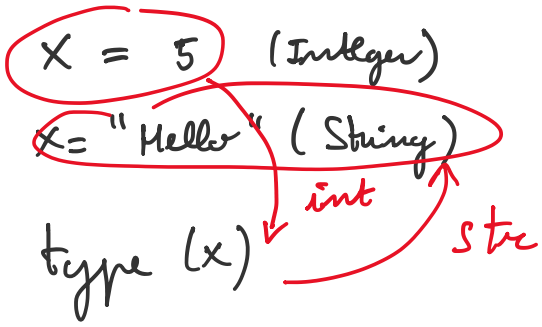
PYTHON

Friday, 19 October 2018 08:59

"PYTHON SCRIPTING FOR COMPUTATIONAL SCIENCE"
by Hans Petter Langtangen

www.w3schools.com/python

Variables



Numbes

- int (integer)
- float
- complex

```
x = 1      # integer
x = 1.0    # float
x = 1j     # complex
x = 2 + 3j
```

comment

Friday, 19 October 2018 09:15

CASTING

int() → cast to integer

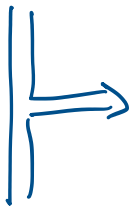
float() → cast to float

complex() → cast to complex

x = int(1) → x will be the integer 1

x = int(2.8) → x will be the integer 2

x = int("3") → x will be the integer 3



Same considerations hold with float(.)

x = str(4.0) → x will be the string "4.0"

Friday, 19 October 2018 09:20

STRING

a = "Hello" print(variable) ← print on the screen the variable

print(a) → on the screen you will get "Hello"

the string is like a vector

a[0] = "H"
a[1] = "e"
⋮

a[0:2] = "He"
a[2:] = "llo"
a[-1] = "o" ← is the last index
a[-2] = "l" ← is the second last

len(a) → gives the length of the string, i.e., 5

OPERATORS

Friday, 19 October 2018 09:28

Arithmetic operators

+	addition
-	subtraction
*	multiplication
/	division
**	x**y → x^y
%	modulus

x = "Hello"

y = "␣World!"

x+y → "Hello␣World!"

Comparison operators

==	equal
!=	equal
>	greater
<	less than
>=	greater or equal
<=	less than or equal

Logical operators

and, or, not

Membership operators

in, not in

ex. "e" in "Hello" returns True

PYTHON COLLECTIONS

Friday, 19 October 2018 09:40

LIST : is a collection, ordered and changeable

TUPLE : is a collection, ordered and not changeable

a = ["Hello", "World!", 3.0, 2]

a[0] = "Hello"

a[2] = 3.0

a[2] = 1.0 in this way a change the third element of the list.

"Hello" in a ⟹ TRUE

a.append("Hi") → a = ["Hello", "World!", 1.0, 2, "Hi"]

↑ add the argument to the list (on the very last position)

a.remove("Hi") → this remove "Hi" from the list

del a[0] → this removes the element at index 0 of the list

IF .. ELSE

Friday, 19 October 2018 09:50

```
if condition :
    statement
else :
    statement
```

Indentation is fundamental in Python

It works like { } in C

EXAMPLE

```
a = 10
b = 11
if a > b :
    print("Hello")
else :
    print("bye")
```

ELIF (ELSE IF)

```
if condition1:
    statement1
elif condition2 :
    statement2
else :
    statement3
```

ONE LINE

```
print("A") if a>b else print("B")
```

WHILE LOOP

Friday, 19 October 2018 10:00

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

→ equivalent to `i += 1`

FOR LOOP

↳ works with lists

for variable in list:
Statement

```
fruits = ["pear", "apple", "peach"]
for x in fruits:
    print(x)
```

first value, last value, step

```
for_range = range(0, 10, 1)
for_range = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
for i in for_range:
    print(i)
```

FUNCTION

Friday, 19 October 2018 10:11

```
def myfunction_name ( list of vaiables):
    statement
    return
```

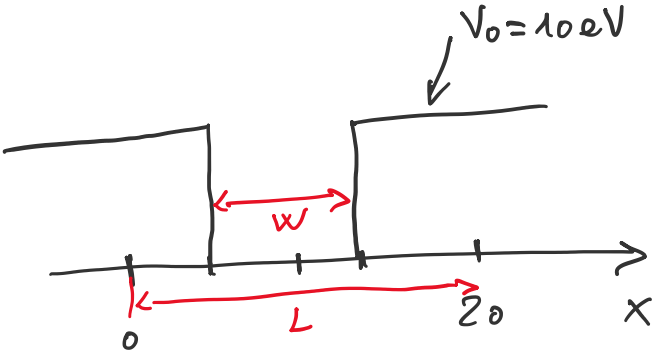
Example

```
def pippo(x):
    y = x+3;
    return y
```

pippo (3) ⟶ 6

Script Python

Thursday 18 October 2018 22:56




```
from numpy import *
from pylab import *
#I define the parameters
Np=300 #Number of grid points
L=20 #length of the discretized domain
W=10  #width of the potential well
hbar=1.05459e-34 #J/s
q=1.60219e-19    #C
m0=9.1095e-31    #Kg

#I create the grid
x=linspace(0,L,Np) # in nanometers

#I create the potential U(x) for the well
ind=nonzero(abs(x-L*0.5)>W*0.5);
U=zeros(Np);
U[ind]=10 #The height of the barrier is 10 eV


# Uncomment these three lines to plot the potential and exit
plot(x,U,'o-')
#show()
#exit(0);

#I create the hopping parameter -hbar**2/(2m0*a**2)
a=x[1]-x[0]
t=-hbar**2/(2*m0*(a*1e-9)**2)/q #eV

#I create the diagonal, updiagonal and lowdiagonal vectors
diago=(U-2*t)*(ones(Np));
lowdiag=t*ones(Np-1);
updiag=t*ones(Np-1);

#I create the Hamiltonian
H=diag(diago)+diag(lowdiag,-1)+diag(updiag,1)
H=array(H)
#H=mat(H)

[E,V]=eig(H)
print (hbar*pi/W*1e9)**2/2/m0/q,min(E)
# I plot the eigenvalues
for i in range(0,Np):
        plot(x,E[i]*ones(Np))

show()

#I plot the first 3 eigenfunctions
#plot(x,V[:,0])
#plot(x,V[:,1])
#plot(x,V[:,2])

show();
```