Python

Strings

What is a string

String, text, characters, non numeric characters

Note: anything on the keyboard is text (string) including numbers

Indexing

Indexing of a string is the same as arrays and lists. First character is at position zero (0).

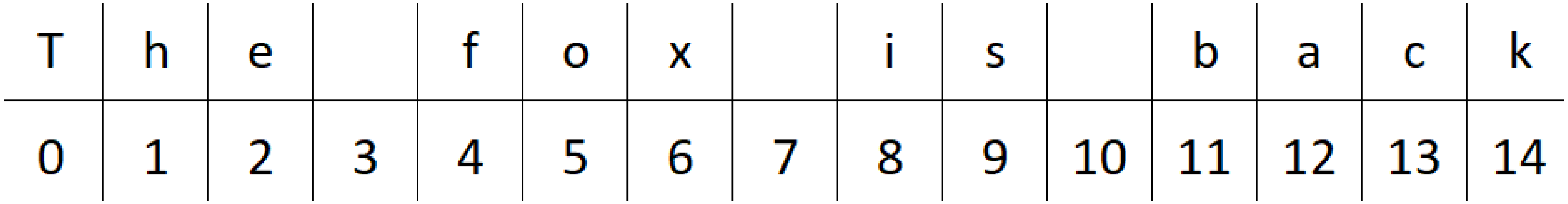
Index number must be an integer and positive.

Attempting to use an index number that is float or number outside length of sting WILL cause an error.

Negative will be treated as a positive number

(absolute value)

Indexing



T	h	e		f	o	x		i	s		b	a	c	k
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Example:

```
str01 = 'The fox is back'
str02 = str01[4]
print(str02)
```

String Iteration

Python can process individual characters in each string

This is done with a **for** loop.

Also, a good use of the break statement

String Iteration

Using a for loop, look at individual characters of a string

Example:

```
str01 = ‘wright’
for ch in str01:
     print(ch)
```

TRY THIS

```
print('\n'*5)
str01 = 'wright'
for ch in str01:
    print(ch)
ans = input('hit enter to continue')
print('-'*20)
str02 = input('enter text, word or something: ')
for ch in str02:
    print(ch)
print('-'*20)
```

stl00

Indexing error

If index is in error (negative, out of bounds)

Then an IndexError is issued

```
str01 = '01/31/2020'
str02 = str01[15]
```

Index beyond end of string

Repetition of characters

Repeat characters without a lot of typing

Example:

```
print('----------')
print
print('-' * 10)
```

Note: asterisk

stl01

Find characters or multi characters in a string

Using an if statement

use: in

results: TRUE is the string

FALSE not in string

use: not in

results: TRUE not in string

FALSE found in string

Format

str01 - characters looking for

str99 - string of characters

```
if str01 in str99:
      statements  (true  -  found)
else:
      statements (false – not found)

If str01 not in str99:
     statements  (true  -  not found)
else:
      statements (false –  found)
```

Using in/not in with if & while

```
str01 = 'The fox is back'
str02 = 'is'
if str02 in str01:
    print('True')
else:
    print('False')
```

Test: change in to not in, change to in and str02 to ‘one’

stl02

Built in functions

String handling

Length - strings only

Form: var1 = len(var2)

Return length of data or zero

Can be used in condition

```
 if (len(var) == 0):
  or
  lenname = len(lname)
  if (lenname == 0):
```

Length test

str22 =“”

str44=“ “

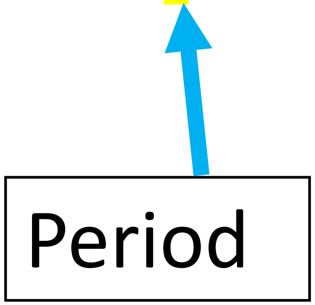
What is the length of str22?

What is the length of str44?

