Q1. Write a for loop to print numbers 1 through 100 in ten rows.

```
In [3]:
```

```
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 99 100
```

Q2. Write a for loop to print first 100 even numbers. Make sure the output is in 10 rows and format the output so that all columns align perfectly.

```
In [13]:
```

```
  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
182 184 186 188 190 192 194 196 198 200
```

Q3. Write a function isPrime(n), that returns a True if n is a prime number, if not False.

```
In [5]:
```

```
In [33]: isPrime(11)
```

```
Out[33]: True
```

```
In [7]: isPrime(10)
```

```
Out[7]: False
```

Q4. Write a for loop to print Prime numbers between 1 and 1000. Format the output so that all columns align perfectly.

In [45]:

```
  2   3   5   7  11  13  17  19  23  29
 31  37  41  43  47  53  59  61  67  71
 73  79  83  89  97 101 103 107 109 113
127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229
233 239 241 251 257 263 269 271 277 281
283 293 307 311 313 317 331 337 347 349
353 359 367 373 379 383 389 397 401 409
419 421 431 433 439 443 449 457 461 463
467 479 487 491 499 503 509 521 523 541
547 557 563 569 571 577 587 593 599 601
607 613 617 619 631 641 643 647 653 659
661 673 677 683 691 701 709 719 727 733
739 743 751 757 761 769 773 787 797 809
811 821 823 827 829 839 853 857 859 863
877 881 883 887 907 911 919 929 937 941
947 953 967 971 977 983 991 997
```

Q5. Write a function to convert temperature in celsius to fahrenheit

Conversion formula: fahrenheit = (celsius * 1.8) + 32

In [47]:

In [61]:
```
cel2fah(0)
```

Out[61]: 32.0

In [62]:
```
cel2fah(20)
```

Out[62]: 68.0

Q6. Write a function to convert temperature in fahrenheit to celsius (Q5 Reverse)

Conversion formula: Think !

In [55]:

In [56]:
```
fah2cel(68)
```

Out[56]: 20.0

In [63]:
```
fah2cel(32)
```

Out[63]:
```
0.0
```

In [64]:
```
fah2cel(212)
```

Out[64]:
```
100.0
```

Q7. Convert the following list of Kilometers to miles.

```
[ 10, 20, 30, 40, 50 ]
Conversion: 1 Km = 0.621371 of a Mile.
```

In [70]:
```

10 kilometers is equal to 6.21371 miles
20 kilometers is equal to 12.42742 miles
30 kilometers is equal to 18.64113 miles
40 kilometers is equal to 24.85484 miles
50 kilometers is equal to 31.068550000000002 miles
```

Q8. Write factorial function fact(n) without using recurrsion.

In [84]:
```

```

In [90]:
```
fact(5)
```

Out[90]:
```
120
```

Q9. Write a factorial function factR(n) using recurrsion.

In [95]:
```

```

In [105]:
```
factR(5)
```

Out[105]:
```
120
```

Q10. Write a function toBinary(n) to convert a decimal number into binary number using recurrsion.

In [103]:
```

```

In [106]:
```
toBinary(100)
```

```
1100100
```

Q11. Write your own mymap() function which works exactly like Python's built-in function map().

```
 mymap(fn, C) -- Takes in 2 parameters fn : a function and C: a contain
er object.
 It applies the given function fn on each item in the container and pro
duces a new
 container.
```

In [1]:
```
```

In [2]:
```
A = [ 1, 2, 3, 4 ]
mymap(lambda x:x+2, A)
```

Out[2]: [3, 4, 5, 6]

In [3]:
```
A = [ 1, 2, 3, 4 ]
mymap(lambda x:(x,x**2), A)
```

Out[3]: [(1, 1), (2, 4), (3, 9), (4, 16)]

Q12. Write your own myreduce() function which works exactly like Python's built-in function reduce().

```
 myreduce(fn, C) -- Takes in 2 parameters fn : a function and C: a cont
ainer object.
 It applies the given function 'fn' continually to sequence to produce
a single value.
```

In [4]:
```
```

In [5]:
```
A = [ 1, 2, 33, 22, 11, 5]
myreduce(lambda x,y: x if x>y else y, A)
```

Out[5]: 33

In [6]:
```
A = [ 10, 2, 33, 22, 11, 5]
myreduce(lambda x,y:x+y, A)
```

Out[6]: 83

Q13. Write your own myfilter() function which works exactly like Python's built-in function filter().

```
 myfilter(fn, C) -- Takes in 2 parameters fn : a function and C: a cont
ainer object.
 Filters out all items of the container object for which function 'fn'
returns a True.
```

In [8]:
```
```

In [9]:
```
A = [ 10, 2, 33, 22, 11, 5]
myfilter(lambda x: x%2==0, A)
```

Out[9]: [10, 2, 22]

In [10]:
```
A = [ 10, 2, 33, 22, 11, 5]
myfilter(lambda x: x>10, A)
```

Out[10]: [33, 22, 11]

