Python pour la physique

nov. 29, 2018

1 Exercices

1.1 Mathematical functions

Mathematical functions can be imported from the `math` module. Python uses standards name for mostly all functions : `sin`, `cos`, `tan`, `log`, `exp`, `sqrt`. The inverse trigonometric functions are `acos`, `asin`, `atan`. The `pi` constant can also be imported from the `math` module.

You can import all the content of the math module using the `import` statement

```
from math import *
```

To get familiar with the python terminal answer the following questions

- Is the `log` function the decimal one or the natural one ? How to choose the base ?
- Calculate $\sqrt{2}$. How many digits are displayed ?
- Calculate $\arccos\left(\frac{\sqrt{2}}{2}\right)$ et compare with the expected value

Solution :

```
from math import *

print("Logarithme of 10 :", log(10))
# This is the natural logarithm

# help(log) to see the documentation
print("Logarithme of 100 in base 10 :", log(100, 10))

print("Square root of 2 :", sqrt(2))
# 16 digits

print(acos(sqrt(2)/2))
print(pi/4)
```

1.2 Calcul of VAT

The VAT rate is 19.6 %.

- Create a function that calculates the price with taxes from the price without

Solution :

```
# The taxe rate should be written in a variable
VAT_rate = 19.6/100.

def HT_to_TTC(price):
    """ Calculate the price with taxes """
    return price*(1 + VAT_rate)

# Note that the name of the vaiable inside the function is explicit (price)
```

— Well, the VAT rate has change and is now 20 %. Modify the function so that the VAT rate is an optional parameter with a default value set by a global constant.

Solution :

1.3 Docstring

Write the documentation string (docstring) of the following function

```
from math import pi

def volume_cone(r,h):
    return pi*r**2*h/3
```

Solution :

```
from math import pi

def volume_cone(r,h):
    """Volume of a cone

    Arguments:
        r : radius of the circle at the botton
        h : height of the cone
    Output :
        Volume of the cone
    """
    return pi*r**2*h/3

# Note that you should avoid to use variable with a one letter name
# Name used in usual math formulae is an exception to this rule
```

1.4 Heron's formula

The Heron's formula is used to calculate the area of a triangle using the length of the three sides.

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

where

$$s = \frac{1}{2}(a+b+c)$$

— write the function that calculate the area of a triangle using the Heron's formula

Solution :

```
def triangle_area(a,b,c):
    """Area of a triangle using the Heron's formula

    The Heron's formula is used to calculate the area of a triangle using␣
↪the length of the three sides.

    Arguments :
        a,b,c : lenght of the three sides
    Output :
        Are of the triangle
    """
    s = (a+b+c)/2
    return sqrt(s*(s-a)*(s-b)*(s-c))
```

— What happens when the triangle does not exist ?

Solution :

```
# There is a ValueError because of the square root of a negative number
```

— Modify the function to raise a specific error when the triangle does not exist

Solution :

```
def triangle_area(a,b,c):
    """Area of a triangle using the Heron's formula

    The Heron's formula is used to calculate the area of a triangle using␣
↪the length of the three sides.

    Arguments :
        a,b,c : lenght of the three sides
    Output :
        Area of the triangle
    """
    s = (a+b+c)/2
    try:
        return sqrt(s*(s-a)*(s-b)*(s-c))
    except ValueError:
        raise ValueError("Cannot calculate the area of the triangle of␣
↪size {a}, {b} and {c} : this triangle doesn't exists".format(a=a, b=b,␣
↪c=c))
```

1.5 Derivative

The derivative of a function can be calculated using the following limit :

$$\lim_{\epsilon\to 0}\frac{f(x+\epsilon)-f(x)}{\epsilon}$$

An approximation of the derivative is obtained using a suffitiently small value of *epsilon*

— By choosing $\epsilon = 10^{-6}$, calculate the derivative of sine for $x = 1$. Compare with the theoretial result

— Compare the result with the one obtined using the formula :

$$\lim_{\epsilon\to 0}\frac{f(x+\epsilon)-f(x-\epsilon)}{2\epsilon}$$

— Write a function that take the **function** f as an argument and return the derivative of f as a **function**.

