

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

```
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 2
1 3 3
1 4 6 4
1 5 10 10 5
1 6 15 20 15 6
1 7 21 35 35 21 7
1 8 28 56 70 56 28 8
1 9 36 84 126 126 84 36 9
1 10 45 120 210 252 210 120 45 10
1 11 55 165 330 462 462 330 165 55 11
1 12 66 252 504 792 924 792 504 252 66 12
1 13 78 351 780 1287 1716 1716 1287 780 351 78 13
1 14 91 462 1092 1960 2730 2730 1960 1092 462 91 14
1 15 105 595 1638 3439 5005 5005 3439 1638 595 105 15
1 16 120 756 2184 4640 7920 9240 7920 4640 2184 756 120 16
1 17 136 936 2730 6188 11628 16796 16796 11628 6188 2730 936 136 17
1 18 153 1140 3439 8445 16796 27300 27300 16796 8445 3439 1140 153 18
1 19 171 1378 4640 12870 27300 46400 46400 27300 12870 4640 1378 171 19
1 20 190 1653 6188 16796 34390 50050 50050 34390 16796 6188 1653 190 20
1 21 210 1960 8445 21840 46400 79200 79200 46400 21840 8445 1960 210 21
1 22 231 2301 10920 30030 61880 92400 92400 61880 30030 10920 2301 231 22
1 24 253 2706 14187 40497 84450 128700 128700 84450 40497 14187 2706 253 24
1 25 276 3150 18486 54285 116280 171600 171600 116280 54285 18486 3150 276 25
1 26 300 3640 23880 70540 154320 238800 238800 154320 70540 23880 3640 300 26
1 27 325 4185 30435 92730 207080 304350 304350 207080 92730 30435 4185 325 27
1 28 351 4786 38440 120660 273000 384400 384400 273000 120660 38440 4786 351 28
1 29 378 5445 48045 154320 343900 480450 480450 343900 154320 48045 5445 378 29
1 30 406 6165 60360 196000 464000 603600 603600 464000 196000 60360 6165 406 30
1 31 435 6951 75600 256200 598500 756000 756000 598500 256200 75600 6951 435 31
1 32 465 7812 94080 336480 819000 940800 940800 819000 336480 94080 7812 465 32
1 33 496 8760 116160 442560 1092000 1161600 1161600 1092000 442560 116160 8760 496 33
1 34 528 9800 142080 588000 1420800 1420800 1420800 588000 142080 142080 9800 528 34
1 35 561 10940 172080 784000 1720800 1720800 1720800 784000 172080 172080 10940 561 35
1 36 595 12180 207360 1041600 2073600 2073600 2073600 1041600 207360 207360 12180 595 36
1 37 630 13530 248160 1360800 2481600 2481600 2481600 1360800 248160 248160 13530 630 37
1 38 666 15000 294720 1756800 2947200 2947200 2947200 1756800 294720 294720 15000 666 38
1 39 703 16590 348480 2380800 3484800 3484800 3484800 2380800 348480 348480 16590 703 39
1 40 741 18300 410160 3206400 4101600 4101600 4101600 3206400 410160 410160 18300 741 40
1 41 780 19140 480960 4224000 4809600 4809600 4809600 4224000 480960 480960 19140 780 41
1 42 820 20120 561600 5443200 5616000 5616000 5616000 5443200 561600 561600 20120 820 42
1 43 861 21240 653760 6988800 6537600 6537600 6537600 6988800 653760 653760 21240 861 43
1 44 903 22500 758400 8870400 7584000 7584000 7584000 8870400 758400 758400 22500 903 44
1 45 946 23910 877440 11121600 8774400 8774400 8774400 11121600 877440 877440 23910 946 45
1 46 990 25470 1011840 13756800 10118400 10118400 10118400 13756800 1011840 1011840 25470 990 46
