REGULAR EXPRESSION

- **IF WE HAVE LOT OF DATA TO CHOOSE PRTICULAR DATA**
- **IT IS A SEARCH PATTERN**
- **TO CHECK PERTICULAR WORD IN STRING**

In [2]:

```
import re
print(dir(re))
```

```
['A', 'ASCII', 'DEBUG', 'DOTALL', 'I', 'IGNORECASE', 'L', 'LOCALE', 'M', 'MULTILINE', 'Ma
tch', 'Pattern', 'RegexFlag', 'S', 'Scanner', 'T', 'TEMPLATE', 'U', 'UNICODE', 'VERBOSE',
'X', '_MAXCACHE', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__load
er__', '__name__', '__package__', '__spec__', '__version__', '_cache', '_compile', '_comp
ile_repl', '_expand', '_locale', '_pickle', '_special_chars_map', '_subx', 'compile', 'co
pyreg', 'enum', 'error', 'escape', 'findall', 'finditer', 'fullmatch', 'functools', 'matc
h', 'purge', 'search', 'split', 'sre_compile', 'sre_parse', 'sub', 'subn', 'template']
```

In [7]:

```
name="harsha vardhan chekuri"
# to search chekuri in name
# name is name of the string
# if not there o/p is none
k=re.search("chekuri",name)
print(k)
```

```
<re.Match object; span=(15, 22), match='chekuri'>
```

In [9]:

```
frnd="joel sri prakash dasari"
a=re.findall("s",frnd)
print(a)
```

```
['s', 's', 's']
```

In [14]:

```
# sub()
# replace 1 string with another
# sub ("sri in i/p","sthri in o/p",stringname)
a=("sri manikanta raju")
b=re.sub("sri","sthri",a)
print(b)
```

```
sthri manikanta raju
```

In [18]:

```
a=("sri manikanta raju")
b=re.sub("raju","kajha",a)
print(b)
# not possible to change 2 diff words at a time
# can change a single letter also
```

```
sri manikanta kajha
```

In [22]:

```
a=("motu patlu pizza boys")
b=re.split("o",a)
print(b)
# "o" will not be there
```

```
['m', 'tu patlu pizza b', 'ys']
```

```
['m', 'tu patlu pizza b', 'ys']
```

In [25]:

```
a=("motu patlu pizza boys")
b=re.split(" ",a)
print (b)
# splitted where the space is given
```

```
['motu', 'patlu', 'pizza', 'boys']
```

In [33]:

```
a="harsha joel mani"
b=re.match("harsha",a)
print(b)
# only first word
```

```
<re.Match object; span=(0, 6), match='harsha'>
```

