Astrophysics III : Stellar and galactic dynamics

Solutions

Problem 1 :

To write a python script on `noto`, you could either open a new "python file" or a "text file" from the launcher (ctrl + shift + L). Alternatively, you could open a new terminal and write the script using a terminal editor like `vim` or `nano`. Regardless of which choice you make, you will have to launch the script from the terminal though.

Option 1 :

1. Write a script with a shebang (must be the top line of the script !).

```
#!/usr/bin/env python3

print('Hello world!')
```

   and save it as the file `my_first_exercise.py` .
2. Make the file executable :
   `$chmod u+x my_first_exercise.py`
3. Execute the file :
   `$./my_first_exercise.py`
   `Hello world!`

Option 2 :

1. Write a script without a shebang (not recommended, but works fine).

```
print('Hello world!')
```

   and save it in as the file `my_first_exercise.py` .
2. Execute the file by providing it to the `python3` executable :
   `$python3 my_first_exercise.py`
   `Hello world!`

Option 2 also always works if you included the shebang as shown in option 1. Hence we recommend to always add the shebang.

Problem 2 : Functions

```
#!/usr/bin/env python3

# defining a function
def add_up(x, y):
    """
    This function adds up x and y and returns the result.
    """
    result = x + y
    return result


# executing it a few times
x = 2
y = 3
print(x, '+', y, '=', add_up(x, y))

x = 23
y = 34
print(x, '+', y, '=', add_up(x, y))
```

Problem 3 : Conditionals

```
#!/usr/bin/env python3

# defining a function to sum up two variables
def add_up(x, y):
    """
    This function adds up x and y and returns the result.
    """
    result = x + y
    return result

# defining a function to tell me whether the result is < 100
def is_smaller_than_100(x, y):
    """
    This function prints to screen whether the result of x + y
    is smaller than 100.
    """
    result = add_up(x, y)
    if result < 100:
        print('The result is < 100')
    else:
        print('The result is > 100')

    return


# executing it a few times
is_smaller_than_100(23, 44)
is_smaller_than_100(123, 32)
```

```
is_smaller_than_100(102, -4)
```

Problem 4 : Writing and importing modules

File `my_first_module.py` :

```
# this module file is not meant to be executed, so it doesn't need
# a shebang. The functions and variables written in here will only
# be imported.

# defining a function to sum up two variables
def add_up(x, y):
    """
    This function adds up x and y and returns the result.
    """
    result = x + y
    return result

# defining a function to tell me whether the result is < 100
def is_smaller_than_100(x, y):
    """
    This function prints to screen whether the result of x + y
    is smaller than 100.
    """
    result = add_up(x, y)
    if result < 100:
        print('The result is < 100')
    else:
        print('The result is > 100')

    return

# variables
x_module = 20
y_module = 35
```

File `problem_4.py` :

— Option 1 :

```
#/usr/bin/env python3

import my_first_module
x = my_frist_module.x_module
y = my_first_module.y_module
my_first_module.is_smaller_than_100(x, y)

# you could just as well do
# my_first_module.is_smaller_than_100(my_frist_module.x_module,
    my_first_module.y_module)
# but this way is easier to read
```

— Option 2 :

```python
#/usr/bin/env python3

import my_first_module as m # just a 'renaming' when you're using it

m.is_smaller_than_100(m.x_module, m.y_module)
```

— Option 3 :

```python
#/usr/bin/env python3

# import only what you need
from my_first_module import x_module, y_module, is_smaller_than

is_smaller_than_100(x_module, y_module)
```

Problem 5 :

```python
#/usr/bin/env python3

# create empty lists
odd = []
even = []

for i in range(101): # starts at 0 by default, 101 not included
    # skip divisible by 5
    if i % 5 == 0:
        continue
    if i % 2 == 0:
        even.append(i)
    else:
        odd.append(i)
```

Problem 6 : Numpy essentials

```python
#/usr/bin/env python3

import numpy as np

array1 = np.array([1, 3, 5, -2, -8, 20, 43, 54, -9, 0])

# minimum
array1.min()

# maximum
array1.max()

# mean
array1.mean()

# sum
array1.sum()
```

```
# standard deviation
array1.std()

# position of maximal value in array
array1.argmax()

# position of minimal value in array
array1.argmin()

# to print out the results, you can just call the function inside
# the print function, e.g.
# print(array1.min())



array2 = np.array([7., 16.0324, 8.29342, 9.234, 724.243, 712.321,
    74293.783, 123.13, 523.423, 423.41])

# square root
np.sqrt(array2)

# ln
np.log(array2)

# base 10 logarithm
np.log10(array2)

# exponential
np.exp(array2)

# ^2.3
array2**2.3

# sine
np.sin(array2)


# array-on-array operations
# no need for loops, numpy does it for you!

new_array1 = array1 + array2
new_array2 = array1 * array2

array3 = 3.4 * array2
array3[:2] *= array1[:2] # :2 = all elements until 2, 2 not included.
array3[-2:] -= array1[:2] # -2: = starting from second to last element,
    take all
```

Problem 7 : Matplotlib essentials

```
#/usr/bin/env python3
```

```
    from matplotlib import pyplot as plt
    import numpy as np

    def f(x):
        return 18 * np.sin(23 * x) * np.exp(-x**2)

    # linear intervals from 0 to pi, 1000 intermediate steps
    x = np.linspace(0, np.pi, 1000)

    plt.figure()
    plt.plot(x, f(x))
    # on the noto server, the function plt.show() doesn't lead to
    # the expected result when launched from a python script. But
    # you can use it inside a jupyter notebook.
    plt.show() # for jupyter notebooks
    # If you're working with a script, you'll have to save the
    # figure first and then open it with the file manager:
    plt.savefig('my_figure_name.png') # for python scripts
```

Problem 8 : Matplotlib advanced

```
#!/usr/bin/env python3

from matplotlib import pyplot as plt
import numpy as np


x = np.linspace(0, 10, 1000)

def f(x):
    return -3 * np.log(x) + 5

def g(x):
    return 2*np.sin(x-0.5*np.pi)

def h(x):
    return 12*np.exp(-(x-4.3)**2)


plt.figure()
plt.plot(x, f(x), label="f(x)")
plt.plot(x, g(x), label="g(x)")
plt.plot(x, h(x), label="h(x)")
plt.legend() # add legend
plt.xlabel("x") # x axis label
plt.ylabel("functions of x") # y axis label
plt.title("The best title in the world") # title
plt.savefig("problem_8_plot.png") # save figure
```