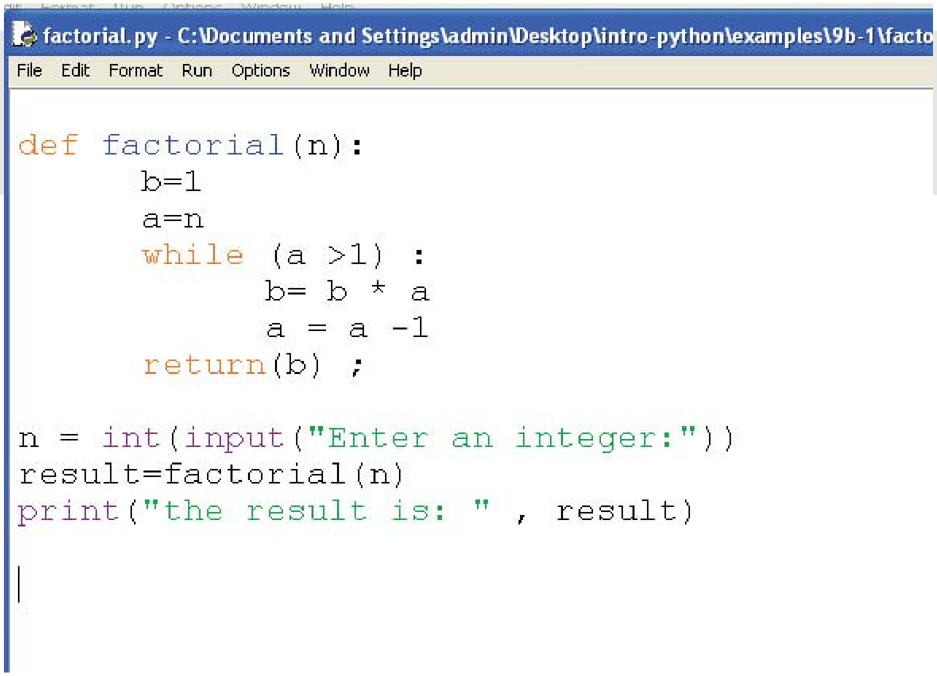
Let us write a program that computes N! . The program checks that N is positive or zero, then computes the factorial.

Factorial


factorial.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\9b-1\facto
File
Edit
Format
Run
Options
Window
Help

```
def factorial(n):
      b=1
      a=n
      while (a >1) :
            b= b * a
            a = a -1
      return(b) ;

n = int(input("Enter an integer:"))
result=factorial(n)
print("the result is: " , result)
```

result	n	n	b	a	
720	6	6	1	6	
			6	5	
			30	4	6x5
			120	3	6x5x4
			360	2	6x5x4x3
			720	1	6x5x4x3x2

