

Program Code: J620-002-4:2020

Program Name: FRONT-END SOFTWARE DEVELOPMENT

Title: Logic and Repetition

N	la	m	e:
---	----	---	----

IC Number:

Date:

Introduction:

**Conclusion:** 

# Module P02: Logic and Repetition

In this module, we'll first understand what procedures are and why you might want to use them. Then you'll start to write some procedures. Next, we'll learn about Python logic, or how to make comparisons.

Finally, we'll go on to repetition, and how to repeat operations in Python using while and for loops.

- Procedural Abstraction
- Computer Logic
- Repetition

## **Procedural Abstraction**

What is a procedure? Why might you want to abstract it?

We use procedures because we want to stop us from doing tedious work. Anything we might want to do again and again we will want to 'abstract'.

## **Functions**

A function is a block of organized, reusable code that is used to perform a single, related action. For example, we can use the same procedure to multiply two numbers, whatever they are. We then specify the

numbers being multiplied as arguements of the function. Many functions return a value which you can then use; they can also print something out immediately

#### Syntax:

```
In [1]: def functionname( parameters ):
    "functionname ... syntax: blabla"
    block_of_code
    return [expression]
```

Notice that you can add some help text after the function header line, so that you can print the help text by:

```
In [2]: print((functionname. doc ))
        functionname ... syntax: blabla
        # void: does not return
In [4]:
        def myPrinter(text):
           print(text)
        myPrinter(234)
         # has return
        def multiplyer(i, j):
            return i*j
        print(multiplyer('hello', 3))
        print (multiplyer (4, 3) + 1)
        multiplyer(3, 4) # return nothing
        234
        hellohello
        13
Out[4]:
        def print info( name, age = 35, height=150, weight=60 ):
In [12]:
            "This prints a passed info into this function"
            print("Name: ", name)
            print("Age ", age)
            print("Height ", height)
            print("Weight ", weight)
            print()
        print info('he', 22, weight=65)
        print info('she')
        Name: he
        Age 22
        Height 150
        Weight 65
        Name: she
        Age 35
        Height 150
        Weight 60
```

**Quick Exercise 1** Let's revisit the Farenheit to Celsius conversion task. Write a function 'F2C' that converts a Fahrenheit value to Celsius, and returns the value.

# Logic

Can a computer think? How does a computer think differently from how we do? We've seen how computers can represent abstract concepts. One of those abstract concepts is logic. We've developed programs that can use logic.

A **boolean** expression is an expression that is either *true* or *false*. The following examples use the operator ==, which compares two operands and produces True if they are equal and False otherwise.

There are three logical operators: *and*, *or*, and *not*. The semantics (meaning) of these operators is similar to their meaning in English.

### String logic

```
is -- object identity

is not -- negated object identity

x in s -- True if an item of s is equal to x, else False

x not in s -- False if an item of s is equal to x, else True

x or y -- if x is false, then y, else x

x and y -- if x is false, then x, else y

not x -- if x is false, then True, else False
```

### **Arithmetic logic**

False True False True False

```
> -- Strictly larger than
== -- Is the identity of
>= -- Greater than or equal to
!= -- Is not the identity of.
```

```
In [14]: print((5 > 2))
    print((2 > 5))
    print((2 is 2))
    print((2 == 3))
    print((5 > 2 or 2 > 1))
    print((5 > 2 and 2 > 2))
    print(('s' in 'datascience'))
    print(('x' in 'datascience'))
True
False
True
```

```
<>:3: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:3: SyntaxWarning: "is" with a literal. Did you mean "=="?
<ipython-input-14-56adce85b2e3>:3: SyntaxWarning: "is" with a literal. Did you mean "=
="?
    print((2 is 2))
```

```
In [15]: # Why is == equals, not = ? Test it out
print((2 = 3))
```

#### **Conditional Execution**

Python does not use { } to enclose blocks of code for if/loops/function etc. like C. Instead, Python uses the colon (:) and indentation/whitespace to group statements.

In order to write useful programs, we almost always need the ability to check conditions and change the behavior of the program accordingly. Conditional statements give us this ability. The simplest form is the if statement:

```
if x > 0 :
    print 'x is positive'
```

The boolean expression after the if statement is called the condition. We end the if statement with a colon character (:) and the line(s) after the if statement are indented.

If the logical condition is true, then the indented statement gets executed. If the logical condition is false, the indented statement is skipped.

#### Alternative execution

A second form of the if statement is alternative execution, in which there are two possibilities and the condition determines which one gets executed. The syntax looks like this:

```
if x%2 == 0 :
    print 'x is even'
else :
    print 'x is odd'
```

If the remainder when x is divided by 2 is 0, then we know that x is even, and the program displays a message to that effect. If the condition is false, the second set of statements is executed.