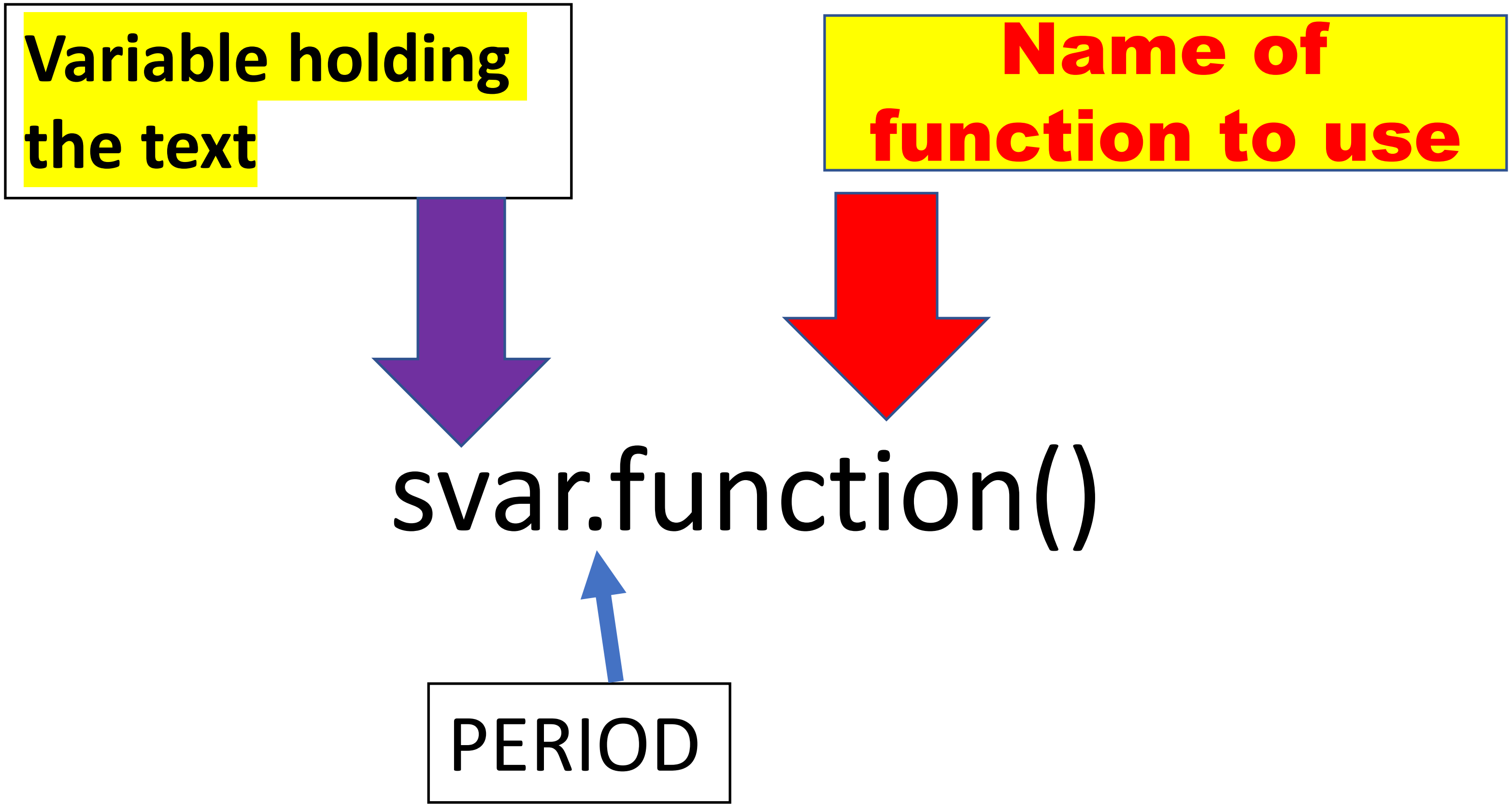
String functions

The variable that is being used as a string, is attached to the function that will manipulate the content of the string variable.

This is done by variable.function.



String functions structure



Variable holding the text

Name of function to use

svar.function()

PERIOD

lower case

lower() - make string all lower case

```
str55=‘CASE’
str88=str55.lower()
print(str88)
```

Results: case

Upper case

upper() - make string all upper case

```
str22=‘down’
str11=str55.upper()
print(str11)
```

Results: DOWN

caps

capitalize() - make first character of string upper case

```
str55=‘middle’
str77=str55. capitalize()
print(str77)
```

Results: Middle

```
def main():
    print('\n'*50)
    str01 = ' '
    str01=input('enter a name-> ')
    strlen1 = len(str01)
    print(' Len of: ',str01,' is ',strlen1)
    str02 = str01.upper()
    print(' All upper case is -> ',str02)
    str02 = str01.lower()
    print(' All lower case is -> ',str02)
    str02 = str01.capitalize()
    print(' Capital is -> ',str02)
main()
```

Try This

stl03

String handling

Count

count() - Count the number of characters in a string.

```
str11 = “ a little string”
nb1 = str11.count(‘t’)
print(nb1)
    result: 3
```

find

find() - Find a character or group of characters in a string.

```
str22 = “a little string”
nb2 = str22.find(‘tt’)
print(nb2)
    result: 4
```

replace

replace(‘old’,’new’) - Replace one set of characters with anther set of characters.

```
str33 = “ a little string”
nb3 = str33.replace(‘tt’,’mm’)
print(nb3)
    result: a limmle string
```

Try This

```
def main():
    print('\n'*50)
    str01 = ' '
    str01=input('enter a sentence -> ')
    numb1 = str01.count('t')
    print('Sentence is: ', str01)
    print("Number is t's is: ",numb1)
    numb2 =  str01.find('tt')
    print(' Position of tt is -> ',numb2)
    str99 = 'tt'
    numb2 =  str01.find(str99)
    print(' Position of first g is -> ',numb2)
    str02 = str01.replace('is','was')
    print('new sentence is -> ',str02)
main()
```

stl04

String slicing 2 ways

String Iteration one way

Using a for loop, look at individual characters of a string

Example:

```
str01 = ‘wright’
for ch in str01:
     print(ch)
```

String slicing another way

NOTE: brackets

```
string[start:end]
```

start & end:

- must be a positive integer
- start must be less than or equal to end

TRY THIS

```
def main():
    str01 = 'The fox is back’
    sn1 =   '123456789012345’
    sn2 =   '         111111’
    print(str01)
    print(sn1)
    print(sn2)
    str02 = str01[4:7]
    print(str02)
    str03 = str01[:3]
    print(str03)
    str04 = str01[11:]
    print(str04)
    return
main()
```

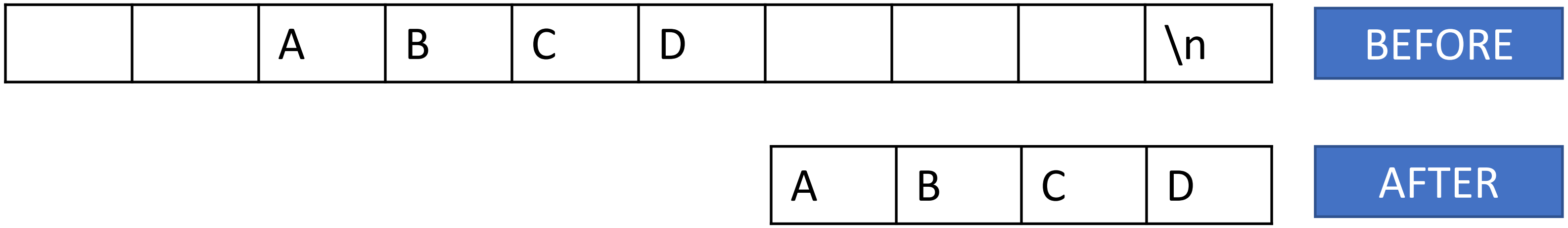
stl05

Remove spaces from strings

Remove space from string

It is a good idea to remove spaces from the beginning and from the end of a string. The results is to save memory and file space.

Also, line feed character ('\n') is treated as a space.



		A	B	C	D				\n	BEFORE
						A	B	C	D	AFTER

Remove spaces left side

lstrip() – remove spaces from the left of the string.

str15 =‘ ABCD   ‘

		A	B	C	D				\n

BEFORE

str25= str15.lstrip()

A	B	C	D				\n

AFTER

Remove spaces - right side

rstrip() – remove spaces from the right of the string.

str16 =‘ ABCD   ‘

		A	B	C	D				\n

BEFORE

str26= str16.rstrip()

		A	B	C	D

AFTER

Remove spaces - both sides

strip() – remove spaces from both the right and left of the string.

```
str19 =‘  ABCD   ‘
```

		A	B	C	D				\n

BEFORE

```
str29 = str19.strip()
```

A	B	C	D

AFTER

Strip - special

You can remove any character from the beginning or end of the string.

Characters like: $ # etc.

Remove specific characters

ch = character to be removed

lstrip(ch) – remove spaces from the left of the string.

rstrip(ch) – remove spaces from the right of the string.

strip(ch) – remove spaces from both the right and left of the string.

Try This

```
def main():
    print('\n'*50)
    str01 = ' '
    str01=input('enter characters -> ‘)
    str02 = str01.lstrip()
    print(' length left is: ',len(str01), ' after ', len(str02))
    str02 = str01.rstrip()
    print(' length right is: ',len(str01), ' after ', len(str02))
    str02 = str01.strip()
    print(' length both is: ',len(str01), ' after ', len(str02))
main()
```

Check for numbers

Numeric

isnumeric()

check if string is numbers.

```
str01.isnumeric()
```

check for 0,1,2,3,4,5,6,7,8,9

isnumeric

1234	True
12ab34	FALSE
12.34	FALSE
1,234	FALSE
12 23	FALSE
.1234	FALSE
9876	TRUE

Alpha checking

isalpha() – string contains alpha characters.

Valid characters:

A-Z, a-z

isalpha

ZXCD	TRUE
ZX CD	FALSE
AB23CD	FALSE
AB,CD	FALSE
ZX$ER	FALSE
abcd	TRUE

Alpha/numeric checking

isalnum() – string contains numbers and letters.

Valid characters:

A-Z, a-z, 0-9

isalnum

ABCD	TRUE
1234	TRUE
AB12CD	TRUE
AB,12	FALSE
12 34	FALSE
12.34	FALSE
ab$er	FALSE
tree	TRUE

Usually used with an if statement

Format:

```
str67 = ‘ ab99cc’
if str67.isalnum():
      true
else:
      false
```

Format:

```
str87 = ‘ ab99cc’
if str87.isalpha():
      true
else:
      false
```

Example test

```
def chkalnum(ss):
    if (ss.isalnum()):
        print(ss, ' - YES alphanumeric')
    else:
        print(ss, ' - not alphanumeric')
    return
def chkalpha(ss):
    if (ss.isalpha()):
        print(ss, ' - YES alphabetic')
    else:
        print(ss, ' - not pure alphabetic')
    return
```

Example test

```
def chknum(ss):
    if (ss.isnumeric()):
        print(ss, ' - YES numeric')
    else:
        print(ss, ' - not numeric')
    return
def chkdigit(ss):
    if (ss.isdigit()):
        print(ss, ' - YES digits')
    else:
        print(ss, ' - not digits')
    return
def chkdecimal(ss):
    if (ss.isdecimal()):
        print(ss, ' - YES decimal number')
    else:
        print(ss, ' - not decimal')
    return
```