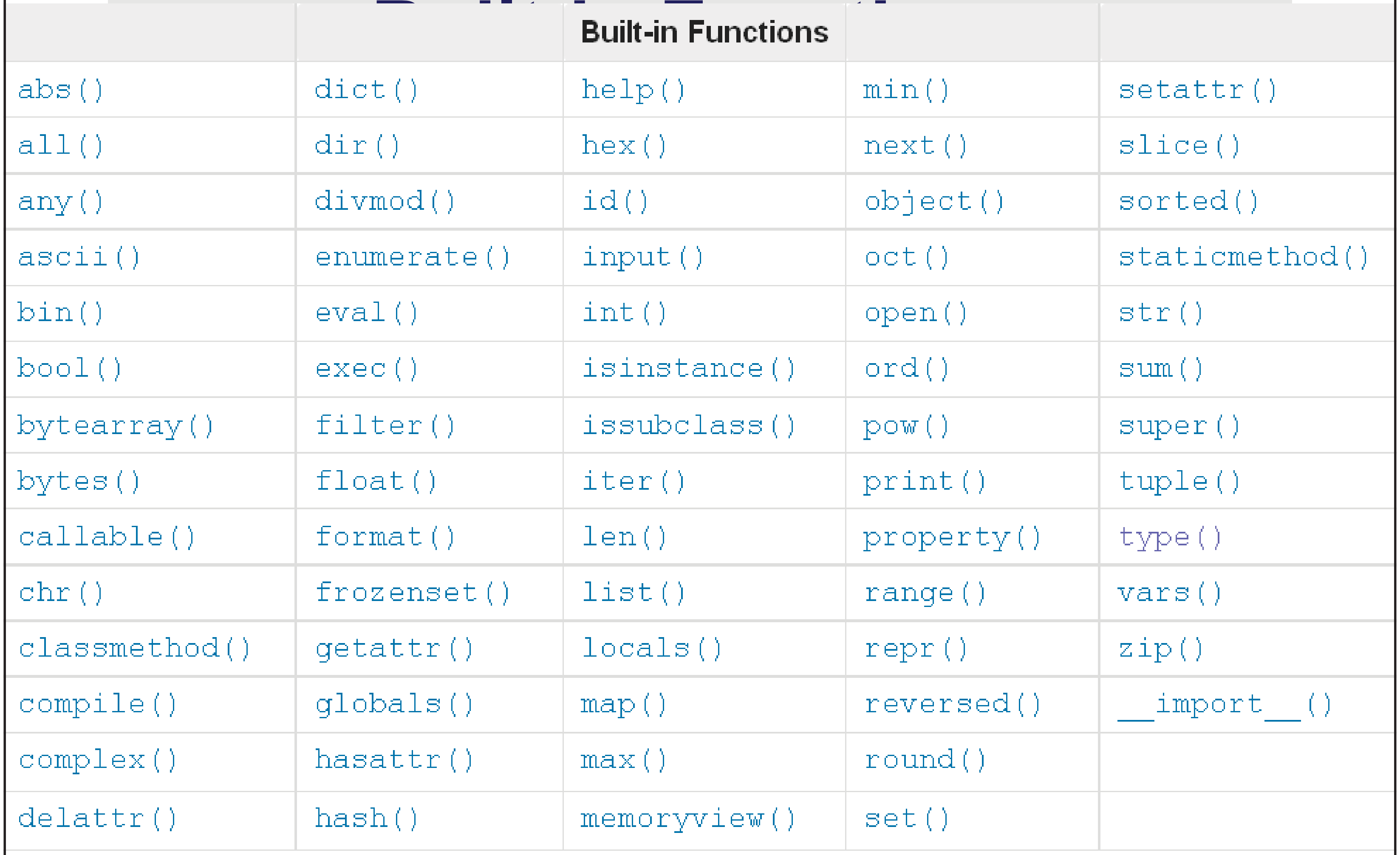

factorial.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\9b-1\facto
File Edit Format Run Options Window Help

```
def factorial(n):
        b=1
        a=n
        while (a >1) :
                b= b * a
                a = a -1
        return(b) ;

n = int(input("Enter an integer:"))
result=factorial(n)
print("the result is: " , result)
```

```
Enter an integer:200
the result is:  7886578673647905035523632139321850622951359776871732632947
42533244359449963403342920304284011984623904177212138919638830257642790242
63710506192662495282993111346285727076331723739698894392244562145166424025
40332918641312274282948532775242424075739032403212574055795686602260319041
70324062351700858796178922222789623703897374720000000000000000000000000000
000000000000000000000
```

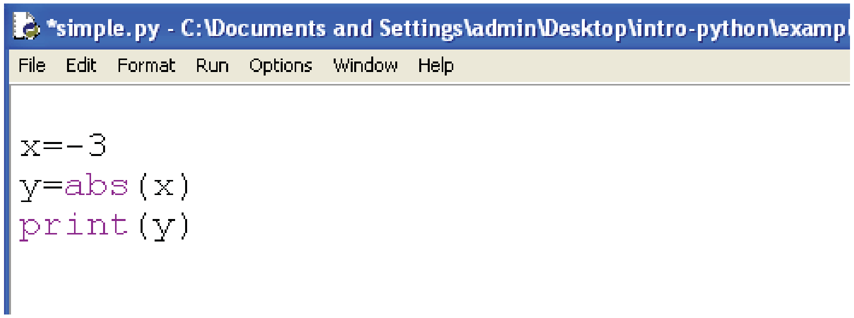
Built-in Functions

- The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.



		Built-in Functions		
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

Built-in Functions


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

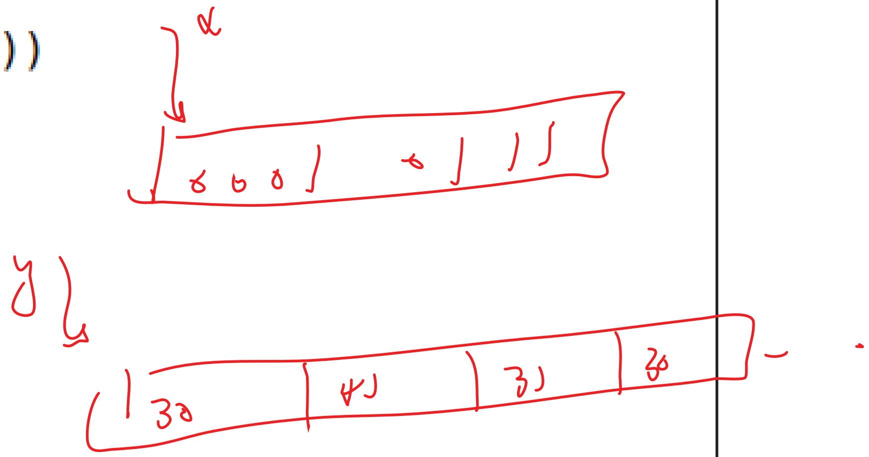
```
x=-3
y=abs(x)
print(y)
```

3

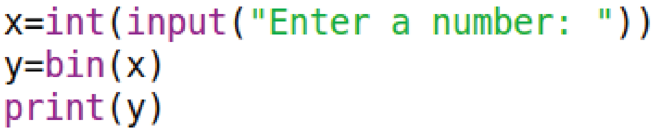
Built-in Functions

```
x=int(input("Enter a number: "))
y=bin(x)
print(y)
```

