Programming-Practice-1-computational_physics_Peniel_PH22DU55

January 13, 2023

0.1 1- function that computes the n-th nmber of the Fibonacci series

[5]:
```
def fibo(n):
    T0 = 0
    T1 = 1
    for i in range(1, n+1):
        T2 = T1
        T3 = T0
        T = T2 + T3
        T0 = T
        T1 = T3
    return T0
fibo(3)
```

[5]: 2

0.2 2- function that checks if a given number is prime or not

[6]:
```
def isPrime(n):
    isprime = False
    if n%2 == 0:
            if n == 2:
                isprime = True
                return isprime
            else:
                return isprime
    elif n == 1:
        print('The number is neither prime nor composite')
    else:
        for i in range(2, n):
            if n%i == 0:
                break
            else:
                isprime = True
        return isprime
isPrime(17)
```

[6]: True

0.3 3- Function that converts a coordinates in 3-D cartesian coordinate system to cylindrical coordinate system

[11]:
```
def carte_cylindrical(x,y,z):
    import math
    theta = math.atan(y/x)
    r = math.sqrt(x ** 2 + y ** 2)
    z = z

    return r, theta, z
carte_cylindrical(2,-5, 4)
```

[11]: (5.385164807134504, -1.1902899496825317, 4)

0.4 4- Program to print the pattern

[4]:
```
n = 7
for i in range(1, n+1):
    l1 = [i for j in range(i)]
    for k in l1:
        print(k, end = '')
    print('\n')
```

```
1

22

333

4444

55555

666666

7777777
```

0.5 5- Let a list of size 3 represent the cartesian components of a vector in 3D. Write functions that compute the following

1. Length of the vector
2. Dot product of two vectors
3. Cross product of two vectors

[10]:
```
vect1 = [2, 3, 5]
vect2 = [2, 6, 9]

def length(vect):
    import math
    abs = math.sqrt(vect[0] ** 2 + vect[1] ** 2 + vect[2] ** 2)

    return abs

def dot_product(vect1, vect2):
    a = 0
    for i in range(3):
        a += vect1[i] * vect2[i]
    return a

def cross_product(vect1, vect2):
    result = []
    for i in range(3):
                if i < 2:
                    a = vect1[i+1] * vect2[i-1] - vect1[i-1] * vect2[i+1]
                    result.append(a)
                else:
                    a = vect1[i-2] * vect2[i-1] - vect1[i-1] * vect2[i-2]
                    result.append(a)
    return result
print(length(vect1),
dot_product(vect1, vect2),
cross_product(vect1, vect2))
```

```
6.164414002968976 67 [-3, -8, 6]
```

[ ]:

[ ]: