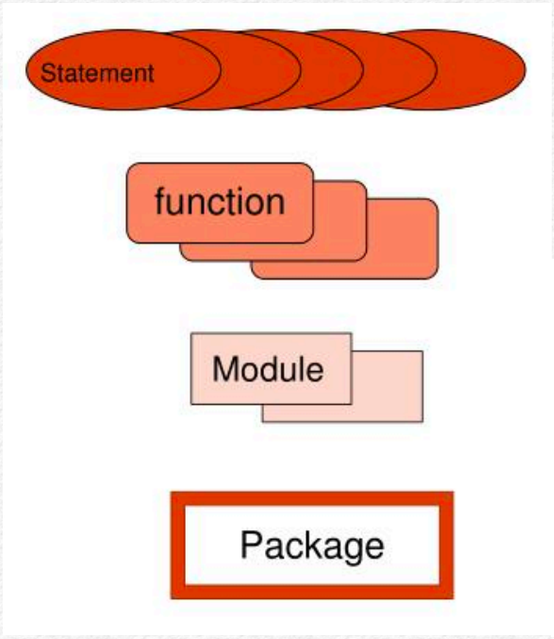
함수만들기

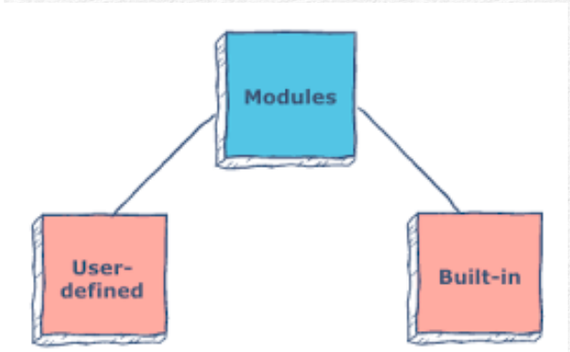
코드 - 함수 - 모듈 - 패키지 순의 크기를 가진다.

개요

개념

파이썬 코드는 interpreter에 입력되고 실행되나 종료하면 사라져 버린다. 향후 반복 작업을 위하여 코드 재사용이 필요하므로 “반복 가치가 있는 코드”를 묶어 설정한 “입력 값”에 사용자가 입력하면 원하는 “결과값” 출력하는 코드 집합을 만들어 놓는다. 이를 함수 < 모듈 < 패키지라 한다.





함수와 클래스

클래스는 C언어에는 없는 개념으로 굳이 클래스 없이도 파이썬 프로그램을 충분히 만들 수 있다. 대부분의 파이썬 프로그램들도 클래스 없이 작성되어 있다.

1장 파이썬 설치하기 참고

객체는 단순히 데이터 (변수)와 메소드(함수)의 모음, 클래스는 객체를 위한 청사진이다.

클래스는 집의 스케치 (원형)라고 생각할 수 있다. 여기에는 바닥, 문, 창문 등에 대한 모든 세부 정보가 포함되어 있습니다. 이러한 설명을 토대로 우리는 집을 짓게 되므로 하우스가 객체이다.

클래스는 프로그램의 원형으로 변수 정의, 함수 등으로 구성되어 있고 함수는 원하는 반복 작업을 위한 입력 ->실행 -> 외부 출력으로 이루어진 코드 묶음

함수와 모듈

함수는 하나의 관련 작업을 수행하는 데 사용되는 체계적이고 재사용 가능한 코드 블록이다. 함수는 응용 프로그램의 모듈성을 높이고 코드 재사용 수준을 높인다. 파이썬은 print () 등과 같은 많은 내장 built-in 함수를 제공하지만 사용자가 직접 함수를 만들 수도 있는데, 이를 사용자 정의 user-defined 함수라고 한다.

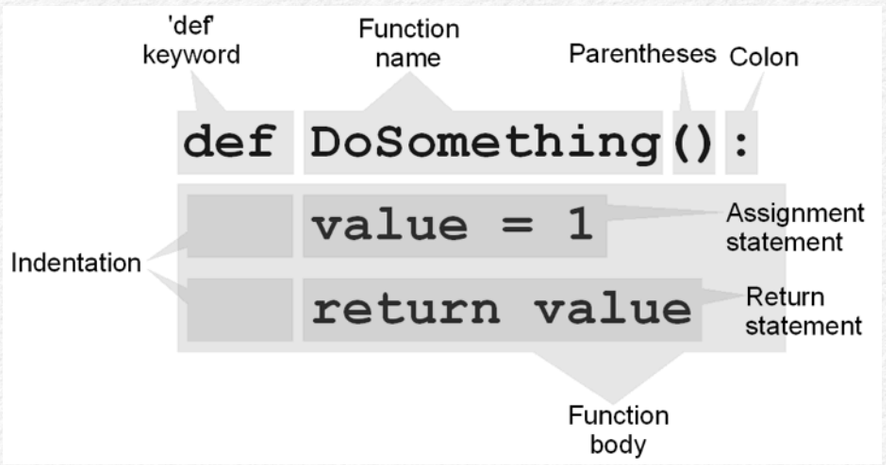
모듈을 사용하면 논리적으로 파이썬 코드를 구성 할 수 있습니다. 관련 코드를 모듈로 그룹화하면 코드를 더 쉽게 이해하고 사용할 수 있다. 모듈은 바인딩하고 참조 할 수있는 임의로 명명 된 속성을 가진 Python 객체이다. 모듈은 파이썬 코드로 구성 된 파일입니다. 모듈은 함수, 클래스 및 변수를 정의 할 수 있다. 모듈은 실행 가능한 코드를 포함 할 수도 있다.