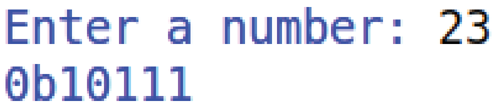
```
Enter a number: 23
0b10111
```



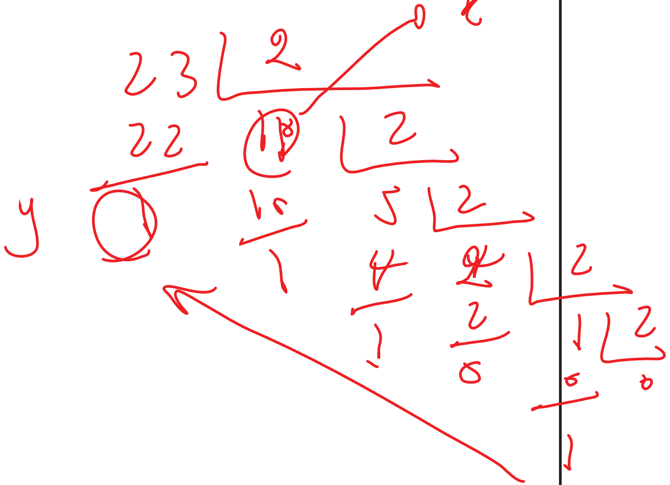
Built-in Functions

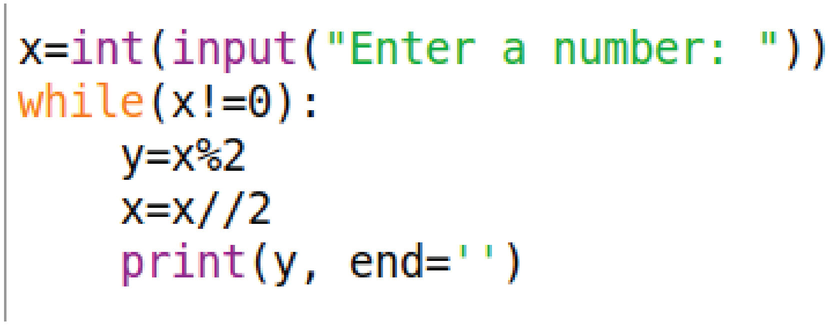


```
x=int(input("Enter a number: "))
y=bin(x)
print(y)
```

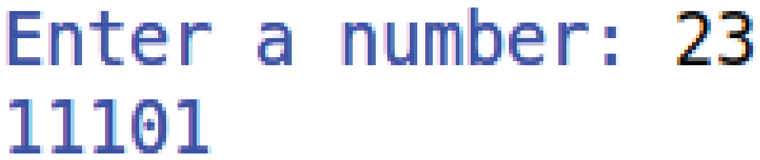


```
Enter a number: 23
0b10111
```



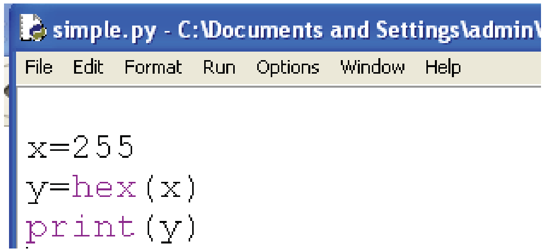


```
x=int(input("Enter a number: "))
while(x!=0):
    y=x%2
    x=x//2
    print(y, end='')
```



```
Enter a number: 23
11101
```

Built-in Functions

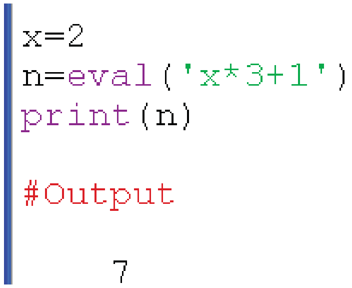

simple.py - C:\Documents and Settings\admin\
File Edit Format Run Options Window Help

```
x=255
y=hex(x)
print(y)
```

1111 1111

- **0xff**

Built-in Functions

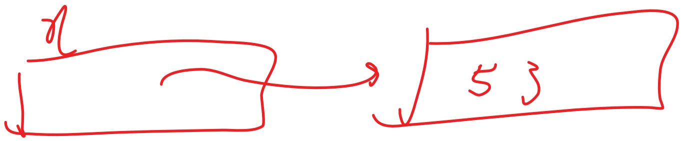


```
x=2
n=eval('x*3+1')
print(n)

#Output

    7
```