patterns

start ^

[a-z,0-9,._]set os chr

{9}

end----> $

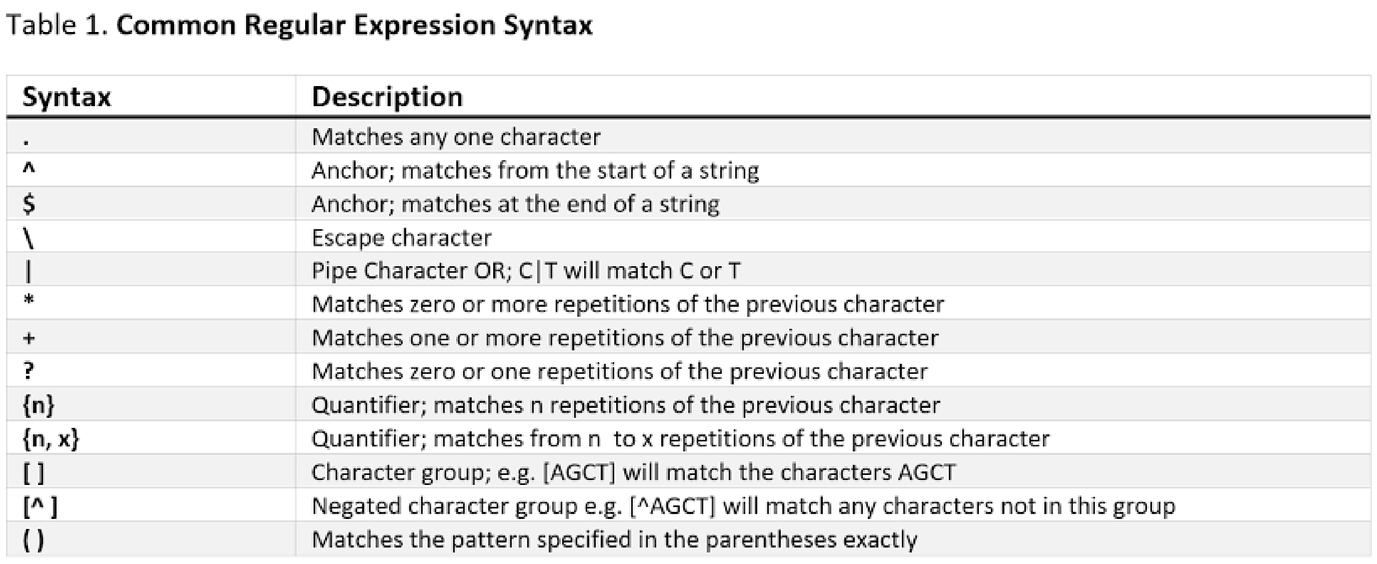

Table 1. **Common Regular Expression Syntax**

Syntax	Description
.	Matches any one character
^	Anchor; matches from the start of a string
$	Anchor; matches at the end of a string
\	Escape character
|	Pipe Character OR; C|T will match C or T
*	Matches zero or more repetitions of the previous character
+	Matches one or more repetitions of the previous character
?	Matches zero or one repetitions of the previous character
{n}	Quantifier; matches n repetitions of the previous character
{n, x}	Quantifier; matches from n to x repetitions of the previous character
[ ]	Character group; e.g. [AGCT] will match the characters AGCT
[^]	Negated character group e.g. [^AGCT] will match any characters not in this group
( )	Matches the pattern specified in the parentheses exactly

In [37]:

```
s="Goodmorning"
pt="^[A-Za-z]{3,11}$"
# given is 11 word string
# 3,11 means len is =>3 or <=11
if re.match(pt,s):
    print("True")
else:
    print("False")
```

```
True
```

In [40]:

```
#a="^[0-9]{10}$"(for any 10 digit )
```

```
#b="9490965774"# gives true
# to start with 6,7,8,9
a="^[6-9]{1}[0-9]{9}$"
b="7670819765"# if starting no is 5 false
if re.match(a,b):
    print("true")
else:
    print("false")
```

```
true
```

In [50]:

```
# phone no validation
a="^[6-9]{1}[0-9]{9}|^[+][9][1][6-9]{1}[0-9]{9}$"
b="+917670819765"# if starting no is 5 false
if re.match(a,b):
    print("true")
else:
    print("false")
```

```
true
```

In [52]:

```
a="^[6-9]{1}[0-9]{9}|^[+][9][1]$"
# "^" is used
b="917670819765"# if starting no is 5 false
if re.match(a,b):
    print("true")
else:
    print("false")
```

```
true
```

In [62]:

```
# email
a="^[a-zA-Z_.0-9]{4,42}[@][a-z]{3,8}[.][a-z]{2,9}$"
b="joeldasari10@gmail.com"
if re.match(a,b):
    print ("ok")
else :
    print ("no")
```

```
ok
```

NUMPY

- **NUM**
- **PY**
- **NUMERICAL PYTHON**
- **ITS IS CREATED IN 2005**
- **LARGE MODULE**
- **1 DIMENTIONAL**
- **2 DIMENTIONAL**