1 47 1035 27180 1162800 16800000 11628000 11628000 11628000 16800000 1162800 1162800 27180 1035 47
1 48 1081 29040 1340800 20280000 13408000 13408000 13408000 20280000 1340800 1340800 29040 1081 48
1 49 1128 31060 1546800 24216000 15468000 15468000 15468000 24216000 1546800 1546800 31060 1128 49
1 50 1176 33240 1782000 28740000 17820000 17820000 17820000 28740000 1782000 1782000 33240 1176 50
1 51 1225 35580 2047200 33984000 20472000 20472000 20472000 33984000 2047200 2047200 35580 1225 51
1 52 1275 38090 2343600 40080000 23436000 23436000 23436000 40080000 2343600 2343600 38090 1275 52
1 53 1326 40770 2672400 47160000 26724000 26724000 26724000 47160000 2672400 2672400 40770 1326 53
1 54 1378 43620 3035040 55368000 30350400 30350400 30350400 55368000 3035040 3035040 43620 1378 54
1 55 1431 46650 3432960 64860000 34329600 34329600 34329600 64860000 3432960 3432960 46650 1431 55
1 56 1485 49860 3866880 75840000 38668800 38668800 38668800 75840000 3866880 3866880 49860 1485 56
1 57 1540 53260 4348320 88464000 43483200 43483200 43483200 88464000 4348320 4348320 53260 1540 57
1 58 1596 56860 4878000 102960000 48780000 48780000 48780000 102960000 4878000 4878000 56860 1596 58
1 59 1653 60660 5457600 119520000 54576000 54576000 54576000 119520000 5457600 5457600 60660 1653 59
1 60 1711 64660 6088800 138360000 60888000 60888000 60888000 138360000 6088800 6088800 64660 1711 60
1 61 1770 68870 6772800 159600000 67728000 67728000 67728000 159600000 6772800 6772800 68870 1770 61
1 62 1830 73290 7520640 183480000 75206400 75206400 75206400 183480000 7520640 7520640 73290 1830 62
1 63 1891 77920 8334240 210160000 83342400 83342400 83342400 210160000 8334240 8334240 77920 1891 63
1 64 1953 82770 9216000 239840000 92160000 92160000 92160000 239840000 9216000 9216000 82770 1953 64
1 65 2016 87840 10178400 272760000 101784000 101784000 101784000 272760000 10178400 10178400 87840 2016 65
1 66 2080 93140 11222400 309240000 112224000 112224000 112224000 309240000 11222400 11222400 93140 2080 66
1 67 2145 98670 12350400 349680000 123504000 123504000 123504000 349680000 12350400 12350400 98670 2145 67
1 68 2211 104430 13564800 394440000 135648000 135648000 135648000 394440000 13564800 13564800 104430 2211 68
1 69 2278 110430 14868000 443920000 148680000 148680000 148680000 443920000 14868000 14868000 110430 2278 69
1 70 2346 116670 16262400 498600000 162624000 162624000 162624000 498600000 16262400 16262400 116670 2346 70
1 71 2415 123160 17750400 559040000 177504000 177504000 177504000 559040000 17750400 17750400 123160 2415 71
1 72 2485 129900 19336800 625800000 193368000 193368000 193368000 625800000 19336800 19336800 129900 2485 72
1 73 2556 136890 21016800 699440000 210168000 210168000 210168000 699440000 21016800 21016800 136890 2556 73
1 74 2628 144140 22804800 780520000 228048000 228048000 228048000 780520000 22804800 22804800 144140 2628 74
1 75 2701 151650 24705600 869600000 247056000 247056000 247056000 869600000 24705600 24705600 151650 2701 75