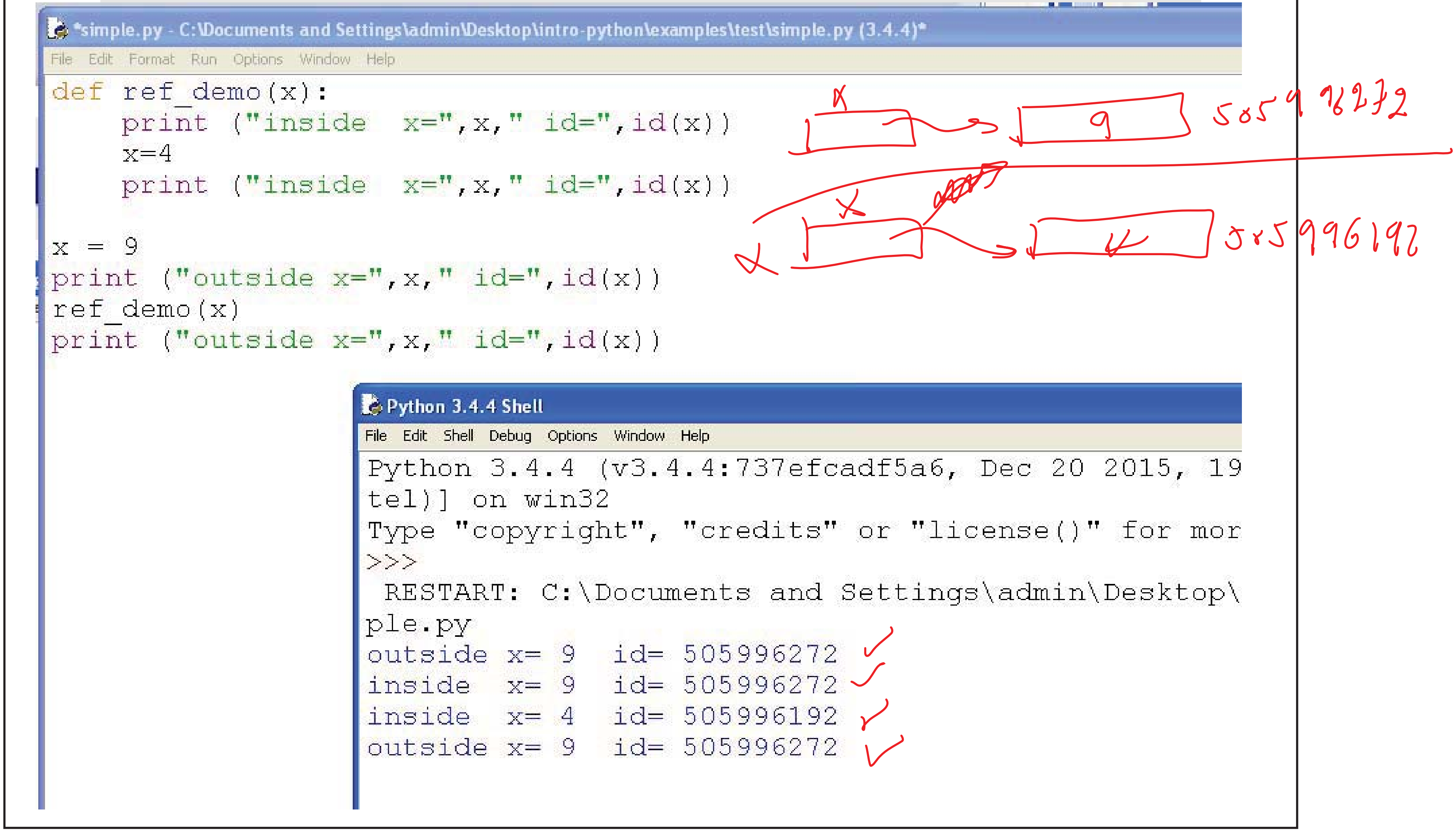
id Functions

- id(*object*) Return the “identity” of an object. This is an integer which is guaranteed to be unique and constant for this object during its lifetime. Two objects with non-overlapping lifetimes may have the same id() value.

x = 53



Parameter passing



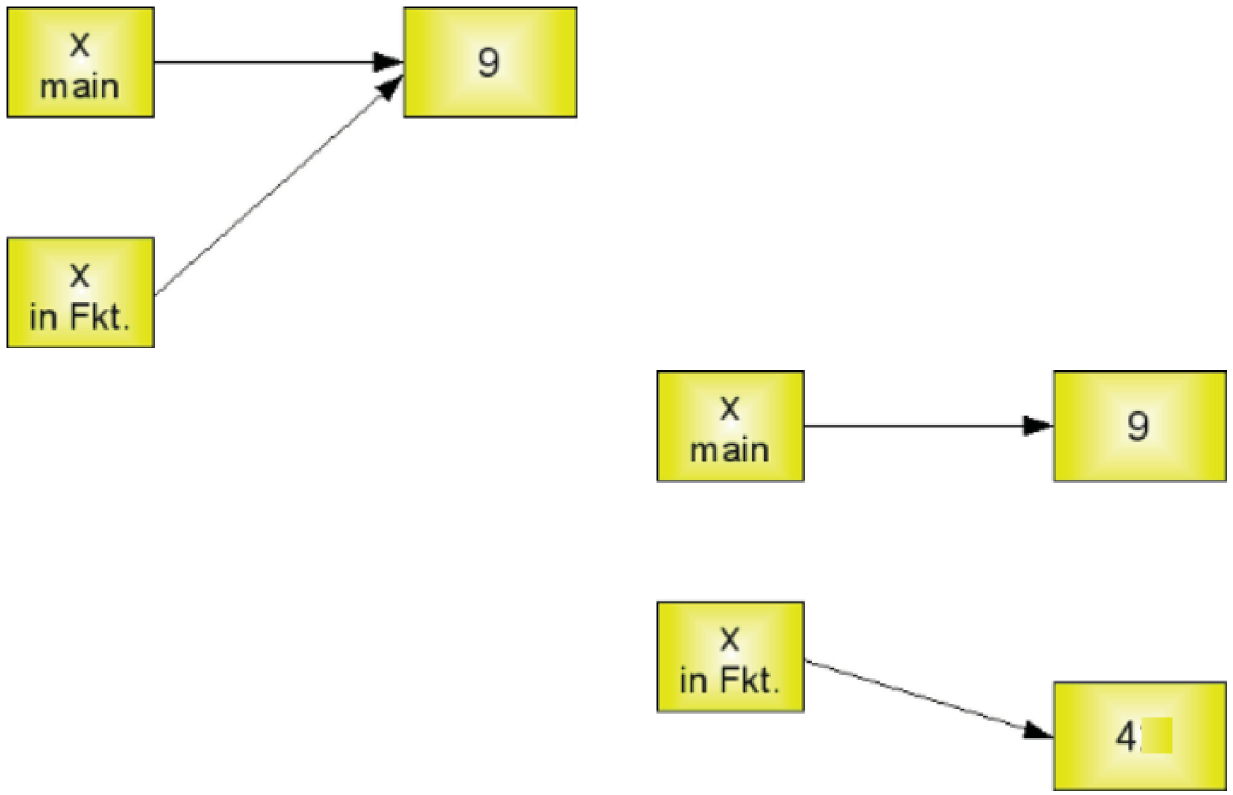
simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)
File Edit Format Run Options Window Help

```
def ref_demo(x):
    print ("inside  x=",x," id=",id(x))
    x=4
    print ("inside  x=",x," id=",id(x))

x = 9
print ("outside x=",x," id=",id(x))
ref_demo(x)
print ("outside x=",x," id=",id(x))
```