Since the condition must be true or false, exactly one of the alternatives will be executed. The alternatives are called branches, because they are branches in the flow of execution.

## **Chained conditionals**

Sometimes there are more than two possibilities and we need more than two branches. One way to express a computation like that is a chained conditional:

```
if x < y:
    print 'x is less than y'
elif x > y:
    print 'x is greater than y'
else:
    print 'x and y are equal'
elif is an abbreviation of "else if". Again, exactly one branch will be executed.
```

There is no limit on the number of elif statements. If there is an else clause, it has to be at the end, but there doesn't have to be one.

```
if False:
In [20]:
            print("block of code 1")
         elif True:
            print("block of code 2")
         else:
            print("block of code 3")
        block of code 2
        speed = 105
In [27]:
         mood = 'terrible'
         if speed >= 80:
             print('License and registration please')
            if mood == 'terrible' and speed >= 100:
                print('You have the right to remain silent.')
             elif mood == 'bad' or speed >= 90:
                 print("I'm going to have to write you a ticket.")
             else:
                 print("Let's try to keep it under 80 ok?")
        License and registration please
        You have the right to remain silent.
In [31]: #string = 'Hi there' # True example
         string = 'Good bye' # False example
         result = string.find('th')
        print(result)
         if result != -1:
            print('Success!')
         else:
            print('Not found!')
        -1
        Not found!
In [38]: # Define starts_with B('Boyce')
         def starts with B(name):
             if name[0] == 'B' or name[0] == 'b':
                print("Yes")
         # call the function
         starts with B('Boyce')
        Yes
```

In [51]: # Define bigger(a,b)
def bigger(a, b):

```
if a > b:
    return a
else:
    return b

# quick alternative
#return max(a, b)
bigger(45, -64)
```

Out[51]: 45

**Quick Exercise 2** Nested if problem: Define a function biggest(a,b,c) which takes the largest of the three inputs and returns the largest one.

```
In [93]: # Write your code here
```