1 76 2775 159420 26724000 967360000 267240000 267240000 267240000 967360000 26724000 26724000 159420 2775 76
1 77 2850 167460 28864800 1074480000 288648000 288648000 288648000 1074480000 28864800 28864800 167460 2850 77
1 78 2926 175770 31132800 1191600000 311328000 311328000 311328000 1191600000 31132800 31132800 175770 2926 78
1 79 3003 184350 33532800 1319440000 335328000 335328000 335328000 1319440000 33532800 33532800 184350 3003 79
1 80 3081 193200 36070400 1458640000 360704000 360704000 360704000 1458640000 36070400 36070400 193200 3081 80
1 81 3160 192330 38750400 1609840000 387504000 387504000 387504000 1609840000 38750400 38750400 192330 3160 81
1 82 3240 191760 41577600 1773600000 415776000 415776000 415776000 1773600000 41577600 41577600 191760 3240 82
1 83 3321 191490 44556000 1950480000 445560000 445560000 445560000 1950480000 44556000 44556000 191490 3321 83
1 84 3403 191520 47692800 2141040000 476928000 476928000 476928000 2141040000 47692800 47692800 191520 3403 84
1 85 3486 191850 50992000 2346000000 509920000 509920000 509920000 2346000000 50992000 50992000 191850 3486 85
1 86 3570 192480 54460800 2566000000 544608000 544608000 544608000 2566000000 54460800 54460800 192480 3570 86
1 87 3655 193410 58104000 2801600000 581040000 581040000 581040000 2801600000 58104000 58104000 193410 3655 87
1 88 3741 194650 61928000 3053600000 619280000 619280000 619280000 3053600000 61928000 61928000 194650 3741 88
1 89 3828 196190 65937600 3322800000 659376000 659376000 659376000 3322800000 65937600 65937600 196190 3828 89
1 90 3916 198030 70137600 3610000000 701376000 701376000 701376000 3610000000 70137600 70137600 198030 3916 90
1 91 4005 199170 74534400 3916000000 745344000 745344000 745344000 3916000000 74534400 74534400 199170 4005 91
1 92 4095 200610 79132800 4241600000 791328000 791328000 791328000 4241600000 79132800 79132800 200610 4095 92
1 93 4186 202350 83939200 4587600000 839392000 839392000 839392000 4587600000 83939200 83939200 202350 4186 93
1 94 4278 204390 88960000 4954800000 889600000 889600000 889600000 4954800000 88960000 88960000 204390 4278 94
1 95 4371 206730 94201600 5344000000 942016000 942016000 942016000 5344000000 94201600 94201600 206730 4371 95
1 96 4465 209370 99670400 5756000000 996704000 996704000 996704000 5756000000 99670400 99670400 209370 4465 96
1 97 4560 212310 105372800 6191600000 1053728000 1053728000 1053728000 6191600000 105372800 105372800 212310 4560 97
1 98 4656 215550 111304000 6651600000 1113040000 1113040000 1113040000 6651600000 111304000 111304000 215550 4656 98
1 99 4753 219090 117470400 7136800000 1174704000 1174704000 1174704000 7136800000 117470400 117470400 219090 4753 99
1 100 4851 222930 123878400 7648000000 1238784000 1238784000 1238784000 7648000000 123878400 123878400 222930 4851 100
1 101 4950 227070 130524000 8186000000 1305240000 1305240000 1305240000 8186000000 130524000 130524000 227070 4950 101
1 102 5050 231510 137412000 8751600000 1374120000 1374120000 1374120000 8751600000 137412000 137412000 231510 5050 102