Python 3.4.4 Shell
File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19
tel)] on win32
Type "copyright", "credits" or "license()" for mor
>>>
 RESTART: C:\Documents and Settings\admin\Desktop\
ple.py
outside x= 9  id= 505996272 ✓
inside  x= 9  id= 505996272 ✓
inside  x= 4  id= 505996192 ✓
outside x= 9  id= 505996272 ✓
```

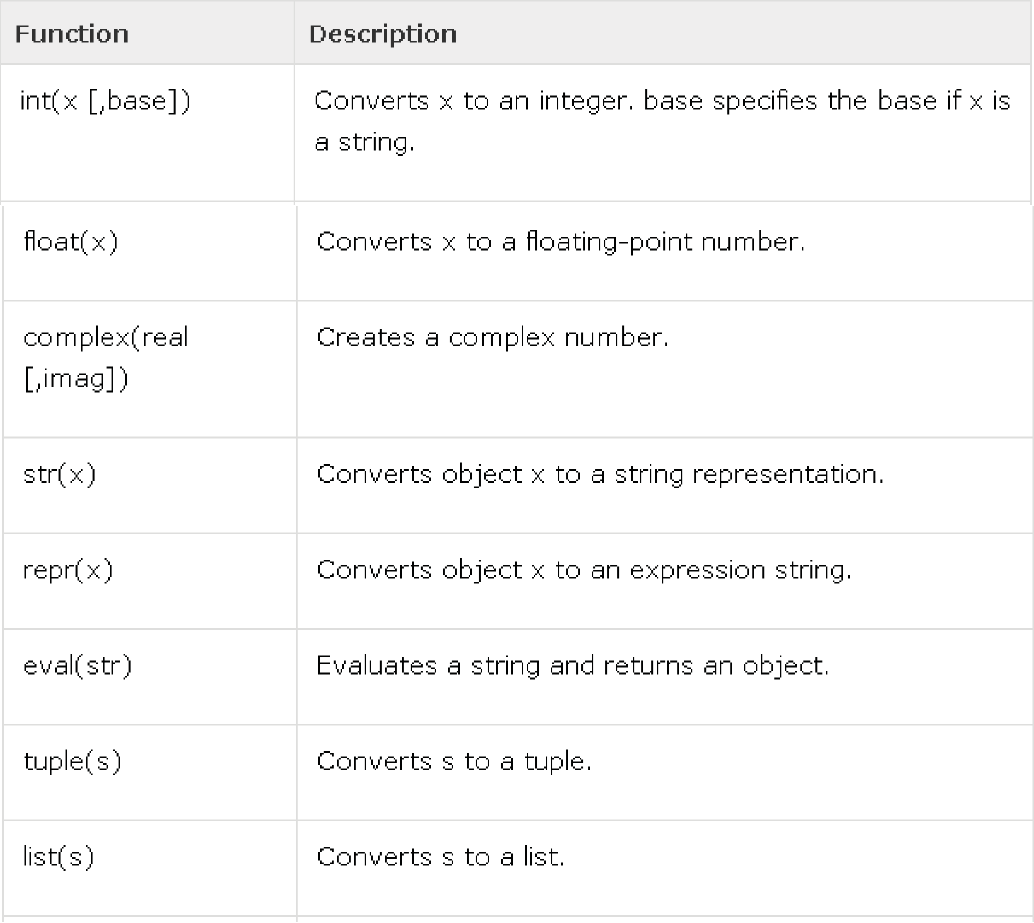
Parameter passing



Built-in Type conversion Functions

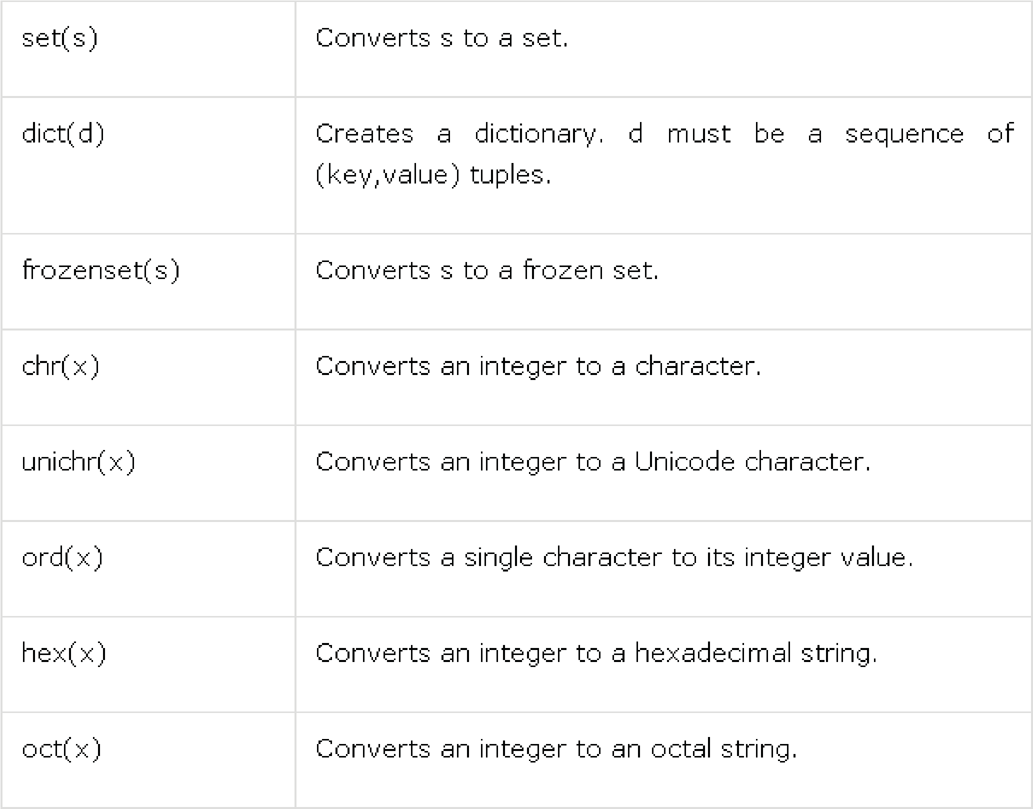
- Sometimes, you may need to perform conversions between the built-in types. To convert between types, you simply use the type name as a function.
- There are several built-in functions to perform conversion from one data type to another. These functions return a new object representing the converted value.

Built-in Type conversion Functions



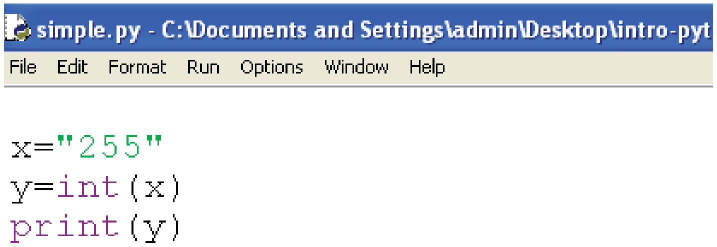
Function	Description
int(x [,base])	Converts x to an integer. base specifies the base if x is a string.
float(x)	Converts x to a floating-point number.
complex(real [,imag])	Creates a complex number.
str(x)	Converts object x to a string representation.
repr(x)	Converts object x to an expression string.
eval(str)	Evaluates a string and returns an object.
tuple(s)	Converts s to a tuple.
list(s)	Converts s to a list.

Built-in Type conversion Functions

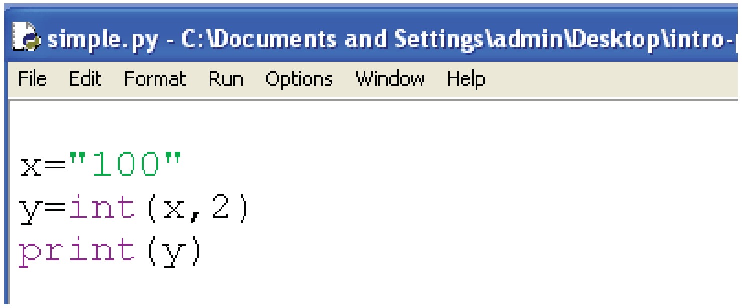


set(s)	Converts s to a set.
dict(d)	Creates a dictionary. d must be a sequence of (key,value) tuples.
frozenset(s)	Converts s to a frozen set.
chr(x)	Converts an integer to a character.
unichr(x)	Converts an integer to a Unicode character.
ord(x)	Converts a single character to its integer value.
hex(x)	Converts an integer to a hexadecimal string.
oct(x)	Converts an integer to an octal string.

Built-in Type conversion Functions


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