# Repetition

### While

The while loop continues iterating until it's condition stops being true:

```
In [47]: i = 0
         while i < 10:
            i = i + 1
             print(i)
         1
         2
         3
         4
         5
         6
         7
         8
         9
         10
In [17]: # What happens here?
         # Try guessing. Run at your own risk! :)
         \#i = 0
         #while i != 11:
         \# i = i+2
              print i
         2
         4
         6
         10
         12
         14
         16
         18
         20
         22
         24
         26
         28
         30
         32
```

```
826
828
830
832
834
836
838
840
842
844
846
848
850
852
854
856
858
860
862
864
866
868
870
872
874
876
878
880
882
884
886
888
890
892
894
896
898
900
902
904
906
908
910
912
914
916
918
920
922
924
926
928
930
932
934
936
938
940
942
944
946
948
950
952
```

```
2542
2544
2546
2548
2550
2552
2554
2556
2558
2560
2562
2564
2566
2568
2570
2572
2574
2576
2578
2580
2582
2584
2586
2588
2590
2592
2594
2596
2598
2600
2602
2604
2606
2608
2610
2612
2614
2616
2618
2620
2622
2624
2626
2628
2630
2632
2634
2636
2638
2640
2642
2644
2646
2648
2650
2652
2654
2656
2658
2660
2662
2664
2666
```

```
2938
2940
2942
2944
2946
2948
2950
2952
2954
2956
2958
2960
2962
2964
2966
2968
2970
2972
2974
2976
2978
2980
2982
2984
2986
2988
2990
2992
2994
2996
2998
3000
3002
3004
3006
3008
3010
3012
3014
3016
3018
3020
3022
3024
3026
3028
3030
3032
3034
3036
3038
3040
3042
3044
3046
3048
3050
3052
3054
3056
3058
3060
3062
3064
```

```
6238
6240
6242
6244
6246
6248
6250
6252
6254
6256
6258
6260
6262
6264
6266
6268
6270
6272
6274
6276
6278
6280
6282
6284
6286
6288
6290
6292
6294
6296
6298
6300
6302
6304
6306
6308
6310
6312
6314
6316
6318
6320
6322
6324
6326
6328
6330
6332
6334
6336
6338
6340
6342
6344
6346
6348
6350
6352
6354
6356
6358
6360
6362
6364
```

```
6502
6504
6506
6508
6510
6512
6514
6516
6518
6520
6522
6524
6526
6528
6530
6532
6534
6536
6538
6540
6542
6544
6546
6548
6550
6552
6554
6556
6558
6560
6562
6564
6566
6568
6570
6572
6574
6576
6578
6580
6582
6584
6586
6588
6590
6592
6594
6596
6598
6600
6602
6604
6606
6608
6610
6612
6614
6616
6618
6620
6622
6624
6626
6628
```

```
8350
8352
8354
8356
8358
8360
8362
8364
8366
8368
8370
8372
8374
8376
8378
8380
8382
8384
8386
8388
8390
8392
8394
8396
8398
8400
8402
8404
8406
8408
8410
8412
8414
8416
8418
8420
8422
8424
8426
8428
8430
8432
8434
8436
8438
8440
8442
8444
8446
8448
8450
8452
8454
8456
8458
8460
8462
8464
8466
8468
8470
8472
8474
8476
```

```
8482
8484
8486
8488
8490
8492
8494
8496
8498
8500
8502
8504
8506
8508
8510
8512
8514
8516
8518
8520
8522
8524
8526
8528
8530
8532
8534
8536
8538
8540
8542
8544
8546
8548
8550
8552
8554
8556
8558
8560
8562
8564
8566
8568
8570
8572
8574
8576
8578
8580
8582
8584
8586
8588
8590
8592
8594
8596
8598
8600
8602
8604
8606
8608
```

```
8878
8880
8882
8884
8886
8888
8890
8892
8894
8896
8898
8900
8902
8904
8906
8908
8910
8912
8914
8916
8918
8920
8922
8924
8926
8928
8930
8932
8934
8936
8938
8940
8942
8944
8946
8948
8950
8952
8954
8956
8958
8960
8962
8964
8966
8968
8970
8972
8974
8976
8978
8980
8982
8984
8986
8988
8990
8992
8994
8996
8998
9000
9002
9004
```

```
9274
9276
9278
9280
9282
9284
9286
9288
9290
9292
9294
9296
9298
9300
9302
9304
9306
9308
9310
9312
9314
9316
9318
9320
9322
9324
9326
9328
9330
9332
9334
9336
9338
9340
9342
9344
9346
9348
9350
9352
9354
9356
9358
9360
9362
9364
9366
9368
9370
9372
9374
9376
9378
9380
9382
9384
9386
9388
9390
9392
9394
9396
9398
9400
```

```
9406
9408
9410
9412
9414
9416
9418
9420
9422
9424
9426
9428
9430
9432
9434
9436
9438
9440
9442
9444
9446
9448
9450
9452
9454
9456
9458
9460
9462
9464
9466
9468
9470
9472
9474
9476
9478
9480
9482
9484
9486
9488
9490
9492
9494
9496
9498
9500
9502
9504
9506
9508
9510
9512
9514
9516
9518
9520
9522
9524
9526
9528
9530
9532
```

```
9538
9540
9542
9544
9546
9548
9550
9552
9554
9556
9558
9560
9562
9564
9566
9568
9570
9572
9574
9576
9578
9580
9582
9584
9586
9588
9590
9592
9594
9596
9598
9600
9602
9604
9606
9608
9610
9612
9614
9616
9618
9620
9622
9624
9626
9628
9630
9632
9634
9636
9638
9640
9642
9644
9646
9648
9650
9652
9654
9656
9658
9660
9662
```

```
9802
9804
9806
9808
9810
9812
9814
9816
9818
9820
9822
9824
9826
9828
9830
9832
9834
9836
9838
9840
9842
9844
9846
9848
9850
9852
9854
9856
9858
9860
9862
9864
9866
9868
9870
9872
9874
9876
9878
9880
9882
9884
9886
9888
9890
9892
9894
9896
9898
9900
9902
9904
9906
9908
9910
9912
9914
9916
9918
9920
9922
9924
9926
```

```
23926
23928
23930
23932
23934
23936
23938
23940
23942
23944
23946
23948
23950
23952
23954
23956
23958
23960
23962
23964
23966
23968
23970
23972
23974
23976
23978
23980
23982
23984
23986
23988
23990
23992
23994
23996
23998
24000
24002
24004
24006
24008
24010
24012
24014
24016
24018
24020
24022
24024
24026
24028
24030
24032
24034
24036
24038
24040
24042
24044
24046
24048
24050
24052
```

```
24322
24324
24326
24328
24330
24332
24334
24336
24338
24340
24342
24344
24346
24348
24350
24352
24354
24356
24358
24360
24362
24364
24366
24368
24370
24372
24374
24376
24378
24380
24382
24384
24386
24388
24390
24392
24394
24396
24398
24400
24402
24404
24406
24408
24410
24412
24414
24416
24418
24420
24422
24424
24426
24428
24430
24432
24434
24436
24438
24440
24442
24444
24446
24448
```

```
24718
24720
24722
24724
24726
24728
24730
24732
24734