1 103 5151 236250 144548000 9345600000 1445480000 1445480000 1445480000 9345600000 144548000 144548000 236250 5151 103
1 104 5253 241290 151937600 9968800000 1519376000 1519376000 1519376000 9968800000 151937600 151937600 241290 5253 104
1 105 5356 246630 159585600 10622000000 1595856000 1595856000 1595856000 10622000000 159585600 159585600 246630 5356 105
1 106 5460 252270 167507200 11306000000 1675072000 1675072000 1675072000 11306000000 167507200 167507200 252270 5460 106
1 107 5565 258210 175718400 12021600000 1757184000 1757184000 1757184000 12021600000 175718400 175718400 258210 5565 107
1 108 5671 264450 184224000 12769600000 1842240000 1842240000 1842240000 12769600000 184224000 184224000 264450 5671 108
1 109 5778 270990 193030400 13550800000 1930304000 1930304000 1930304000 13550800000 193030400 193030400 270990 5778 109
1 110 5886 277830 202142400 14376000000 2021424000 2021424000 2021424000 14376000000 202142400 202142400 277830 5886 110
1 111 5995 284970 211564800 15246000000 2115648000 2115648000 2115648000 15246000000 211564800 211564800 284970 5995 111
1 112 6105 292410 221312000 16161600000 2213120000 2213120000 2213120000 16161600000 221312000 221312000 292410 6105 112
1 113 6216 300150 231390400 17123600000 2313904000 2313904000 2313904000 17123600000 231390400 231390400 300150 6216 113
1 114 6328 308190 241804800 18133600000 2418048000 2418048000 2418048000 18133600000 241804800 241804800 308190 6328 114
1 115 6441 316530 252560000 19192400000 2525600000 2525600000 2525600000 19192400000 252560000 252560000 316530 6441 115
1 116 6555 325170 263670400 20300800000 2636704000 2636704000 2636704000 20300800000 263670400 263670400 325170 6555 116
1 117 6670 334110 275139200 21459600000 2751392000 2751392000 2751392000 21459600000 275139200 275139200 334110 6670 117
1 118 6786 343350 286971200 22670400000 2869712000 2869712000 2869712000 22670400000 286971200 286971200 343350 6786 118
1 119 6903 352890 299171200 23934000000 2991712000 2991712000 2991712000 23934000000 299171200 299171200 352890 6903 119
1 120 7021 362730 311744000 25251200000 3117440000 3117440000 3117440000 25251200000 311744000 311744000 362730 7021 120
1 121 7140 372870 324694400 26623600000 3246944000 3246944000 3246944000 26623600000 324694400 324694400 372870 7140 121
1 122 7260 383310 338027200 28052000000 3380272000 3380272000 3380272000 28052000000 338027200 338027200 383310 7260 122
1 123 7381 394050 351747200 29538000000 3517472000 3517472000 3517472000 29538000000 351747200 351747200 394050 7381 123
1 124 7503 405090 365859200 31082400000 3658592000 3658592000 3658592000 31082400000 365859200 365859200 405090 7503 124
1 125 7626 416430 380368000 32696000000 3803680000 3803680000 3803680000 32696000000 380368000 380368000 416430 7626 125
1 126 7750 428070 395288000 34379600000 3952880000 3952880000 3952880000 34379600000 395288000 395288000 428070 7750 126
1 127 7875 440010 41

Khayyam triangle