Q14. Write list comprehensions to produce the following three simple lists

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
[1, 6, 11, 16, 21, 26, 31, 36, 41, 46]
```

In [11]:
```
```

Out[11]: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

In [13]:
```
```

Out[13]: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

In [15]:
```
```

Out[15]: [1, 6, 11, 16, 21, 26, 31, 36, 41, 46]

Q15. Write List comprehensions to produce the following Lists

['P', 'y', 't', 'h', 'o', 'n']

['x', 'xx', 'xxx', 'xxxx', 'y', 'yy', 'yyy', 'yyyy', 'z', 'zz', 'zzz', 'zzzz']

['x', 'y', 'z', 'xx', 'yy', 'zz', 'xx', 'yy', 'zz', 'xxxx', 'yyyy', 'zzzz']

[[2], [3], [4], [3], [4], [5], [4], [5], [6]]

[[2, 3, 4, 5], [3, 4, 5, 6], [4, 5, 6, 7], [5, 6, 7, 8]]

[(1, 1), (2, 1), (3, 1), (1, 2), (2, 2), (3, 2), (1, 3), (2, 3), (3, 3)]

```
In [1]:
```

```
Out[1]: ['P', 'y', 't', 'h', 'o', 'n']
```

```
In [9]:
```

```
Out[9]: ['x', 'xx', 'xxx', 'xxxx', 'y', 'yy', 'yyy', 'yyyy', 'z', 'zz', 'z
        zz', 'zzzz']
```

```
In [10]:
```

```
Out[10]: ['x', 'y', 'z', 'xx', 'yy', 'zz', 'xx', 'yy', 'zz', 'xxxx', 'yyy
         y', 'zzzz']
```

```
In [11]:
```

```
Out[11]: [[2], [3], [4], [3], [4], [5], [4], [5], [6]]
```

```
In [13]:
```

```
Out[13]: [[2, 3, 4, 5], [3, 4, 5, 6], [4, 5, 6, 7], [5, 6, 7, 8]]
```

```
In [14]:
```

```
Out[14]: [(1, 1), (2, 1), (3, 1), (1, 2), (2, 2), (3, 2), (1, 3), (2, 3),
         (3, 3)]
```

Q16. Write Dictionary comprehensions to produce the following Dictionaries:

```
In [9]:
```

```
Out[9]: {1: 1, 2: 8, 3: 27, 4: 64, 5: 125, 6: 216, 7: 343, 8: 512, 9: 729}
```

In [49]:

```
```

Out[49]:
```
{'a': 'a', 'b': 'bb', 'c': 'ccc', 'd': 'dddd', 'e': 'eeeee', 'f':
'ffffff'}
```

Q17. Write a function longestWord() that takes a list of words and returns the longest one.

In [54]:

```
```

In [55]:

```
longestWord(['Apple', 'Apricot', 'Asparagus', 'Avocado'])
```

```
Asparagus
```

Q18. A palindrome is a word, phrase, number, or other sequence of characters which reads the same backward or forward. Allowances may be made for adjustments to capital letters, punctuation, and word dividers.

In [50]:

```
```

In [51]:

```
palindrome("noon")
palindrome("civic")
palindrome("racecar")
palindrome("kayak")
```

```
noon: is a Palindrome.
civic: is a Palindrome.
racecar: is a Palindrome.
kayak: is a Palindrome.
```

In [52]:

```
palindrome("stack cats")
palindrome("step on no pets")
palindrome( "Was it a car or a cat I saw?")
palindrome("A man, a plan, a canal, Panama!")
```

```
stack cats: is a Palindrome.
step on no pets: is a Palindrome.
Was it a car or a cat I saw?: is a Palindrome.
A man, a plan, a canal, Panama!: is a Palindrome.
```

19. A pangram is a sentence that contains all the letters of the English alphabetat least once. example: "The quick brown fox jumps over the lazy dog".

Write a function to check if a sentence is a pangram or not.

In [67]:

```
```

In [68]:
```
isPangram( "The quick brown fox jumps over the lazy dog")
```

```
Yes, it is a Panagram sentence.
```

20. Write a function charFreq() that takes a string and creates a frequency listing of the characters contained in it using a dictionary object.

charFreq("abcabcxyzxyzkkkmmmnnnsssqqqqkkaabbcc")

In [78]:
```

```

In [79]:
```
charFreq("abcabcxyzxyzkkkmmmnnnsssqqqqkkaabbcc")
```

```
{'z': 2, 'c': 4, 'n': 3, 'a': 4, 'k': 5, 's': 3, 'q': 4, 'b': 4,
'm': 3, 'y': 2, 'x': 2}
```

21. Using random module, generate 10,000 numbers between 1 and 5. Find the number for which largest numbers generated. Also create a dictionary with key-values where value > 2000, using dictionary comprehensions.

import random

Use the method: random.randint(1, 5)

In [110]:
```

```

```
{1: 1968, 2: 1934, 3: 2047, 4: 2087, 5: 1964}
Maximum generated number: 4 ---> 2087 times.
```

Out[110]:
```
{3: 2047, 4: 2087}
```

In [ ]:
```

```