24736
24738
24740
24742
24744
24746
24748
24750
24752
24754
24756
24758
24760
24762
24764
24766
24768
24770
24772
24774
24776
24778
24780
24782
24784
24786
24788
24790
24792
24794
24796
24798
24800
24802
24804
24806
24808
24810
24812
24814
24816
24818
24820
24822
24824
24826
24828
24830
24832
24834
24836
24838
24840
24842
```

```
27622
27624
27626
27628
27630
27632
27634
27636
27638
27640
27642
27644
27646
27648
27650
27652
27654
27656
27658
27660
27662
27664
27666
27668
27670
27672
27674
27676
27678
27680
27682
27684
27686
27688
27690
27692
27694
27696
27698
27700
27702
27704
27706
27708
27710
27712
27714
27716
27718
27720
27722
27724
27726
27728
27730
27732
27734
27736
27738
27740
27742
27744
27746
27748
```

```
34222
34224
34226
34228
34230
34232
34234
34236
34238
34240
34242
34244
34246
34248
34250
34252
34254
34256
34258
34260
34262
34264
34266
34268
34270
34272
34274
34276
34278
34280
34282
34284
34286
34288
34290
34292
34294
34296
34298
34300
34302
34304
34306
34308
34310
34312
34314
34316
34318
34320
34322
34324
34326
34328
34330
34332
34334
34336
34338
34340
34342
34344
34346
34348
```

```
34618
34620
34622
34624
34626
34628
34630
34632
34634
34636
34638
34640
34642
34644
34646
34648
34650
34652
34654
34656
34658
34660
34662
34664
34666
34668
34670
34672
34674
34676
34678
34680
34682
34684
34686
34688
34690
34692
34694
34696
34698
34700
34702
34704
34706
34708
34710
34712
34714
34716
34718
34720
34722
34724
34726
34728
34730
34732
34734
34736
34738
34740
34742
34744
```

```
40822
40824
40826
40828
40830
40832
40834
40836
40838
40840
40842
40844
40846
40848
40850
40852
40854
40856
40858
40860
40862
40864
40866
40868
40870
40872
40874
40876
40878
40880
40882
40884
40886
40888
40890
40892
40894
40896
40898
40900
40902
40904
40906
40908
40910
40912
40914
40916
40918
40920
40922
40924
40926
40928
40930
40932
40934
40936
40938
40940
40942
40944
40946
40948
```

```
42538
42540
42542
42544
42546
42548
42550
42552
42554
42556
42558
42560
42562
42564
42566
42568
42570
42572
42574
42576
42578
42580
42582
42584
42586
42588
42590
42592
42594
42596
42598
42600
42602
42604
42606
42608
42610
42612
42614
42616
42618
42620
42622
42624
42626
42628
42630
42632
42634
42636
42638
42640
42642
42644
42646
42648
42650
42652
42654
42656
42658
42660
42662
42664
```

```
42934
42936
42938
42940
42942
42944
42946
42948
42950
42952
42954
42956
42958
42960
42962
42964
42966
42968
42970
42972
42974
42976
42978
42980
42982
42984
42986
42988
42990
42992
42994
42996
42998
43000
43002
43004
43006
43008
43010
43012
43014
43016
43018
43020
43022
43024
43026
43028
43030
43032
43034
43036
43038
43040
43042
43044
43046
43048
43050
43052
43054
43056
43058
43060
```

```
44518
44520
44522
44524
44526
44528
44530
44532
44534
44536
44538
44540
44542
44544
44546
44548
44550
44552
44554
44556
44558
44560
44562
44564
44566
44568
44570
44572
44574
44576
44578
44580
44582
44584
44586
44588
44590
44592
44594
44596
44598
44600
44602
44604
44606
44608
44610
44612
44614
44616
44618
44620
44622
44624
44626
44628
44630
44632
44634
44636
44638
44640
44642
44644
```

```
47422
47424
47426
47428
47430
47432
47434
47436
47438
47440
47442
47444
47446
47448
47450
47452
47454
47456
47458
47460
47462
47464
47466
47468
47470
47472
47474
47476
47478
47480
47482
47484
47486
47488
47490
47492
47494
47496
47498
47500
47502
47504
47506
47508
47510
47512
47514
47516
47518
47520
47522
47524
47526
47528
47530
47532
47534
47536
47538
47540
47542
47544
47546
47548
```

```
47818
47820
47822
47824
47826
47828
47830
47832
47834
47836
47838
47840
47842
47844
47846
47848
47850
47852
47854
47856
47858
47860
47862
47864
47866
47868
47870
47872
47874
47876
47878
47880
47882
47884
47886
47888
47890
47892
47894
47896
47898
47900
47902
47904
47906
47908
47910
47912
47914
47916
47918
47920
47922
47924
47926
47928
47930
47932
47934
47936
47938
47940
47942
47944
```

```
52834
52836
52838
52840
52842
52844
52846
52848
52850
52852
52854
52856
52858
52860
52862
52864
52866
52868
52870
52872
52874
52876
52878
52880
52882
52884
52886
52888
52890
52892
52894
52896
52898
52900
52902
52904
52906
52908
52910
52912
52914
52916
52918
52920
52922
52924
52926
52928
52930
52932
52934
52936
52938
52940
52942
52944
52946
52948
52950
52952
52954
52956
52958
52960
```

```
54022
54024
54026
54028
54030
54032
54034
54036
54038
54040
54042
54044
54046
54048
54050
54052
54054
54056
54058
54060
54062
54064
54066
54068
54070
54072
54074
54076
54078
54080
54082
54084
54086
54088
54090
54092
54094
54096
54098
54100
54102
54104
54106
54108
54110
54112
54114
54116
54118
54120
54122
54124
54126
54128
54130
54132
54134
54136
54138
54140
54142
54144
54146
54148
```

```
54418
54420
54422
54424
54426
54428
54430
54432
54434
54436
54438
54440
54442
54444
54446
54448
54450
54452
54454
54456
54458
54460
54462
54464
54466
54468
54470
54472
54474
54476
54478
54480
54482
54484
54486
54488
54490
54492
54494
54496
54498
54500
54502
54504
54506
54508
54510
54512
54514
54516
54518
54520
54522
54524
54526
54528
54530
54532
54534
54536
54538
54540
54542
54544
```

```
54814
54816
54818
54820
54822
54824
54826
54828
54830
54832
54834
54836
54838
54840
54842
54844
54846
54848
54850
54852
54854
54856
54858
54860
54862
54864
54866
54868
54870
54872
54874
54876
54878
54880
54882
54884
54886
54888
54890
54892
54894
54896
54898
54900
54902
54904
54906
54908
54910
54912
54914
54916