```
Python 3.4.4 Shell
File Edit Shell Debug Options Window Help
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:18) [MSC v.1600 32-bit Intel] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Documents and Settings\admin\Desktop\intro-python\examples\khayyam.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

$f(n) = f(n-1) + f(n-2) \quad n \geq 2$

$f(1) = 1$

$f(n) = 0 \quad n \leq 0$

$$\begin{aligned} f(10) &= f(9) + f(8) \\ &= 34 + 21 \\ &= 55 \end{aligned}$$

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	ca	nxt	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 fm
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	nxt	n	cu	nxt
4		4	0	1
		3	1	1
		2	1	2
		1	2	3
		0	3	5

greatest common divisor

- $\text{Gcd}(54, 24) = 6$

a b

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

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	a ₁	a ₂	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:

$$N \text{ Factorial} == N! == N * (N-1) * (N-2) * (N-3) * \dots * 4 * 3 * 2 * 1$$

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
	6	6	1	6
			6	5
			30	4
			120	3
			360	2
			720	1
			<u>720</u>	

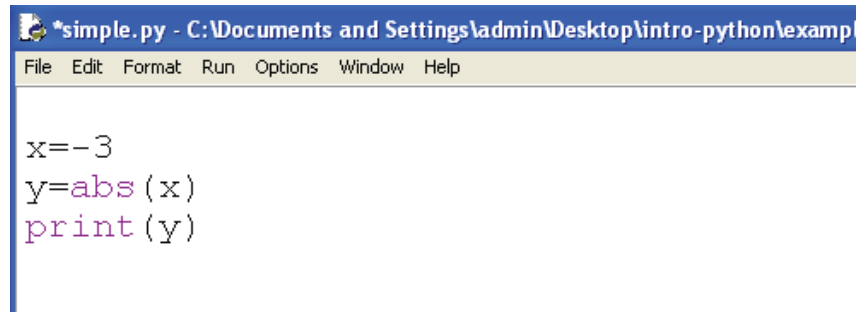
6 * 5
 6 * 5 * 4
 6 * 5 * 4 * 3
 6 * 5 * 4 * 3 * 2

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

Built-in Functions



A screenshot of a Python IDE window titled '*simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examp'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code editor contains the following Python code:

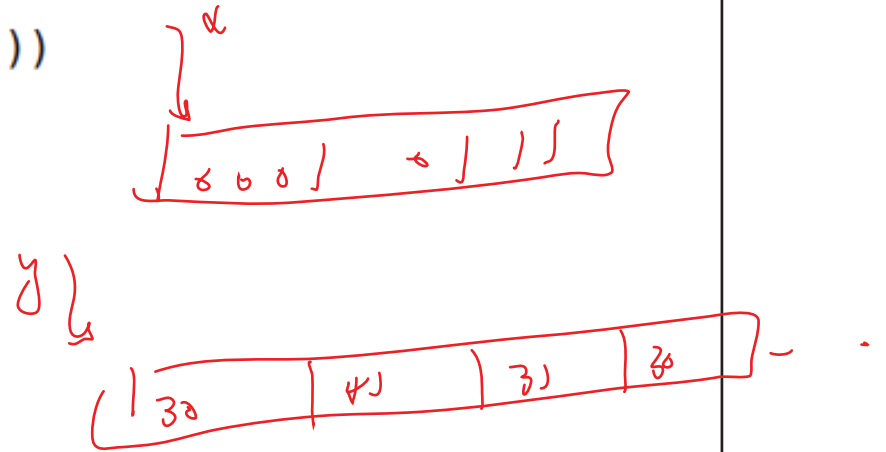
```
x=-3  
y=abs(x)  
print(y)
```

3

Built-in Functions

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

Enter a number: 23
0b10111



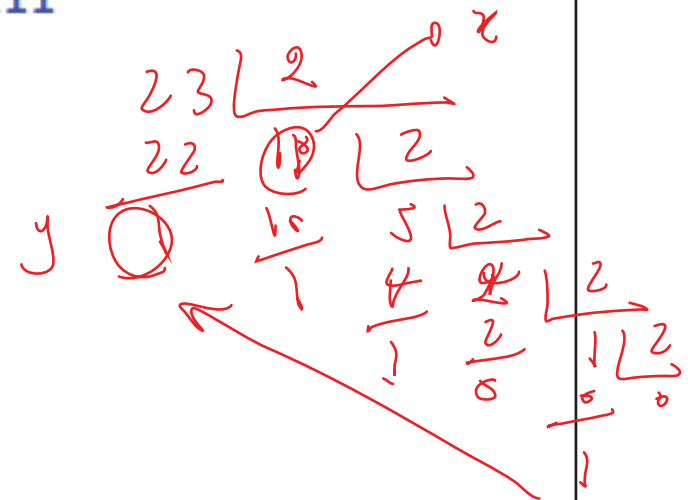
Built-in Functions

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

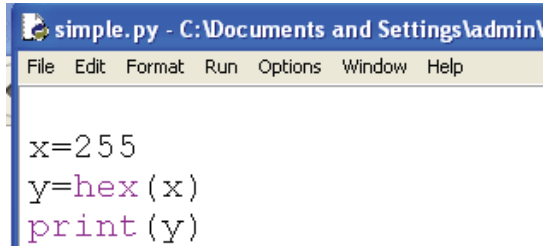
Enter a number: 23
0b10111

```
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)
```



- **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

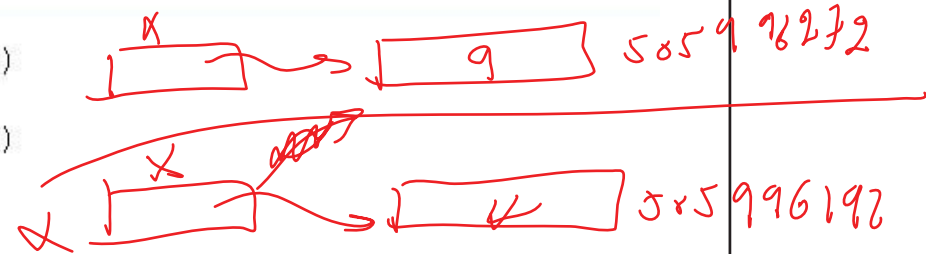


Parameter passing

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

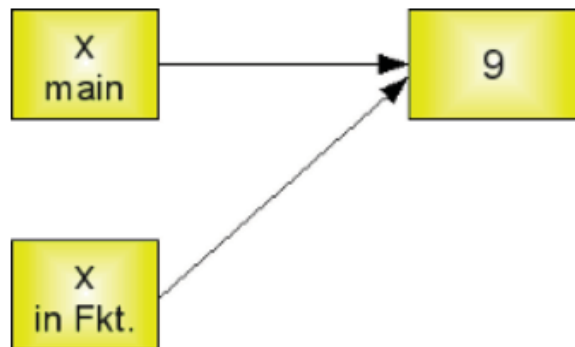
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19
tel)] on win32

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.

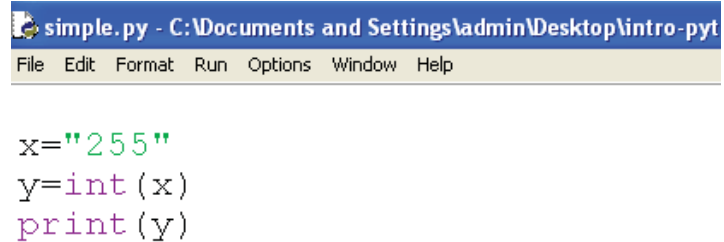
Built-in Type conversion Functions

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

Built-in Type conversion Functions

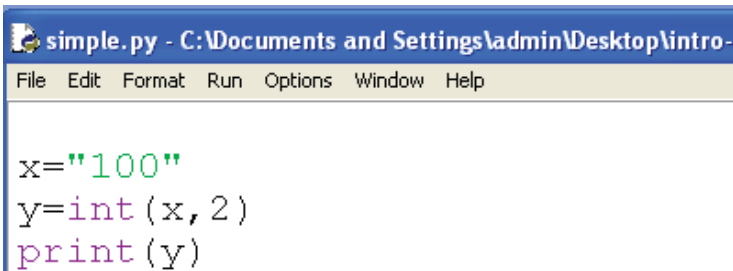
<code>set(s)</code>	Converts <code>s</code> to a set.
<code>dict(d)</code>	Creates a dictionary. <code>d</code> must be a sequence of (key,value) tuples.
<code>frozenset(s)</code>	Converts <code>s</code> to a frozen set.
<code>chr(x)</code>	Converts an integer to a character.
<code>unichr(x)</code>	Converts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

