lab

January 17, 2020

0.1 Introduction to Scientific Computing

Jeff Shen, 1004911526

PHY224 | Winter 2020

24 Jan 2020

[1]:
```
# imports
import numpy as np
from scipy.integrate import simps
```

0.1.1 Question 1: numpy operations

[2]:
```
a = 1
b = np.array([3.0,2.3,1.0])
c = np.array([3,.3,.03])
d = np.array([[2,4],[4,6],[7,8]])
f = [9,90,900]
```

[3]:
```
# subquestion 1:
print(a+b)

# subquestion 2:
print(c+b)

# subquestion 3:
print(np.expand_dims(c, 1) + d)

# subquestion 4:
print(b+f)

# subquestion 5:
print(type(d), type(f))

# subquestion 6:
print(len(d), len(f))
```

```
[4.  3.3 2. ]
[6.   2.6  1.03]
[[5.   7.  ]
 [4.3  6.3 ]
 [7.03 8.03]]
[ 12.   92.3 901. ]
<class 'numpy.ndarray'> <class 'list'>
3 3
```

0.1.2 Question 2: numpy functions

[4]:
```
t = np.array([0. , 0.1155, 0.2287, 0.3404, 0.4475,0.5546, 0.6607, 0.7753, 0.
 ↪8871, 1. ])
y = np.array([0. , 0.1655, 0.2009, 0.1124,-0.0873, -0.3996, -0.8197, -1.3977,-2.
 ↪0856, -2.905 ])
```

[5]:
```
# subquestion 1:
print(y.mean())

# subquestion 2:
print(y.std(ddof=1))

# subquestion 3:
print(np.diff(y)/np.diff(t))

# subquestion 4:
print(simps(1/2*np.square(y), t))
```

```
-0.7216099999999999
1.0777910583431485
[ 1.43290043  0.31272085 -0.79230081 -1.86461251 -2.91596639 -3.9594722
 -5.04363002 -6.1529517  -7.25775022]
0.6249779509514343
```

0.1.3 Question 3: Implementing equations in Python

[6]:
```
# subquestion 1:
g = 9.81 # meters per second
def pendulum_acceleration(length, period):
    return 2*np.pi*np.sqrt(length / g)

print(pendulum_acceleration(2.50, 5.16))

# subquestion 2:
# rearrange equation to solve for g:
# g = 4pi * l / T
def uncertain(l, l_uncert, t, t_uncert):
```

```
    l_uncert_perc = l_uncert / l # as % of l
    t_sq_uncert_perc = (t_uncert / t) * 2 # as % of t^2
    l_over_t_sq_uncert_perc = l_uncert_perc + t_sq_uncert_perc # as % of l/t^2
    return 4 * np.square(np.pi) * l_over_t_sq_uncert_perc

print(uncertain(2.4, 0.01, 5., 0.01))
```

```
3.1718699246097066
0.3224070771022523
```

0.1.4 Question 4: Conditionals

[7]:
```
def gpa(mark):
    if mark >= 85: return 4.
    elif mark >= 80: return 3.7
    elif mark >= 77: return 3.3
    elif mark >= 73: return 3.
    elif mark >= 70: return 2.7
    elif mark >= 67: return 2.3
    elif mark >= 63: return 2.
    elif mark >= 60: return 1.7
    elif mark >= 57: return 1.3
    elif mark >= 53: return 1.
    elif mark >= 50: return 0.7
    else: return 0.
```

0.1.5 Question 5: Loops

[8]:
```
marks = np.array([72, 82, 72, 72, 79, 57, 59, 71, 66, 80, 67, 62, 91, 74, 77,␣
↪62, 71, 78, 65, 80, 70, 74, 70, 95, 76, 66, 85, 64, 79, 57, 63, 78, 84, 78,␣
↪75, 73, 62, 69, 72, 87])
```

[9]:
```
vectorized_gpa = np.vectorize(gpa)
print(vectorized_gpa(marks).mean())
```

```
2.77
```

0.1.6 Question 6: Data Comparison

[10]:
```
def agrees(measurement, uncertainty, known):
    return np.abs(known - measurement) <= uncertainty

print(agrees(19.2, 0.1, 19.41))
print(agrees(19.5, 0.8, 19.41))
print(agrees(19.5, 0.1, 19.41))
```

```
False
True
True
```