Example test

```
def main():
    print('\n'*50)
    str01 = ' '
    ans = 'y'
    while ans.upper() == 'Y':
        str01=input('enter characters -> ')
        chkalnum(str01)
        chkalpha(str01)
        chknum(str01)
        chkdigit(str01)
        chkdecimal(str01)
        ans = input('Another test (y/n):')
main()
```

Test:
Upper / lower case
/ spaces

Upper / Lower / spaces

islower() – test if string is all lower case.

```
str03 = ‘case’
if str03.islower():
    true
else
    false
```

Upper / Lower / spaces

isupper() – test if string is all upper case.

```
str04 = ‘HIGH’
 if str04.isupper():
       true
 else
       false
```

isspace

isspace() – check for spaces

```
 str01.isspace()
```

Check for: spaces (blanks)

Upper / Lower / spaces

isspace() – test if string is spaces or empty.

```
str03 = ‘      ’
 if str03.isspace():
        true
 else
        false
```

Try This Part 1

```
def chkupper(ss):
    if (ss.isupper()):
        rr ='yes'
    else:
        rr = 'no'
    return rr
def chklower(ss):
    if (ss.islower()):
        rr = 'yes'
    else:
        rr = 'no'
    return rr
```

```
def chkspace(ss):
    if (ss.isspace()):
        rr = 'yes'
    else:
        rr = 'no'
    return rr
```

Try This Part 2

```
def main():
    ans = 'y'
    while (ans.upper() =='Y'):
        print('\n'*5)
        str01 = ' '
        str01=input('enter characters -> ')
        aa=chkupper(str01)
        print('input upper case -> ',aa)
        bb=chklower(str01)
        print('input lower case ->',bb)
        cc=chkspace(str01)
        print('input is blank ->',cc)
        ans = input('\ncontinue y/n -> ')
    print('\n ---done---')
main()
```

TIP

One use of upper / lower

```
while (ans == ‘Y’) or (ans == ‘y’):
```

Replace with

```
while (ans.upper() == ‘Y’):
          or
while (ans.lower() == ‘y’):
```

Shell program

```
import datetime
def somefunc():
    print('\n***** function does something')
    return
def pgm():
    print('program start ',datetime.datetime.now())
    ans = 'y'
    while (ans.upper() == 'Y'):
        somefunc()
        ans = input('\nAgain? enter y=yes, n==no :')
    print('program done ',datetime.datetime.now())
    return
pgm()
```

Concatenation

Joining string together

Use the **+** to put the string together

Example: Fname + ‘ ‘+ Lname

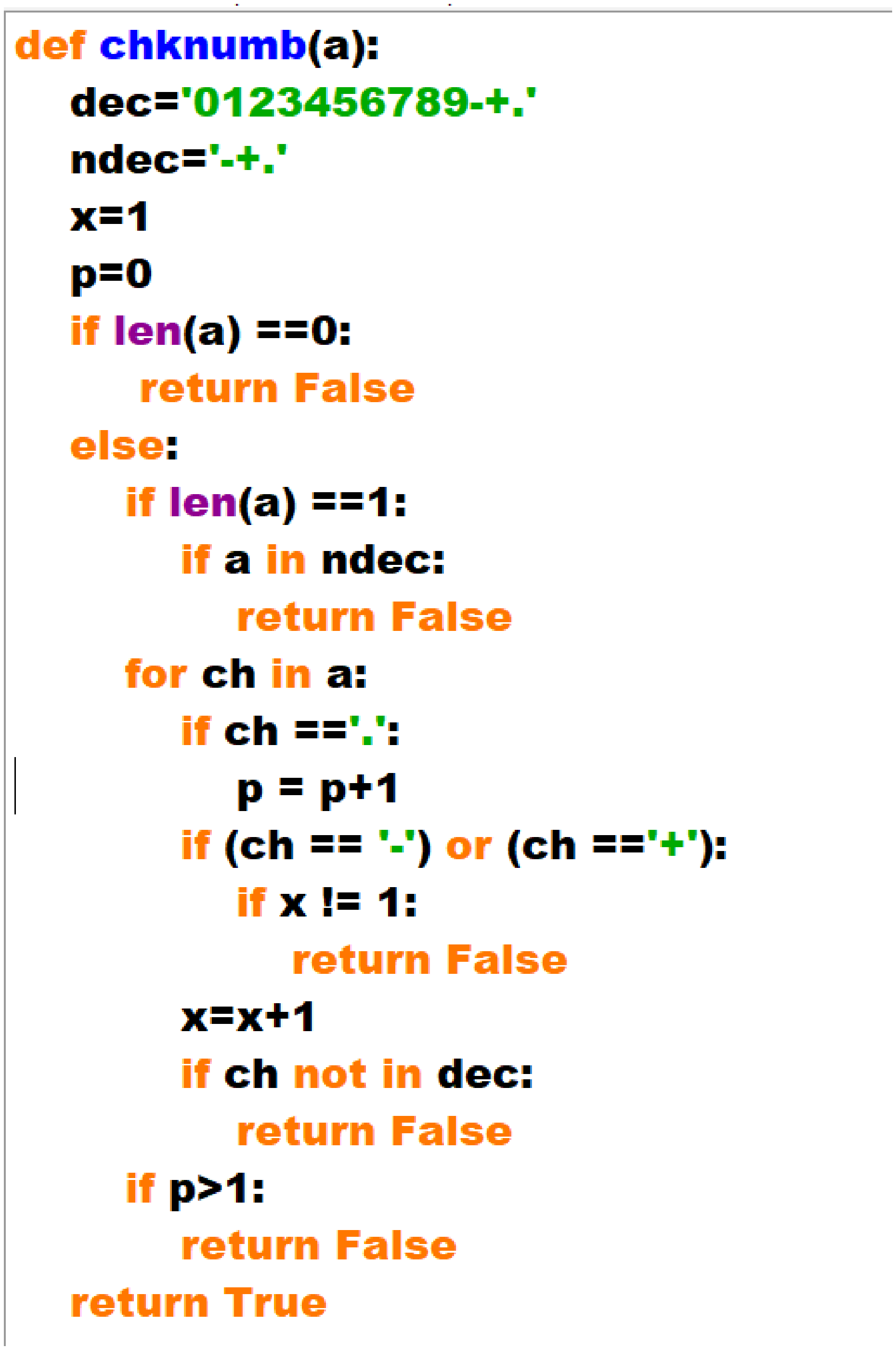
Example Program: concat

```
def stl10():
    cm = ','
    a = input('type something in: ')
    b = input('type something in: ')
    c = input('type something in: ')
    d = input('type something in: ')
    x = input(' inset commas 1=yes 0=no :')
    if x == '0':
        line = a + b + c + d
    else:
        line = a + cm + b + cm  + c + cm + d
    print('\n',line,'\n')
def main():
    ans = 'Y'
    while ans.upper()=='Y':
        stl10()
        ans=input(' again y/n :')
main()
```