In [1]:

```
import numpy as np
print (dir(np))
```

```
['ALLOW_THREADS', 'AxisError', 'BUFSIZE', 'CLIP', 'ComplexWarning', 'DataSource', 'ERR_CA
LL', 'ERR_DEFAULT', 'ERR_IGNORE', 'ERR_LOG', 'ERR_PRINT', 'ERR_RAISE', 'ERR_WARN', 'FLOAT
ING_POINT_SUPPORT', 'FPE_DIVIDEBYZERO', 'FPE_INVALID', 'FPE_OVERFLOW', 'FPE_UNDERFLOW', '
False_', 'Inf', 'Infinity', 'MAXDIMS', 'MAY_SHARE_BOUNDS', 'MAY_SHARE_EXACT', 'MachAr', '
ModuleDeprecationWarning', 'NAN', 'NINF', 'NZERO', 'NaN', 'PINF', 'PZERO', 'RAISE', 'Rank
Warning', 'SHIFT_DIVIDEBYZERO', 'SHIFT_INVALID', 'SHIFT_OVERFLOW', 'SHIFT_UNDERFLOW', 'Sc
```

```
alarType', 'Tester', 'TooHardError', 'True_', 'UFUNC_BUFSIZE_DEFAULT', 'UFUNC_PYVALS_NAME
', 'VisibleDeprecationWarning', 'WRAP', '_NoValue', '_UFUNC_API', '__NUMPY_SETUP__', '__a
ll__', '__builtins__', '__cached__', '__config__', '__dir__', '__doc__', '__file__', '__g
etattr__', '__git_revision__', '__loader__', '__name__', '__package__', '__path__', '__sp
ec__', '__version__', '_add_newdoc_ufunc', '_distributor_init', '_globals', '_mat', '_pyt
esttester', 'abs', 'absolute', 'absolute_import', 'add', 'add_docstring', 'add_newdoc', '
add_newdoc_ufunc', 'alen', 'all', 'allclose', 'alltrue', 'amax', 'amin', 'angle', 'any',
 'append', 'apply_along_axis', 'apply_over_axes', 'arange', 'arccos', 'arccosh', 'arcsin',
 'arcsinh', 'arctan', 'arctan2', 'arctanh', 'argmax', 'argmin', 'argpartition', 'argsort',
 'argwhere', 'around', 'array', 'array2string', 'array_equal', 'array_equiv', 'array_repr'
, 'array_split', 'array_str', 'asanyarray', 'asarray', 'asarray_chkfinite', 'ascontiguous
array', 'asfarray', 'asfortranarray', 'asmatrix', 'asscalar', 'atleast_1d', 'atleast_2d',
 'atleast_3d', 'average', 'bartlett', 'base_repr', 'binary_repr', 'bincount', 'bitwise_and
', 'bitwise_not', 'bitwise_or', 'bitwise_xor', 'blackman', 'block', 'bmat', 'bool', 'bool
8', 'bool_', 'broadcast', 'broadcast_arrays', 'broadcast_to', 'busday_count', 'busday_off
set', 'busdaycalendar', 'byte', 'byte_bounds', 'bytes0', 'bytes_', 'c_', 'can_cast', 'cas
t', 'cbrt', 'cdouble', 'ceil', 'cfloat', 'char', 'character', 'chararray', 'choose', 'cli
p', 'clongdouble', 'clongfloat', 'column_stack', 'common_type', 'compare_chararrays', 'co
mpat', 'complex', 'complex128', 'complex256', 'complex64', 'complex_', 'complexfloating',
 'compress', 'concatenate', 'conj', 'conjugate', 'convolve', 'copy', 'copysign', 'copyto',
 'core', 'corrcoef', 'correlate', 'cos', 'cosh', 'count_nonzero', 'cov', 'cross', 'csingle