```
x="255"
y=int(x)
print(y)
```


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

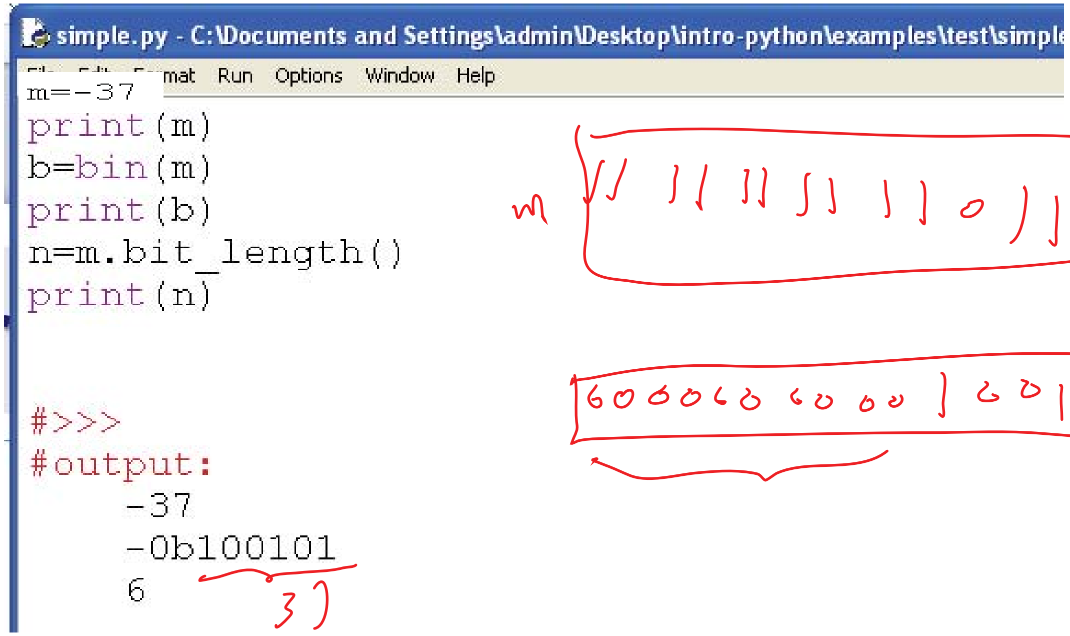
```
x="100"
y=int(x,2)
print(y)
```

4

Functions for types

- **Each types have some functions and attributes**
  - int
    - int.bit_length()
    - int.to_bytes(*length*, *byteorder*, *, *signed=False*)
  - Float
    - float.is_integer()
  - string
    - str.capitalize()

Functions for types


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

```
m=-37
print(m)
b=bin(m)
print(b)
n=m.bit_length()
print(n)


#>>>
#output:
    -37
    -0b100101
    6
```

37

m 1111111111011011 -37

0000000000100101 37

Python Modules

- A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.
- Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.
- Python has very predefined modules such as
  - math
  - cmath
  - parser

import statement

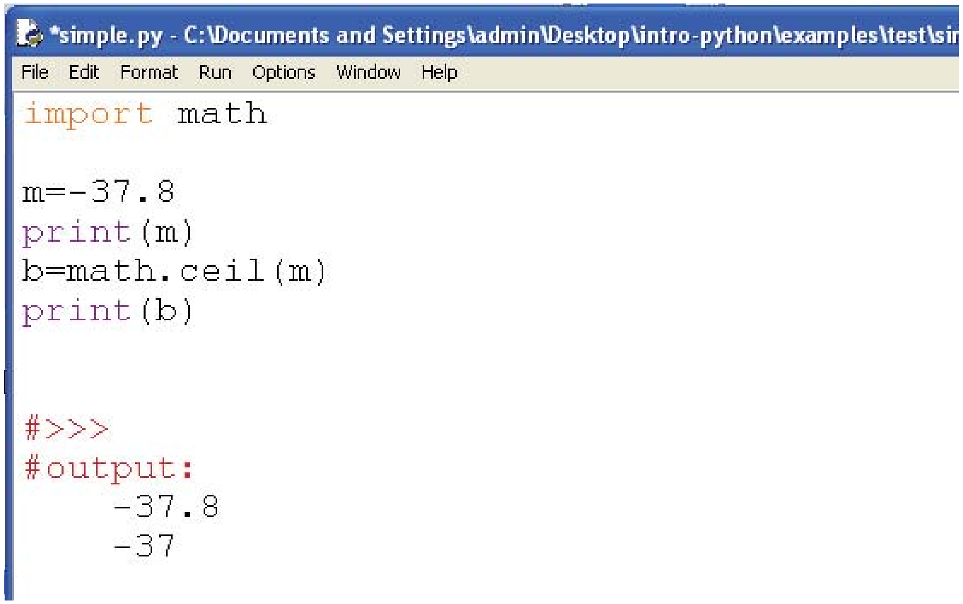
- You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax:

```
import module1[, module2[,... moduleN]
```

- When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. For example, to import the module hello.py, you need to put the following command at the top of the script

import statement

- **math.ceil(x): Return the ceiling of *x*, the smallest integer greater than or equal to *x*.**


*simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\si
File Edit Format Run Options Window Help

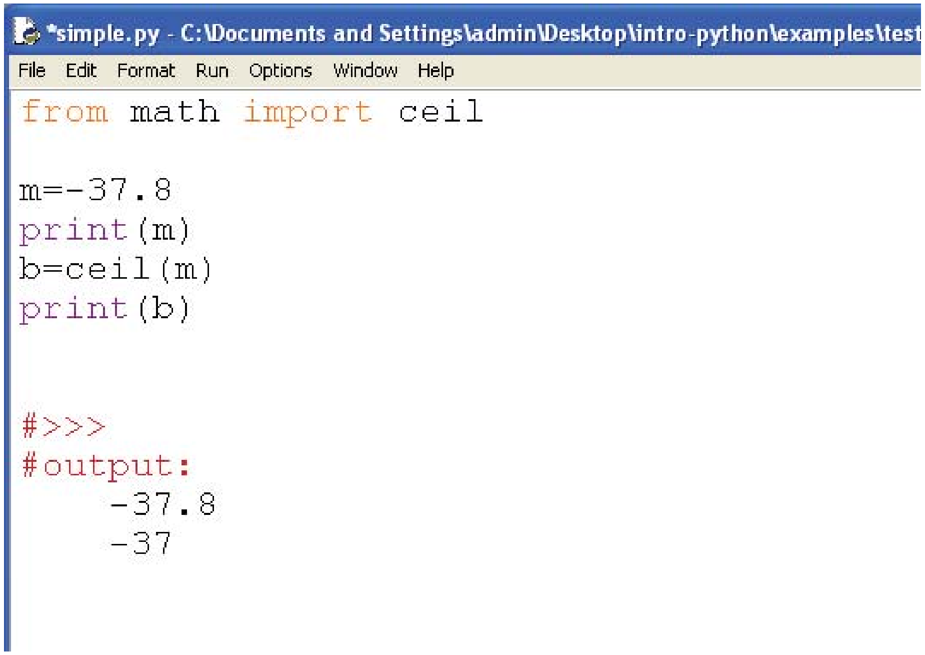
```
import math

m=-37.8
print(m)
b=math.ceil(m)
print(b)


#>>>
#output:
    -37.8
    -37
```

The *from...import* Statement

- Python's *from* statement lets you import specific attributes from a module into the current namespace. The *from...import* has the following syntax :

```
from modname import name1[, name2[, ... nameN]]
```

The *from...import* Statement


*simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test
File Edit Format Run Options Window Help

```
from math import ceil

m=-37.8
print(m)
b=ceil(m)
print(b)


#>>>
#output:
    -37.8
    -37
```

The *from...import* * Statement

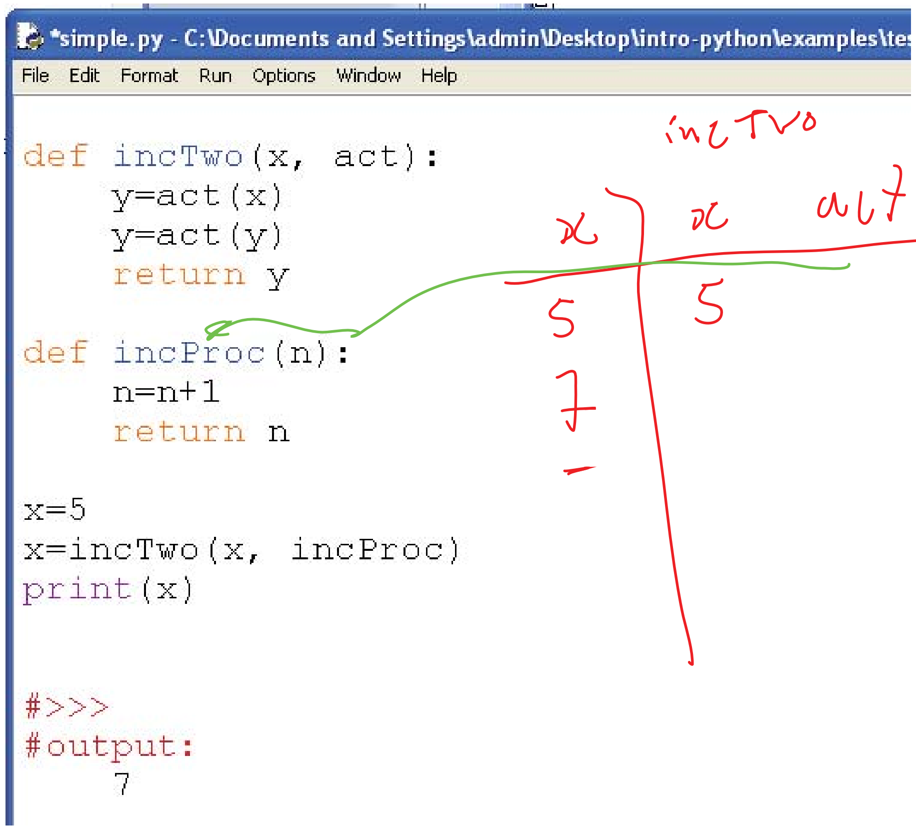
- **It is also possible to import all names from a module into the current namespace by using the following import statement:**