stl10

Using strings to validate (string) numbers

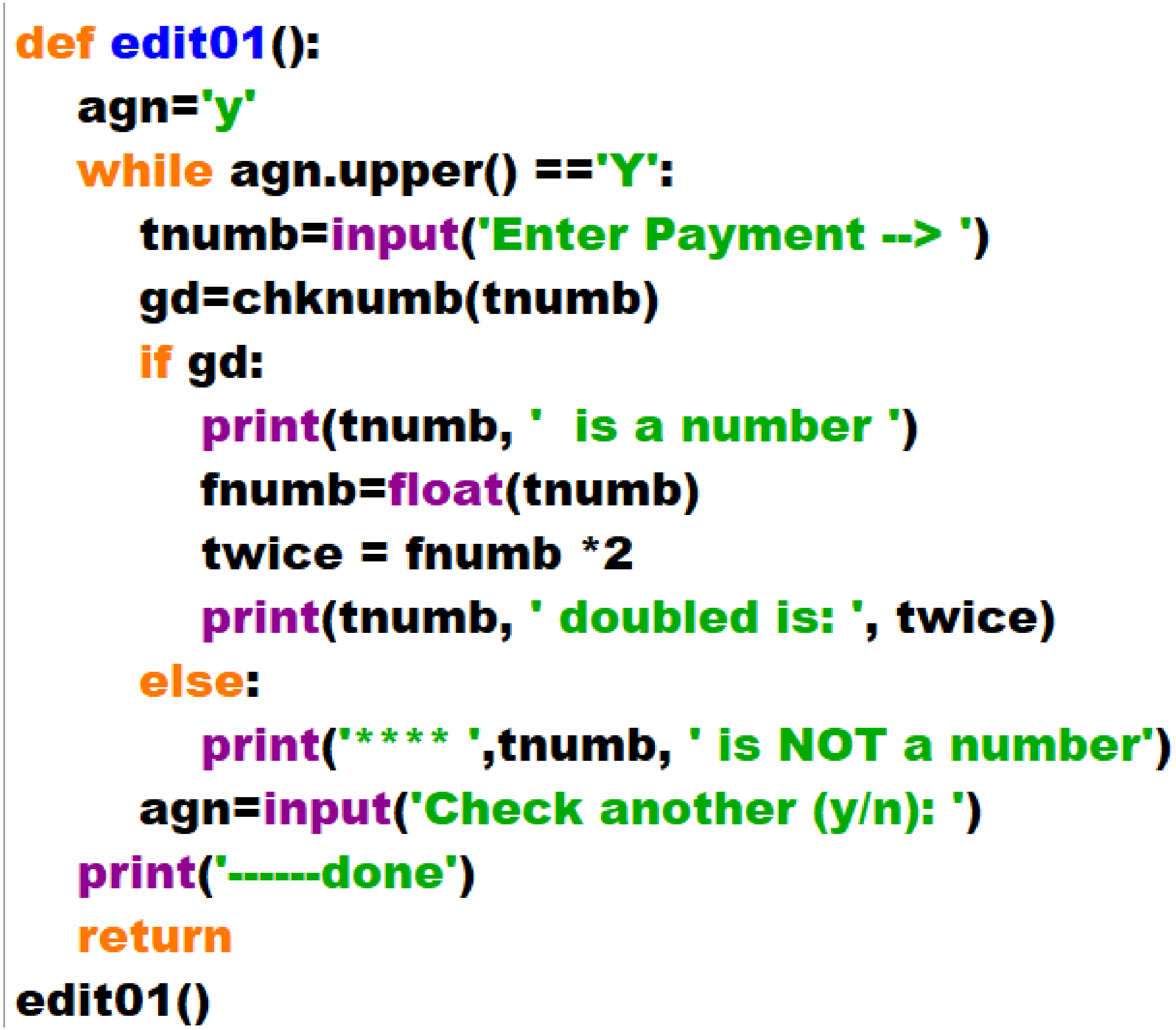
Part 1



```
def chknumb(a):
    dec='0123456789-+.'
    ndec='-+.'
    x=1
    p=0
    if len(a) ==0:
         return False
    else:
        if len(a) ==1:
            if a in ndec:
                return False
        for ch in a:
            if ch ==".":
                p = p+1
            if (ch == '-') or (ch =='+'):
                if x != 1:
                    return False
            x=x+1
            if ch not in dec:
                return False
        if p>1:
            return False
    return True
```

stredit1

Part 2



```
def edit01():
    agn='y'
    while agn.upper() =='Y':
        tnumb=input('Enter Payment --> ')
        gd=chknumb(tnumb)
        if gd:
            print(tnumb, '  is a number ')
            fnumb=float(tnumb)
            twice = fnumb *2
            print(tnumb, ' doubled is: ', twice)
        else:
            print('**** ',tnumb, ' is NOT a number')
        agn=input('Check another (y/n): ')
    print('------done')
    return
edit01()
```

done