', 'ctypeslib', 'cumprod', 'cumproduct', 'cumsum', 'datetime64', 'datetime_as_string', 'd
atetime_data', 'deg2rad', 'degrees', 'delete', 'deprecate', 'deprecate_with_doc', 'diag',
 'diag_indices', 'diag_indices_from', 'diagflat', 'diagonal', 'diff', 'digitize', 'disp',
 'divide', 'division', 'divmod', 'dot', 'double', 'dsplit', 'dstack', 'dtype', 'e', 'ediff
1d', 'einsum', 'einsum_path', 'emath', 'empty', 'empty_like', 'equal', 'errstate', 'euler
_gamma', 'exp', 'exp2', 'expand_dims', 'expm1', 'extract', 'eye', 'fabs', 'fastCopyAndTra
nspose', 'fft', 'fill_diagonal', 'find_common_type', 'finfo', 'fix', 'flatiter', 'flatnon
zero', 'flexible', 'flip', 'fliplr', 'flipud', 'float', 'float128', 'float16', 'float32',
 'float64', 'float_', 'float_power', 'floating', 'floor', 'floor_divide', 'fmax', 'fmin',
 'fmod', 'format_float_positional', 'format_float_scientific', 'format_parser', 'frexp', '
frombuffer', 'fromfile', 'fromfunction', 'fromiter', 'frompyfunc', 'fromregex', 'fromstri
ng', 'full', 'full_like', 'fv', 'gcd', 'generic', 'genfromtxt', 'geomspace', 'get_array_w
rap', 'get_include', 'get_printoptions', 'getbufsize', 'geterr', 'geterrcall', 'geterrobj
', 'gradient', 'greater', 'greater_equal', 'half', 'hamming', 'hanning', 'heaviside', 'hi
stogram', 'histogram2d', 'histogram_bin_edges', 'histogramdd', 'hsplit', 'hstack', 'hypot
', 'i0', 'identity', 'iinfo', 'imag', 'in1d', 'index_exp', 'indices', 'inexact', 'inf', '
info', 'infty', 'inner', 'insert', 'int', 'int0', 'int16', 'int32', 'int64', 'int8', 'int
_', 'int_asbuffer', 'intc', 'integer', 'interp', 'intersect1d', 'intp', 'invert', 'ipmt',
 'irr', 'is_busday', 'isclose', 'iscomplex', 'iscomplexobj', 'isfinite', 'isfortran', 'isi
n', 'isinf', 'isnan', 'isnat', 'isneginf', 'isposinf', 'isreal', 'isrealobj', 'isscalar',
 'issctype', 'issubclass_', 'issubdtype', 'issubsctype', 'iterable', 'ix_', 'kaiser', 'kro
n', 'lcm', 'ldexp', 'left_shift', 'less', 'less_equal', 'lexsort', 'lib', 'linalg', 'lins
pace', 'little_endian', 'load', 'loads', 'loadtxt', 'log', 'log10', 'log1p', 'log2', 'log
addexp', 'logaddexp2', 'logical_and', 'logical_not', 'logical_or', 'logical_xor', 'logspa
ce', 'long', 'longcomplex', 'longdouble', 'longfloat', 'longlong', 'lookfor', 'ma', 'mafr
omtxt', 'mask_indices', 'mat', 'math', 'matmul', 'matrix', 'matrixlib', 'max', 'maximum',
 'maximum_sctype', 'may_share_memory', 'mean', 'median', 'memmap', 'meshgrid', 'mgrid', 'm
in', 'min_scalar_type', 'minimum', 'mintypecode', 'mirr', 'mod', 'modf', 'moveaxis', 'mso
rt', 'multiply', 'nan', 'nan_to_num', 'nanargmax', 'nanargmin', 'nancumprod', 'nancumsum'
, 'nanmax', 'nanmean', 'nanmedian', 'nanmin', 'nanpercentile', 'nanprod', 'nanquantile',
 'nanstd', 'nansum', 'nanvar', 'nbytes', 'ndarray', 'ndenumerate', 'ndfromtxt', 'ndim', 'n
dindex', 'nditer', 'negative', 'nested_iters', 'newaxis', 'nextafter', 'nonzero', 'not_eq
ual', 'nper', 'npv', 'numarray', 'number', 'obj2sctype', 'object', 'object0', 'object_',
 'ogrid', 'oldnumeric', 'ones', 'ones_like', 'outer', 'packbits', 'pad', 'partition', 'per
centile', 'pi', 'piecewise', 'place', 'pmt', 'poly', 'poly1d', 'polyadd', 'polyder', 'pol
ydiv', 'polyfit', 'polyint', 'polymul', 'polynomial', 'polysub', 'polyval', 'positive', '
power', 'ppmt', 'print_function', 'printoptions', 'prod', 'product', 'promote_types', 'pt
p', 'put', 'put_along_axis', 'putmask', 'pv', 'quantile', 'r_', 'rad2deg', 'radians', 'ra
ndom', 'rate', 'ravel', 'ravel_multi_index', 'real', 'real_if_close', 'rec', 'recarray',
 'recfromcsv', 'recfromtxt', 'reciprocal', 'record', 'remainder', 'repeat', 'require', 're
shape', 'resize', 'result_type', 'right_shift', 'rint', 'roll', 'rollaxis', 'roots', 'rot
90', 'round', 'round_', 'row_stack', 's_', 'safe_eval', 'save', 'savetxt', 'savez', 'save
z_compressed', 'sctype2char', 'sctypeDict', 'sctypeNA', 'sctypes', 'searchsorted', 'selec
t', 'set_numeric_ops', 'set_printoptions', 'set_string_function', 'setbufsize', 'setdiff1
d', 'seterr', 'seterrcall', 'seterrobj', 'setxor1d', 'shape', 'shares_memory', 'short', '
show_config', 'sign', 'signbit', 'signedinteger', 'sin', 'sinc', 'single', 'singlecomplex
', 'sinh', 'size', 'sometrue', 'sort', 'sort_complex', 'source', 'spacing', 'split', 'sqr
t', 'square', 'squeeze', 'stack', 'std', 'str', 'str0', 'str_', 'string_', 'subtract', 's
um', 'swapaxes', 'sys', 'take', 'take_along_axis', 'tan', 'tanh', 'tensordot', 'test', 't
esting', 'tile', 'timedelta64', 'trace', 'tracemalloc_domain', 'transpose', 'trapz', 'tri
', 'tril', 'tril_indices', 'tril_indices_from', 'trim_zeros', 'triu', 'triu_indices', 'tr
```

```
iu_indices_from', 'true_divide', 'trunc', 'typeDict', 'typeNA', 'typecodes', 'typename',
'ubyte', 'ufunc', 'uint', 'uint0', 'uint16', 'uint32', 'uint64', 'uint8', 'uintc', 'uintp
', 'ulonglong', 'unicode', 'unicode_', 'union1d', 'unique', 'unpackbits', 'unravel_index'
, 'unsignedinteger', 'unwrap', 'ushort', 'vander', 'var', 'vdot', 'vectorize', 'version',
'void', 'void0', 'vsplit', 'vstack', 'warnings', 'where', 'who', 'zeros', 'zeros_like']
```

