Python Modules

Introduction.

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code

Objectives

Objectives by the end of this topic you should be able to:

- Roles of Python Module
- import Statement
- Standard Library Modules in Python
- How does import Work?

Learning activities

Learning Activity 11.1: Reading

Read further on python modules.

Learning Activity 11.2: Journal

Describe the usage of dir() function.

Learning Activity 11.3: Discussion

In groups of three discuss python standard modules.

Assessment

Topic resources

1. The Python Tutorial. (n.d.). Retrieved from https://docs.python.org/3/tutorial/index.html
2. Mueller, J. P. (n.d.). *Beginning Programming with Python For Dummies*. S.l.: For Dummies.
3. (n.d.). Python 3.7.4 documentation. Retrieved from https://docs.python.org/3
4. (n.d.). Git Handbook. Retrieved from https://guides.github.com/introduction/git-handbook/
5. Shaw, Z. (2017). *Learn Python 3 the hard way: a very simple introduction to the terrifyingly beautiful world of computers and code*. Boston: Addison-Wesley.
6. Bader, D. (2018). *Python tricks: the book*. Vancouver, BC: Dan Bader.
7. Downey, A. B. (2015). *Think Python*. Sebastopol: OReilly.

8. Ramalho, L. (2016). *Fluent Python:*Beijing: OReilly.

URL Links

https://www.tutorialspoint.com/python/python_modules.htm
https://www.w3schools.in/python-tutorial/modules/
https://data-flair.training/blogs/python-modules/
https://www.w3schools.com/python/python_modules.asp

Python Modules

A module is a file containing Python definitions and statements. A module can define functions, classes and variables. A module can also include runnable code. Grouping related code into a module makes the code easier to understand and use.

Example:

```
# A simple module, calc.py

def add(x, y):
    return (x+y)

def subtract(x, y):
    return (x-y)
```

The *import* statement

We can use any Python source file as a module by executing an import statement in some other Python source file.
When interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches for importing a module. For example, to import the module calc.py, we need to put the following command at the top of the script :

```
# importing module calc.py
import calc

print(add(10, 2))
```

Output:

```
12
```

The *from import* Statement

Python’s *from* statement lets you import specific attributes from a module. The *from* .. *import* .. has the following syntax :

```python
# importing sqrt() and factorial from the
# module math
from math import sqrt, factorial

# if we simply do "import math", then
# math.sqrt(16) and math.factorial()
# are required.
print(sqrt(16) )
print(factorial(6))
```

Output:

```
4.0

720
```

The dir() function

The dir() built-in function returns a sorted list of strings containing the names defined by a module. The list contains the names of all the modules, variables and functions that are defined in a module.

```python
# Import built-in module random
import random
print(dir(random))
```

Output:

```
['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random',

'SG_MAGICCONST', 'SystemRandom', 'TWOPI', 'WichmannHill',

'_BuiltinMethodType', '_MethodType', '__all__',

'__builtins__', '__doc__', '__file__', '__name__',

'__package__', '_acos', '_ceil', '_cos', '_e', '_exp',

'_hashlib', '_hexlify', '_inst', '_log', '_pi', '_random',

'_sin', '_sqrt', '_test', '_test_generator', '_urandom',

'_warn', 'betavariate', 'choice', 'division',

'expovariate', 'gammavariate', 'gauss', 'getrandbits',

'getstate', 'jumpahead', 'lognormvariate', 'normalvariate',
'paretovariate', 'randint', 'random', 'randrange',

'sample', 'seed', 'setstate', 'shuffle', 'triangular',

'uniform', 'vonmisesvariate', 'weibullvariate']
```

Code Snippet illustrating python built-in modules:

```
# Cosine of 0.5 radians
print math.cos(0.5)

# Tangent of 0.23 radians
print math.tan(0.23)

# 1 * 2 * 3 * 4 = 24
print math.factorial(4)


# importing built in module random
import random

# printing random integer between 0 and 5
print (random.randint(0, 5))

# print random floating point number between 0 and 1
print (random.random() )

# random number between 0 and 100
print (random.random() * 100)

List = [1, 4, True, 800, "python", 27, "hello"]

# using choice function in random module for choosing
# a random element from a set such as a list
print (random.choice(List) )


# importing built in module datetime
import datetime
from datetime import date
import time
```

```
# Returns the number of seconds since the
# Unix Epoch, January 1st 1970
print (time.time() )

# Converts a number of seconds to a date object
print (date.fromtimestamp(454554))
```

Output:

```
5.0

3.14159265359

114.591559026

1.0471975512

0.909297426826

0.87758256189

0.234143362351

24

3

0.401533172951

88.4917616788

True

1461425771.87

1970-01-06
```

Revision questions

1. Explain how to execute python modules as scripts
2. List and explain various standard python modules
3. What does the dir() function return