```
1 i = 1
2 while i <= 4:
3       print(i)
4       i += 1
```

```
1
2
3
4
```

Python Loops

Python has two primitive loop commands:

- **while loops**
- **for loops**

The while Loop

With the while loop we can execute a set of statements as long as a condition is true.

Note: remember to increment i, or else the loop will continue forever.

The while loop requires relevant variables to be ready, in this example we need to define an indexing variable, i, which we set to 1.

Example

Print i as long as i is less than 6:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Explanation of the above code:

```
i = 1

# Create a variable called i and assign it the value of 1
# A variable is like a box that can store some data, such as a number, a word, or a list of things.
# You can give a variable any name you want, as long as it follows some rules, such as starting with a letter and not using spaces or symbols.
# The = sign means that you are putting the value on the right side into the box on the left side.
# In this case, we are creating a box named i and putting the number 1 inside it.
# We are using i as a counter or an iterator, which means that we will use it to keep track of how many times we loop.
# We are starting with 1 because we want to count from 1 to 5, but you can start with any number you want.

while i < 6:

# Start a loop that will repeat as long as i is less than 6
# A loop is a way to repeat some actions over and over again, without having to write the same code many times.
# The word while means that we are going to keep looping as long as the condition after it is true.
# A condition is a question that can be answered with yes or no, such as is i less than 6?
# The < sign means that we are comparing the values on both sides, and it returns yes if the left side is smaller than the right side, and no o
# The : sign means that we are ending the condition and starting the block of statements that we want to loop.
# A block of statements is a group of instructions that are indented under the same level, using spaces or tabs.
# The indentation tells the computer which statements belong to the loop and which ones do not.

 print(i)

  # Print the value of i on the screen
  # The word print is a function, which is a special kind of instruction that can do something for us.
  # A function has a name, followed by parentheses, and sometimes has some arguments inside the parentheses.
  # An argument is a piece of data that we give to the function to use.
  # In this case, the argument is i, which means that we are giving the value of i to the print function to show us.

   i += 1

  # Add 1 to the value of i and assign the result back to i
  # The += sign is a shortcut way of saying that we are taking the value of i, adding 1 to it, and then putting the result back into the box na
  # So, for example, if i was 1, then i += 1 would make i 2.
```

```
    # This is how we are making i increase by 1 every time we loop.

# End the loop and the program
# When the condition becomes false, such as when i is 6 or more, the loop stops and the program moves on to the next statement.
# In this case, there is no next statement, so the program ends.
```

Try it ourself below:

```
nessa = 0
while nessa < 6:
  print(nessa)
  nessa += 1
```

```
0
1
2
3
4
5
```

```
nessa = -6
while nessa < 6:
  print(nessa)
  nessa += 1
```

```
-6
-5
-4
-3
-2
-1
0
1
2
3
4
5
```

```
#what if i give an increemnt of 2
nessa = 10
while nessa < 6:
  print(nessa)
  nessa += 2
```

```
0
2
4
```

```
# what if the starting point is already greater than 6?
nessa = 15
while nessa > 6:
  print(nessa)
  if nessa == 8:
    break
  nessa -= 1
```

```
15
14
13
12
11
10
9
8
```

```
# what of the condition in while statement is never satisfied? LOOP OF DEATH
nessa = 7
while nessa > 6:
  print(nessa)
  nessa += 1
```

```
Streaming output truncated to the last 5000 lines.
85749
85750
85751
85752
85753
85754
85755
85756
85757
85758
85759
85760
85761
85762
85763
85764
85765
85766
85767
85768
85769
85770
85771
85772
85773
85774
85775
85776
85777
85778
85779
85780
85781
85782
85783
85784
85785
85786
85787
85788
85789
85790
85791
85792
85793
85794
85795
85796
85797
85798
85799
85800
85801
85802
85803
85804
85805
85806
85807
85808
85809
85810
85811
85812
85813
85814
85815
85816
85817
85818
85819
85820
85821
85822
85823
85824
85825
85826
85827
85828
85829
85830
85831
85832
85833
85834
85835
85836
85837
```

```
85838
85839
85840
85841
85842
85843
85844
85845
85846
85847
85848
85849
85850
85851
85852
85853
85854
85855
85856
85857
85858
85859
85860
85861
85862
85863
85864
85865
85866
85867
85868
85869
85870
85871
85872
85873
85874
85875
85876
85877
85878
85879
85880
85881
85882
85883
85884
85885
85886
85887
85888
85889
85890
85891
85892
85893
85894
85895
85896
85897
85898
85899
85900
85901
85902
85903
85904
85905
85906
85907
85908
85909
85910
85911
85912
85913
85914
85915
85916
85917
85918
85919
85920
85921
85922
85923
85924
85925
85926
85927
85928
```

```
85928
85929
85930
85931
85932
85933
85934
85935
85936
85937
85938
85939
85940
85941
85942
85943
85944
85945
85946
85947
85948
85949
85950
85951
85952
85953
85954
85955
85956
85957
85958
85959
85960
85961
85962
85963
85964
85965
85966
85967
85968
85969
85970
85971
85972
85973
85974
85975
85976
85977
85978
85979
85980
85981
85982
85983
85984
85985
85986
85987
85988
85989
85990
85991
85992
85993
85994
85995
85996
85997
85998
85999
86000
86001
86002
86003
86004
86005
86006
86007
86008
86009
86010
86011
86012
86013
86014
86015
86016
86017
86018
```

```
86019
86020
86021
86022
86023
86024
86025
86026
86027
86028
86029
86030
86031
86032
86033
86034
86035
86036
86037
86038
86039
86040
86041
86042
86043
86044
86045
86046
86047
86048
86049
86050
86051
86052
86053
86054
86055
86056
86057
86058
86059
86060
86061
86062
86063
86064
86065
86066
86067
86068
86069
86070
86071
86072
86073
86074
86075
86076
86077
86078
86079
86080
86081
86082
86083
86084
86085
86086
86087
86088
86089
86090
86091
86092
86093
86094
86095
86096
86097
86098
86099
86100
86101
86102
86103
86104
86105
86106
86107
86108
86109
```

```
86109
86110
86111
86112
86113
86114
86115
86116
86117
86118
86119
86120
86121
86122
86123
86124
86125
86126
86127
86128
86129
86130
86131
86132
86133
86134
86135
86136
86137
86138
86139
86140
86141
86142
86143
86144
86145
86146
86147
86148
86149
86150
86151
86152
86153
86154
86155
86156
86157
86158
86159
86160
86161
86162
86163
86164
86165
86166
86167
86168
86169
86170
86171
86172
86173
86174
86175
86176
86177
86178
86179
86180
86181
86182
86183
86184
86185
86186
86187
86188
86189
86190
86191
86192
86193
86194
86195
86196
86197
86198
86199
```

```
86200
86201
86202
86203
86204
86205
86206
86207
86208
86209
86210
86211
86212
86213
86214
86215
86216
86217
86218
86219
86220
86221
86222
86223
86224
86225
86226
86227
86228
86229
86230
86231
86232
86233
86234
86235
86236
86237
86238
86239
86240
86241
86242
86243
86244
86245
86246
86247
86248
86249
86250
86251
86252
86253
86254
86255
86256
86257
86258
86259
86260
86261
86262
86263
86264
86265
86266
86267
86268
86269
86270
86271
86272
86273
86274
86275
86276
86277
86278
86279
86280
86281
86282
86283
86284
86285
86286
86287
86288
86289
86290
```