```
from modname import *
```

Function as a parameter

- It is also possible to pass a function as a parameter to another function

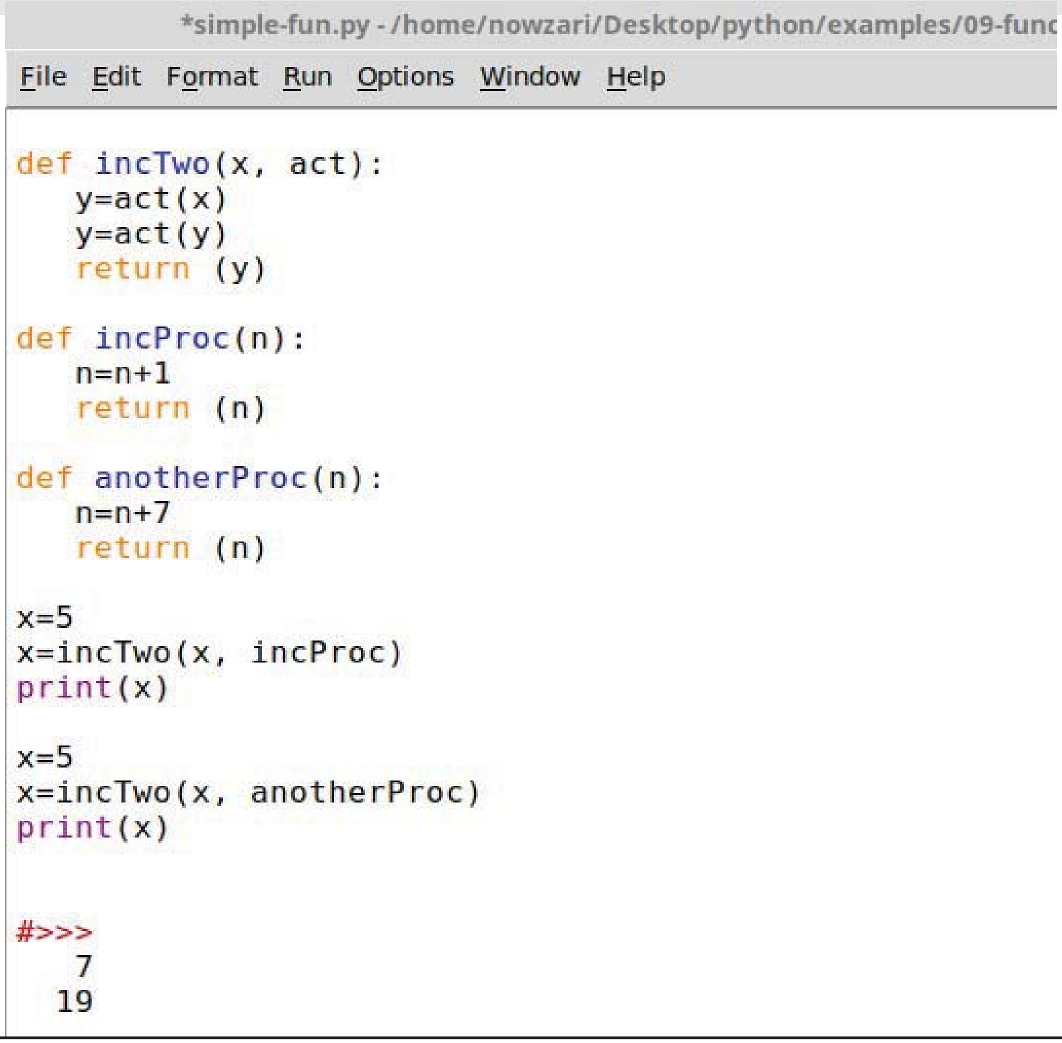
Function as a parameter


*simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\te
File
Edit
Format
Run
Options
Window
Help

```
def incTwo(x, act):
    y=act(x)
    y=act(y)
    return y

def incProc(n):
    n=n+1
    return n

x=5
x=incTwo(x, incProc)
print(x)


#>>>
#output:
    7
```

	incTwo			inProc
x	x	act	y	n
5	5		6	5
7			7	6
–				6
				7

Function as a parameter


*simple-fun.py - /home/nowzari/Desktop/python/examples/09-func
File
Edit
Format
Run
Options
Window
Help

```
def incTwo(x, act):
   y=act(x)
   y=act(y)
   return (y)

def incProc(n):
   n=n+1
   return (n)

def anotherProc(n):
   n=n+7
   return (n)

x=5
x=incTwo(x, incProc)
print(x)

x=5
x=incTwo(x, anotherProc)
print(x)


#>>>
   7
  19
```

End