m { 1 1 1 1 1 1 0 1 0 1 } -37

{ 6 0 0 0 0 0 0 0 1 0 0 1 } 37

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

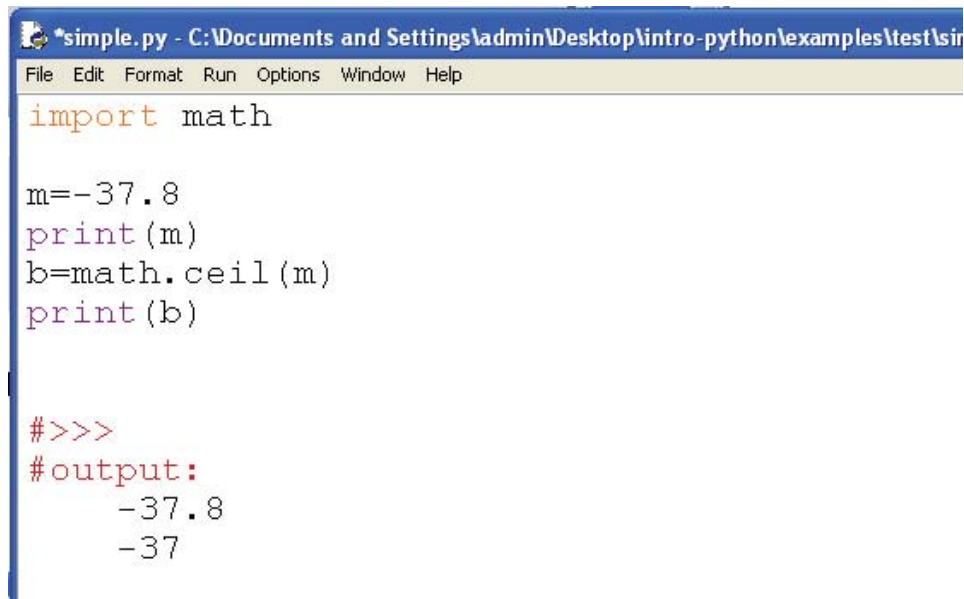
- You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax:

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\sin
File Edit Format Run Options Window Help
import math

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

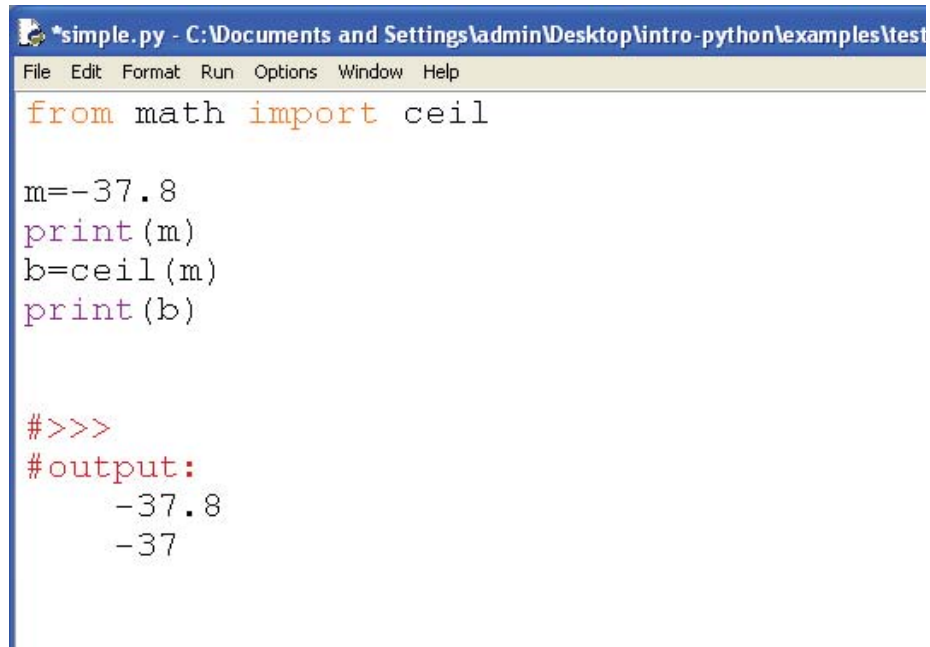
#>>>
#output:
-37.8
-37
```

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\test
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
```

Handwritten table illustrating the execution of the code:

incTwo		incProc	
x	act y	n	
5	5	5	5
		6	6
		7	6
			7

Diagram details: A green arrow points from the `incProc` function definition to the `act` parameter in the `incTwo` function definition. A red bracket groups the `incTwo` header and its first two rows. A red bracket groups the `incProc` header and its four rows.

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