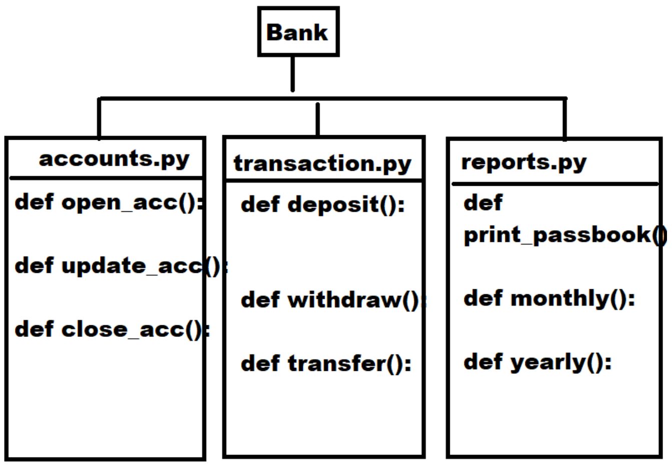
Modules and Packages

What is modular programming?

Modular programming allows dividing application functionality into number of programs (modules).

Advantage:

1. Easy to understand and maintain code
2. Reusability between programs
3. Efficient way of developing projects



What is module?

Python program is called module (OR)
Module is nothing but a python program (OR) .py file
A module is collection of variables, functions and classes.

Python modules are 2 types

1. Predefined modules
2. User defined modules

Predefined modules

Existing modules/programs are called predefined modules.
These are libraries.
Example: sys, datetime, calendar, os,..

User defined modules

Programmer developed modules/programs are called user defined modules. These are application specific modules.

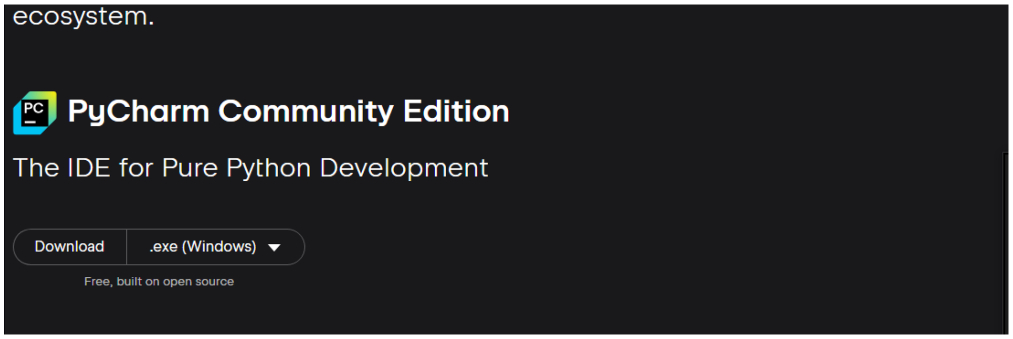
Creating module is nothing but writing python program.

PyCharm

PyCharm is python editor or IDE

How to download pycharm

https://www.jetbrains.com/pycharm/download/?section=windows



ecosystem.

PyCharm Community Edition

The IDE for Pure Python Development

Modules can be,

1. Executable modules
2. Reusable modules

What is executable module?
A module or program which is able executed (OR) which are having executable statements.

What is reusable module?
A module or program which does not have executable statements is called reusable module. The content of this module is used inside other modules or programs.

How to use the content of one program/module inside another module/program?

import keyword

import is a keyword, this keyword is used for importing or using the content of one module inside another module.

Syntax-1: import <module-name>
Syntax-2: import <module-name> as <alias-name>
Syntax-3: from <module-name> import <content>(variables/function/classes>
Syntax-4: from <module-name> import <content> as <alias-name>
Syntax-5: from <module-name> import *

import module-name
This syntax import the module-name as part of current module.

Module1.py	**Module2.py**

```
x=100 # global variable
y=200 # global variable
def fun1():
    print("inside fun1 of module1")

def fun2():
    print("inside fun2 of module1")

def fun3():
    print("inside fun3 of module1")
```
	```
	import module1

	module1.fun1()
	module1.fun2()
	module1.fun3()
	print(module1.x)
	print(module1.y)
	```

Syntax-2: import module-name as alias-name
This syntax allows importing module with alias name or another name

umath.py	**module3.py**

```
def factorial(num):
    fact=1
    for i in
range(1,num+1):
        fact=fact*i
    return fact

def is_prime(num):
    c=0
    for i in
range(1,num+1):
        if num%i==0:
            c=c+1
    return c==2

def count_digits(num):
    c=0
    while num>0:
        num=num//10
        c=c+1
    return c
```

```
import umath as um

res1=um.factorial(4)
res2=um.count_digits(369)
res3=um.is_prime(7)

print(res1,res2,res3)
```

Syntax-3: from module-name import identifiers

This syntax import identifiers (variables, functions and classes) of module as part of current module (namespace)

Users.py

```
users_dict={'nit':'n123',

'naresh':'naresh1',

'suresh':'s321'}

def login(user,pwd):
    if user in users_dict
and
```

Module4.py

```
from users import login

username=input("UserName
")
pwd=input("Password ")
b=login(username,pwd)
if b:
    print(f"{username}
Welcome")
```

```
users_dict[user]==pwd:
            return True
      else:
            return False
```

```
else:
      print("invalid
username or password")
```

Syntax-4: from <module-name> import <content> as <alias-name>

This syntax allows importing identifiers with alias name or alternative name

module5.py

```
x=100
y=200

def add():
      return x+y
def sub():
      return x-y
def multiply():
      return x*y
def div():
      return x/y
```

module6.py

```
from module5 import add as
add_two
from module5 import sub as
sub_two
def add():
      return "NIT"+"PYTHON"

res1=add_two()
print(f'Sum is {res1}')
res2=sub_two()
print(f'Diff is {res2}')
res3=add()
print(res3)
```

Syntax-5: from <module-name> import *
This syntax import all the identifiers as a part of current module

Module7.py

```
from module5 import *

res1=add()
res2=sub()
res3=multiply()
```

```
res4=div()

print(res1,res2,res3,res4)
```

Q: What is __name__?
__name__ is a predefined variable. It is available in default module imported by python program (__builtins__)

The value of __name__ is module-name or __main__

The value of __name__ is module name, if it is imported inside another module.
The value of __name__ is __main__, if is executed as module

If __name__=='__main__':

This condition is included in every module, to use executable module as reusable module.

Module5.py

```
x=100
y=200

def add():
    return x+y
def sub():
    return x-y
def multiply():
    return x*y
def div():
    return x/y

if __name__=='__main__':
    res1=add()
    res2=sub()
    res3=multiply()
    res4=div()
```

Module7.py

```
import module5

print(module5.add())
```

`print(res1,res2,res3,res4)`	

Dynamic loading modules