54918
54920
54922
54924
54926
54928
54930
54932
54934
54936
54938
54940
```

```
62338
62340
62342
62344
62346
62348
62350
62352
62354
62356
62358
62360
62362
62364
62366
62368
62370
62372
62374
62376
62378
62380
62382
62384
62386
62388
62390
62392
62394
62396
62398
62400
62402
62404
62406
62408
62410
62412
62414
62416
62418
62420
62422
62424
62426
62428
62430
62432
62434
62436
62438
62440
62442
62444
62446
62448
62450
62452
62454
62456
62458
62460
62462
62464
62466
```

```
62866
62868
62870
62872
62874
62876
62878
62880
62882
62884
62886
62888
62890
62892
62894
62896
62898
62900
62902
62904
62906
62908
62910
62912
62914
62916
62918
62920
62922
62924
62926
62928
62930
62932
62934
62936
62938
62940
62942
62944
62946
62948
62950
62952
62954
62956
62958
62960
62962
62964
62966
62968
62970
62972
62974
62976
62978
62980
62982
62984
62986
62988
62990
62992
```

```
63526
63528
63530
63532
63534
63536
63538
63540
63542
63544
63546
63548
63550
63552
63554
63556
63558
63560
63562
63564
63566
63568
63570
63572
63574
63576
63578
63580
63582
63584
63586
63588
63590
63592
63594
63596
63598
63600
63602
63604
63606
63608
63610
63612
63614
63616
63618
63620
63622
63624
63626
63628
63630
63632
63634
63636
63638
63640
63642
63644
63646
63648
63650
63652
```

```
63790
63792
63794
63796
63798
63800
63802
63804
63806
63808
63810
63812
63814
63816
63818
63820
63822
63824
63826
63828
63830
63832
63834
63836
63838
63840
63842
63844
63846
63848
63850
63852
63854
63856
63858
63860
63862
63864
63866
63868
63870
63872
63874
63876
63878
63880
63882
63884
63886
63888
63890
63892
63894
63896
63898
63900
63902
63904
63906
63908
63910
63912
63914
63916
63918
```

```
63922
63924
63926
63928
63930
63932
63934
63936
63938
63940
63942
63944
63946
63948
63950
63952
63954
63956
63958
63960
63962
63964
63966
63968
63970
63972
63974
63976
63978
63980
63982
63984
63986
63988
63990
63992
63994
63996
63998
64000
64002
64004
64006
64008
64010
64012
64014
64016
64018
64020
64022
64024
64026
64028
64030
64032
64034
64036
64038
64040
64042
64044
64046
64048
```

```
64318
64320
64322
64324
64326
64328
64330
64332
64334
64336
64338
64340
64342
64344
64346
64348
64350
64352
64354
64356
64358
64360
64362
64364
64366
64368
64370
64372
64374
64376
64378
64380
64382
64384
64386
64388
64390
64392
64394
64396
64398
64400
64402
64404
64406
64408
64410
64412
64414
64416
64418
64420
64422
64424
64426
64428
64430
64432
64434
64436
64438
64440
64442
64444
```

```
64582
64584
64586
64588
64590
64592
64594
64596
64598
64600
64602
64604
64606
64608
64610
64612
64614
64616
64618
64620
64622
64624
64626
64628
64630
64632
64634
64636
64638
64640
64642
64644
64646
64648
64650
64652
64654
64656
64658
64660
64662
64664
64666
64668
64670
64672
64674
64676
64678
64680
64682
64684
64686
64688
64690
64692
64694
64696
64698
64700
64702
64704
64706
64708
```

```
64846
64848
64850
64852
64854
64856
64858
64860
64862
64864
64866
64868
64870
64872
64874
64876
64878
64880
64882
64884
64886
64888
64890
64892
64894
64896
64898
64900
64902
64904
64906
64908
64910
64912
64914
64916
64918
64920
64922
64924
64926
64928
64930
64932
64934
64936
64938
64940
64942
64944
64946
64948
64950
64952
64954
64956
64958
64960
64962
64964
64966
64968
64970
64972
```

```
82534
82536
82538
82540
82542
82544
82546
82548
82550
82552
82554
82556
82558
82560
82562
82564
82566
82568
82570
82572
82574
82576
82578
82580
82582
82584
82586
82588
82590
82592
82594
82596
82598
82600
82602
82604
82606
82608
82610
82612
82614
82616
82618
82620
82622
82624
82626
82628
82630
82632
82634
82636
82638
82640
82642
82644
82646
82648
82650
82652
82654
82656
82658
82660
```

```
82930
82932
82934
82936
82938
82940
82942
82944
82946
82948
82950
82952
82954
82956
82958
82960
82962
82964
82966
82968
82970
82972
82974
82976
82978
82980
82982
82984
82986
82988
82990
82992
82994
82996
82998
83000
83002
83004
83006
83008
83010
83012
83014
83016
83018
83020
83022
83024
83026
83028
83030
83032
83034
83036
83038
83040
83042
83044
83046
83048
83050
83052
83054
83056
```

```
84118
84120
84122
84124
84126
84128
84130
84132
84134
84136
84138
84140
84142
84144
84146
84148
84150
84152
84154
84156
84158
84160
84162
84164
84166
84168
84170
84172
84174
84176
84178
84180
84182
84184
84186
84188
84190
84192
84194
84196
84198
84200
84202
84204
84206
84208
84210
84212
84214
84216
84218
84220
84222
84224
84226
84228
84230
84232
84234
84236
84238
84240
84242
84244
```

```
84778
84780
84782
84784
84786
84788
84790
84792
84794
84796
84798
84800
84802
84804
84806
84808
84810
84812
84814
84816
84818
84820
84822
84824
84826
84828
84830
84832
84834
84836
84838
84840
84842
84844
84846
84848
84850
84852
84854
84856
84858
84860
84862
84864
84866
84868
84870
84872
84874
84876
84878
84880
84882
84884
84886
84888
84890
84892
84894
84896
84898
84900
84902
84904
```

```
92566
92568
92570
92572
92574
92576
92578
92580
92582
92584
92586
92588
92590
92592
92594
92596
92598
92600
92602
92604
92606
92608
92610
92612
92614
92616
92618
92620
92622
92624
92626
92628
92630
92632
92634
92636
92638
92640
92642
92644
92646
92648
92650
92652
92654
92656
92658
92660
92662
92664
92666
92668
92670
92672
92674
92676
92678
92680
92682
92684
92686
92688
92690
92692
```

```
92830
92832
92834
92836
92838
92840
92842
92844
92846
92848
92850
92852
92854
92856
92858
92860
92862
92864
92866
92868
92870
92872
92874
92876
92878
92880
92882
92884
92886
92888
92890
92892
92894
92896
92898
92900
92902
92904
92906
92908
92910
92912
92914
92916
92918
92920
92922
92924
92926
92928
92930
92932
92934
92936
92938
92940
92942
92944
92946
92948
92950
92952
92954
92956
```

```
93622
93624
93626
93628
93630
93632
93634
93636
93638
93640
93642
93644
93646
93648
93650
93652
93654
93656
93658
93660
93662
93664
93666
93668
93670
93672
93674
93676
93678
93680
93682
93684
93686
93688
93690
93692
93694
93696
93698
93700
93702
93704
93706
93708
93710
93712
93714
93716
93718
93720
93722
93724
93726
93728
93730
93732
93734
93736
93738
93740
93742
93744
93746
93748
```

```
94018
94020
94022
94024
94026
94028
94030
94032
94034
94036
94038
94040
94042
94044
94046
94048
94050
94052
94054
94056
94058
94060
94062
94064
94066
94068
94070
94072
94074
94076
94078
94080
94082
94084
94086
94088
94090
94092
94094
94096
94098
94100
94102
94104
94106
94108
94110
94112
94114
94116
94118
94120
94122
94124
94126
94128
94130
94132
94134
94136
94138
94140
94142
94144
```

```
94150
94152
94154
94156
94158
94160
94162
94164
94166
94168
94170
94172
94174
94176
94178
94180
94182
94184
94186
94188
94190
94192
94194
94196
94198
94200
94202
94204
94206
94208
94210
94212
94214
94216
94218
94220
94222
94224
94226
94228
94230
94232
94234
94236
94238
94240
94242
94244
94246
94248
94250
94252
94254
94256
94258
94260
94262
94264
94266
94268
94270
94272
94274
94276
```

```
94282
94284
94286
94288
94290
94292
94294
94296
94298
94300
94302
94304
94306
94308
94310
94312
94314
94316
94318
94320
94322
94324
94326
94328
94330
94332
94334
94336
94338
94340
94342
94344
94346
94348
94350
94352
94354
94356
94358
94360
94362
94364
94366
94368
94370
94372
94374
94376
94378
94380
94382
94384
94386
94388
94390
94392
94394
94396
94398
94400
94402
94404
94406
94408
```

```
94414
94416
94418
94420
94422
94424
94426
94428
94430
94432
94434
94436
94438
94440
94442
94444
94446
94448
94450
94452
94454
94456
94458
94460
94462
94464
94466
94468
94470
94472
94474
94476
94478
94480
94482
94484
94486
94488
94490
94492
94494
94496
94498
94500
94502
94504
94506
94508
94510
94512
94514
94516
94518
94520
94522
94524
94526
94528
94530
94532
94534
94536
94538
94540
```

```
94546
94548
94550
94552
94554
94556
94558
94560
94562
94564
94566
94568
94570
94572
94574
94576
94578
94580
94582
94584
94586
94588
94590
94592
94594
94596
94598
94600
94602
94604
94606
94608
94610
94612
94614
94616
94618
94620
94622
94624
94626
94628
94630
94632
94634
94636
94638
94640
94642
94644
94646
94648
94650
94652
94654
94656
94658
94660
94662
94664
94666
94668
94670
94672
94674
94676
```

```
94678
94680
94682
94684
94686
94688
94690
94692
94694
94696
94698
94700
94702
94704
94706
94708
94710
94712
94714
94716
94718
94720
94722
94724
94726
94728
94730
94732
94734
94736
94738
94740
94742
94744
94746
94748
94750
94752
94754
94756
94758
94760
94762
94764
94766
94768
94770
94772
94774
94776
94778
94780
94782
94784
94786
94788
94790
94792
94794
94796
94798
94800
94802
94804
```

```
94810
94812
94814
94816
94818
94820
94822
94824
94826
94828
94830
94832
94834
94836
94838
94840
94842
94844
94846
94848
94850
94852
94854
94856
94858
94860
94862
94864
94866
94868
94870
94872
94874
94876
94878
94880
94882
94884
94886
94888
94890
94892
94894
94896
94898
94900
94902
94904
94906
94908
94910
94912
94914
94916
94918
94920
94922
94924
94926
94928
94930
94932
94934
94936
```

```
94942
94944
94946
94948
94950
94952
94954
94956
94958
94960
94962
94964
94966
94968
94970
94972
94974
94976
94978
94980
94982
94984
94986
94988
94990
94992
94994
94996
94998
95000
95002
95004
95006
95008
95010
95012
95014
95016
95018
95020
95022
95024
95026
95028
95030
95032
95034
95036
95038
95040
95042
95044
95046
95048
95050
95052
95054
95056
95058
95060
95062
95064
95066
95068
```

```
95338
95340
95342
95344
95346
95348
95350
95352
95354
95356
95358
95360
95362
95364
95366
95368
95370
95372
95374
95376
95378
95380
95382
95384
95386
95388
95390
95392
95394
95396
95398
95400
95402
95404
95406
95408
95410
95412
95414
95416
95418
95420
95422
95424
95426
95428
95430
95432
95434
95436
95438
95440
95442
95444
95446
95448
95450
95452
95454
95456
95458
95460
95462
95464
```

```
96922
96924
96926
96928
96930
96932
96934
96936
96938
96940
96942
96944
96946
96948
96950
96952
96954
96956
96958
96960
96962
96964
96966
96968
96970
96972
96974
96976
96978
96980
96982
96984
96986
96988
96990
96992
96994
96996
96998
97000
97002
97004
97006
97008
97010
97012
97014
97016
97018
97020
97022
97024
97026
97028
97030
97032
97034
97036
97038
97040
97042
97044
97046
97048
```

```
98374
98376
98378
98380
98382
98384
98386
98388
98390
98392
98394
98396
98398
98400
98402
98404
98406
98408
98410
98412
98414
98416
98418
98420
98422
98424
98426
98428
98430
98432
98434
98436
98438
98440
98442
98444
98446
98448
98450
98452
98454
98456
98458
98460
98462
98464
98466
98468
98470
98472
98474
98476
98478
98480
98482
98484
98486
98488
98490
98492
98494
98496
98498
98500
```

```
99826
99828
99830
99832
99834
99836
99838
99840
99842
99844
99846
99848
99850
99852
99854
99856
99858
99860
99862
99864
99866
99868
99870
99872
99874
99876
99878
99880
99882
99884
99886
99888
99890
99892
99894
99896
99898
99900
99902
99904
99906
99908
99910
99912
99914
99916
99918
99920
99922
99924
99926
99928
99930
99932
99934
99936
99938
99940
99942
99944
99946
99948
99950
99952
```

\_\_\_\_\_

```
KeyboardInterrupt
                                          Traceback (most recent call last)
<ipython-input-17-dbd07f5666f0> in <module>
      4 while i != 11:
            i = i+2
---> 6
           print (i)
D:\Anaconda3\envs\python-dscourse\lib\site-packages\ipykernel\iostream.py in write(self,
 string)
    402
                    is child = (not self. is master process())
    403
                    # only touch the buffer in the IO thread to avoid races
--> 404
                    self.pub thread.schedule(lambda : self. buffer.write(string))
    405
                    if is child:
    406
                        # mp.Pool cannot be trusted to flush promptly (or ever),
D:\Anaconda3\envs\python-dscourse\lib\site-packages\ipykernel\iostream.py in schedule(se
lf, f)
    203
                    self. events.append(f)
                    # wake event thread (message content is ignored)
   204
--> 205
                    self. event pipe.send(b'')
    206
                else:
    207
D:\Anaconda3\envs\python-dscourse\lib\site-packages\zmq\sugar\socket.py in send(self, da
ta, flags, copy, track, routing id, group)
    414
                                         copy threshold=self.copy threshold)
    415
                    data.group = group
--> 416
               return super (Socket, self).send (data, flags=flags, copy=copy, track=trac
    417
    418
            def send multipart(self, msg parts, flags=0, copy=True, track=False, **kwarg
s):
zmq/backend/cython/socket.pyx in zmq.backend.cython.socket.Socket.send()
zmq/backend/cython/socket.pyx in zmq.backend.cython.socket.Socket.send()
zmq/backend/cython/socket.pyx in zmq.backend.cython.socket. send copy()
D:\Anaconda3\envs\python-dscourse\lib\site-packages\zmq\backend\cython\checkrc.pxd in zm
q.backend.cython.checkrc. check rc()
KeyboardInterrupt:
```

## For

The for-loop iterates for all elements in a data structure (can be an array, list, dict, data structures that behave like iterators in general).

If we do not have a so-called container of elements, we can create a list of numbers on the go, and have the for loop to cycle through it... (but of course this has not much use, normally we will have a container of data)

A while-loop operates in a slightly different way. A condition is given to control the loop (to continue running or terminate).

```
In [50]: basket = ['banana', 'apple', 'durian', 'orange', 'rambutan']
i=0
while i < 3:
    print(basket[i])
    i = i + 1  # this is the update to move the counter

banana
apple
durian</pre>
```

Let's now use a loop to count the number of items in the basket

```
# one way
In [56]:
         c = 0
         for fruits in basket:
            c = c + 1
        print(c)
         # another way
         for i, fruits in enumerate(basket):
            i = i + 1
         print(i)
        5
        5
        # write a for loop to sum all numbers in this list [3, 41, 12, 9, 74, 15]
In [57]:
         sum = 0
         for i in [3, 41, 12, 9, 74, 15]:
           sum = sum + i
         print(sum)
        154
         # Define a function called factorial(n)
In [97]:
         def factorial(n):
            f = 1
            while n > 1:  # keep multiplying while it has not reached 1
                f = f*n
                n = n-1
            return f
        print(factorial(10))
```

3628800

Quick Exercise 3 Write a loop that prints out all the even numbers between 0 and 99.

```
In [ ]: # Write your code here
```