In [9]:

```
k=np.arange(10)
print(k)
print(type(k))
```

```
[0 1 2 3 4 5 6 7 8 9]
<class 'numpy.ndarray'>
```

In [10]:

```
k=np.arange(0,20,2)
print(k)
print(type(k))
```

```
[ 0  2  4  6  8 10 12 14 16 18]
<class 'numpy.ndarray'>
```

In [12]:

```
a=np.array([0,1,2,3,4,5,6,7,8,9])
print(a)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

In [15]:

```
a=np.identity(5)# identity matrix
print(a)
print(type(a))
```

```
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
<class 'numpy.ndarray'>
```

In [17]:

```
a=np.arange(100)
print(a)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
 96 97 98 99]
```

In [20]:

```
a=np.arange(100)
print(a.min())
print(a.max())
print(a.std())
print(a.mean())
```

```
0
99
28.86607004772212
49.5
```

In [24]:

```
a=np.arange(100)
```

```
print(a+2)
```

```
[  2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37
  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55
  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73
  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91
  92  93  94  95  96  97  98  99 100 101]
```

In [26]:

```
a=np.arange(100)
print(a-2)
```

```
[-2 -1  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69
 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93
 94 95 96 97]
```

In [28]:

```
a=np.arange(100)
print(a*2)
```

```
[  0   2   4   6   8  10  12  14  16  18  20  22  24  26  28  30  32  34
  36  38  40  42  44  46  48  50  52  54  56  58  60  62  64  66  68  70
  72  74  76  78  80  82  84  86  88  90  92  94  96  98 100 102 104 106
 108 110 112 114 116 118 120 122 124 126 128 130 132 134 136 138 140 142
 144 146 148 150 152 154 156 158 160 162 164 166 168 170 172 174 176 178
 180 182 184 186 188 190 192 194 196 198]
```

In [47]:

```
a=np.arange(100)
print(a/2)
```

```
[ 0.   0.5  1.   1.5  2.   2.5  3.   3.5  4.   4.5  5.   5.5  6.   6.5
  7.   7.5  8.   8.5  9.   9.5 10.  10.5 11.  11.5 12.  12.5 13.  13.5
 14.  14.5 15.  15.5 16.  16.5 17.  17.5 18.  18.5 19.  19.5 20.  20.5
 21.  21.5 22.  22.5 23.  23.5 24.  24.5 25.  25.5 26.  26.5 27.  27.5
 28.  28.5 29.  29.5 30.  30.5 31.  31.5 32.  32.5 33.  33.5 34.  34.5
 35.  35.5 36.  36.5 37.  37.5 38.  38.5 39.  39.5 40.  40.5 41.  41.5
 42.  42.5 43.  43.5 44.  44.5 45.  45.5 46.  46.5 47.  47.5 48.  48.5
 49.  49.5]
```

In [50]:

```
a=np.zeros(10)
print(a)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

In [52]:

```
a=np.ones(10)
print(a)
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

In [55]:

```
a=np.full_like(1,4)
print(a)
```

```
4
```

In [56]:

```
a=np.full_like(np.arange(10),4)
print(a)
```

```
[4 4 4 4 4 4 4 4 4 4]
```

In [58]:

```
a=np.arange(10)
b=np.full_like(a,2)
print(b)
```

```
[2 2 2 2 2 2 2 2 2 2]
```

In [60]:

```
a=np.full([2,3],(6))
print(a)
```

```
[[6 6 6]
 [6 6 6]]
```

In [63]:

```
nsize=np.arange(100).resize(50,2)
print(nsize)
```

```
None
```

In [67]:

```
nsize=np.arange(100).reshape(50,2)# rows & coloumns
print(nsize)
```

```
[[ 0  1]
 [ 2  3]
 [ 4  5]
 [ 6  7]
 [ 8  9]
 [10 11]
 [12 13]
 [14 15]
 [16 17]
 [18 19]
 [20 21]
 [22 23]
 [24 25]
 [26 27]
 [28 29]
 [30 31]
 [32 33]
 [34 35]
 [36 37]
 [38 39]
 [40 41]
 [42 43]
 [44 45]
 [46 47]
 [48 49]
 [50 51]
 [52 53]
 [54 55]
 [56 57]
 [58 59]
 [60 61]
 [62 63]
 [64 65]
 [66 67]
 [68 69]
 [70 71]
 [72 73]
 [74 75]
 [76 77]
 [78 79]
 [80 81]
 [82 83]
 [84 85]
 [86 87]
 [88 89]
 [90 91]
 [92 93]
 [94 95]
 [96 97]
 [98 99]]
```

In [66]:

```
nsize=np.arange(100).reshape(10,10)
print(nsize)
```

```
[[ 0  1  2  3  4  5  6  7  8  9]
 [10 11 12 13 14 15 16 17 18 19]
 [20 21 22 23 24 25 26 27 28 29]
 [30 31 32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47 48 49]
 [50 51 52 53 54 55 56 57 58 59]
 [60 61 62 63 64 65 66 67 68 69]
 [70 71 72 73 74 75 76 77 78 79]
 [80 81 82 83 84 85 86 87 88 89]
 [90 91 92 93 94 95 96 97 98 99]]
```