Solution :

```
from math import sin, cos

epsilon = 1E-6
x = 1.

derivee_calculee = (sin(x+epsilon) - sin(x))/epsilon
derivee_theorique = cos(x)

print('Valeur numerique :',derivee_calculee)
print("Ecart avec la valeur theorique (formule 1):",  derivee_calculee-
↪derivee_theorique)

derivee_calculee = (sin(x+epsilon) - sin(x-epsilon))/(2*epsilon)
print("Ecart avec la valeur theorique (formule 2):",  derivee_calculee-
↪derivee_theorique)


def derivative(f, epsilon=1E-5):
    def derivative_of_f(x, epsilon=epsilon):
        return (f(x+epsilon) - f(x-epsilon))/(2*epsilon)
    derivative_of_f.__name__ = f.__name__ + '_prime'
    derivative_of_f.__doc__ = "Derivative of the function {f.__name__}\n\n
↪{f.__doc__}".format(f=f)
    return derivative_of_f

# Note that one can modify the name and docstring of the function
# For nested function (function inside functions), assignements to name go␣
↪to the innermost scope.
# Insinde "derivative_of_f", the name f correspond to the variable 'f' of␣
↪the local scope of
# the function derivative.
```

1.6 Floating point numbers

The number *x=1.0* is stored as a float on 64 bits (double precision, using the IEEE 754, http://en.wikipedia.org/wiki/IEEE_754 .

Answer the following questions :

— What is the result of (x+2E-20) - x ? Why ?
— What is the smallest number y such that x + y is not equal to x
— What is **exactly** the difference between (x+1E-15) and x ?
— For *epsilon<1* what is the order of magnitude of the relative difference between ((x+epslion) - x) and epsilon ?

Solution [: :] # 0 because the number of digit stored is about 16

# This number is of the order of $2{\cdot}10^{-16}$ (2^{-52}). The mantissa of x+y is then exactly # 00000....01

# We should calculate the mantissa of $1 + 10^{-15}$, i.e find m such that # $1 + m2^{-52}$ is closed to $1 + 10^{-15}$. We obtain 5. The result # is (x+1E-15) - x = $5 \cdot 2^{-52}$

# The difference between ((x+epslion) - x) and epsilon is about 2^{-52} i.e. $2 \cdot 10^{-16}$. # The relative error is then 2E-16 / epsilon

2 Loops

2.1 Sequence limit

Consider the following sequence

$$u_{n+1} = \frac{1}{1+u_n}$$

with

$$u_0 = 0$$

Thie sequence converge. The objective is to calculate its limit

— First calculate the N first elements (take N=10)

Solution :

```
i = 0
u = 0
N = 10
while i<N:
    u = 1/(1+u)
    i = i+1

print("La valeur de u_10 est :", u)
```

— Use a while loop that stops when the difference between two terms is less than ϵ. The calculation will be done with $\epsilon = 10^{-8}$

Solution :

```
u_precedent = 0.
u = 1.
epsilon = 1E-8
while abs(u_precedent - u)>epsilon:
    u_precedent = u
    u_n = 1/(1+u)

print("La value of u is :", u)
```

— Compare with the theoretical limit :

$$\frac{-1+\sqrt{5}}{2}$$

Solution :

```
from math import sqrt
print("Theoretical value", (-1 + sqrt(5))/2)
```

2.2 Series calculation

Calculate the sin function from its expansion :

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{(2n+1)}}{(2n+1)!}$$

Solution :

```
# TODO
```

2.3 For loop

Print all the odd number below 100 that are multiple of 3 but not of 5

Solution :

```
for i in range(1,100,2):
    if (i%3 == 0) and not (i%5==0):
        print(i)
```

2.4 Prime number

Write a function that returns `True` if a number is prime and `False` else wise. We will test if a number is a prime number by succesfully testing if it can be divided by a number larger than 2 and smaller than $\sqrt{n}$

— Write the function

Solution :

```
from math import sqrt, ceil
def is_prime(n):
    """Return wether a number is prime or not"""
    n_max = int(ceil(sqrt(n)))
    for i in range(2, n_max+1):
        if n%i==0:
            return False
    return True
```

— Is 2011 a prime number ?

Solution :

```
if is_prime(2011):
    print("2011 is a prime number")
else:
    print("2011 is not a prime number")
```

— During the 21st century, how many year numbers will be prime ?

Solution :

```
nb_prime_years = 0
for year in range(2001, 2101):
    if is_prime(year):
        nb_prime_years += 1

print("The number of prime year is ", nb_prime_years)
```

3 List

3.1 simple list

In the Python console, create an empty list called py_list. Add the number 2.7, 5.1 and 7 at the end of the list. Add the number 4 at the begining of the list and the number 2.9 at the position 2. Supress the item number 2. Print the list and compare to your neighbour.

Solution :

```
my_list = []
my_list.append(2.7)
my_list.append(5.1)
my_list.append(7)

my_list.insert(0, 4)
my_list.insert(1, 2.9)
del my_list[1]

print(my_list)
```

3.2 Research inside a list

— How to find the index of an item inside a list ? (use `help list`)
— Using a for loop, write your own function : from a list `l` and a value `x` the function will return the smallest `i` such that `l[i]==x`.

Solution :

```
def recherche(l,x):
    """Recherche dans une liste

    Cette fonction recherche dans la liste l la premier occurence de x.
    Elle renvoie l'indice si il y a un succés.
    """
    for i,elm in enumerate(l):
        if elm==x:
            return i
    return None

print(recherche([1,2,5,3], 5))
```

3.3 List of random number

The `random` package contains functions to generate random number. To obtain a number between 1 and 10 use

```
from random import randint
print(randint(1,10))
```

We have two dices (with 6 faces). We consider the sum of the value of the two dices.
— Create a list containing `n` random realization.
— What is the number of 8 in the list.

— Using 100000 realization, what is the probability to get 8 ?

Solution :

```
from random import randint

def liste_tirage_deux_des(n):
    liste = []
    for i in range(n):
        liste.append(randint(1,6)+randint(1,6))
    return liste

ma_liste = liste_tirage_deux_des(100)
ma_liste.count(7)

n = 100000
print(liste_tirage_deux_des(n).count(8)/n)
# It shoud be close to 5/36.
```

— Same question with 4 dices (2 with 6 faces and 2 with 4 faces).

Solution :

```
def liste_tirage_quatre_des(n):
    liste = []
    for i in range(n):
        liste.append(randint(1,6)+randint(1,6)+randint(1,4)+randint(1,4))
    return liste

n = 100000
print(liste_tirage_quatre_des(n).count(8)/n)
```

4 Strings

4.1 Unicode strings

There are 25 letters in the greek alphabet. In the unicode, they start from 945 to 969.

— Display all of them (use the chr function).

— Write a function that return the greek letter from its position in the greek alaphabet. For example, the function will return α if the argument is 1. Create an exception if necessary.

Solution :

```
for i in range(945, 945+25):
    print(chr(i))

def lettre_grec(n):
    if (n>=1) and (n<=25):
        return chr(945-1 + n)
    else:
        raise Exception("n should be less than 25")

print(lettre_grec(3))
```

4.2 String and file

Take a large text file (for example Romeo and Juliet from Shakespeare http://www.gutenberg.org/files/47960/47960-0.txt).

- Print the distribution of the letter.

Solution :

```python
import urllib.request
url = "http://www.gutenberg.org/files/47960/47960-0.txt"
f = urllib.request.urlopen(url)
txt = f.read().decode('latin-1').lower()

letters = set([chr(i) for i in range(ord('a'), ord('a')+26)])

for char in letters:
    print(char, txt.count(char))
```

- How many words starts with an `e` ?

Solution :

```python
cpt = 0
for word in txt.split(' '):
    if word.startswith('e'):
        cpt += 1

print('{cpt} words starts with an e'.format(cpt=cpt))
```

4.3 Module and package

We want to store physical constants in a package.

- Create a package `constants` with a two modules : the first one called `fundamental` and the second called `atomic_mass`. They will be used as follow

  ```python
  from constants.fundamental import mu_0, h, e, c
  from constants.atomic_mass import rubidium_87
  ```

In case you don't know, in SI units, one has :

$$c = 299792458; \mu_0 = 4\pi \times 10^{-7}; \epsilon_0 = 1/\mu_0 c^2; G = 6.6738 \times 10^{-11};$$
$$h = 6.6260695 \times 10^{-34}; \hbar = h/2\pi;$$

And for atomic masses :

$$M(^{87}\mathrm{Rb}) = 86.909180527 m_u; M(^{85}\mathrm{Rb}) = 86.909180527 m_u;$$

- Modify the `__init__.py` so that `from constants import mu_0,h,e` works.
- Create a setup.py, install it (using `pip install -e . --user`) and try to use it from a different directory

Solution :