함수

정의

```
def your_function_name(parameters):
        statements
        return expression
```

- 시작은 define의 약어인 def로 시작하고 함수명(입력값 지정, 생략 가능), 끄리고 콜론(:)으로 끝난다.
- 원하는 작업 실행이 가능한 코드를 적는다.
- 출력되는 함수 결과값을 지정한다.



예제 : 피보나치 수열 화면 출력하기

- print() 내의 end=' ' 옵션은 공백 하나와 함께 연속 출력을 의미한다. 없으면 행 출력이 된다.

In [6]:
```
def fib(n):    # write Fibonacci series up to n
    """Print a Fibonacci series up to n."""
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
fib(2000)
```

```
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

예제2 : 리스트에 저장하기

- result = [ ] : 저장할 변수 초기값 지정함, list 자료구조임
- result.append(a) : a값이 result에 추가 저장된다.

In [9]:
```
def fib2(n):  # return Fibonacci series up to n
    """Return a list containing the Fibonacci series up to n."""
    result = [ ]  #결과 저장변수 설정
    a, b = 0, 1
    while a < n:
        result.append(a)    # 피보나치 수열 추가
        a, b = b, a+b
    return result

fib2(2000)
```

Out[9]:
```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597]
```

예제3 : 모수 기본 인자값 지정

- 두 변수(retries, reminder)의 값은 지정되어 들어갔으므로 실제 입력 모수는 prompt 만 있음

In [10]:
```
def ask_ok(prompt, retries=4, reminder='Please try again!'):
    while True:
        ok = input(prompt)
        if ok in ('y', 'ye', 'yes'):
            return True
        if ok in ('n', 'no', 'nop', 'nope'):
            return False
        retries = retries - 1
        if retries < 0:
            raise ValueError('invalid user response')
        print(reminder)
```

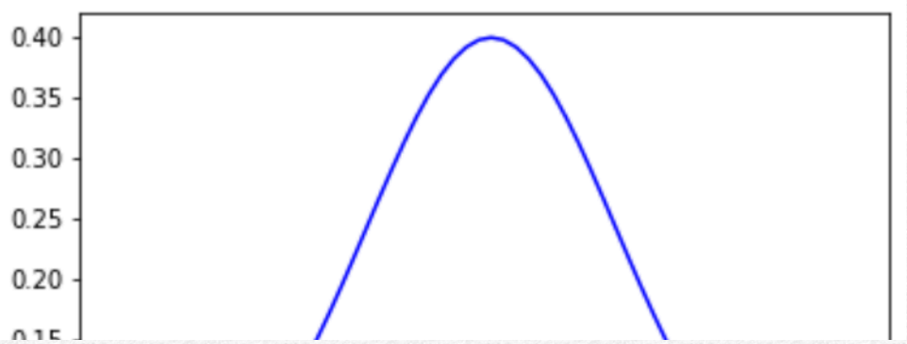
In [12]:
```
ask_ok('End??')
```
```
End??y
```
Out[12]: True

In [13]:
```
ask_ok('End??')
```
```
End??sss
Please try again!
End??n
```
Out[13]: False

In [16]:
```
ask_ok('Stop ??')
```
```
Stop ??s
Please try again!
Stop ??s
Please try again!
Stop ??s
Please try again!
Stop ??ss
Please try again!
Stop ??sss

---------------------------------------------------------------------------
ValueError
<ipython-input-16-
```

정규분포함수 그리기

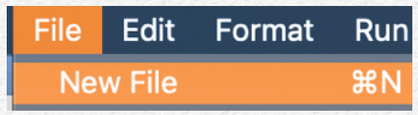
In [21]:
```
def normplt(m,s):
    import numpy as np
    import matplotlib.pyplot as plt
    from scipy.stats import norm
    x=np.arange(m-3*s,m+3*s,0.1)
    plt.plot(x, norm.pdf(x,m,s),'b')
    plt.show()
normplt(0,1)
```



사용자 정의 함수 모듈 만들기 & 사용하기

모듈 만들기 in IDLE

(1) 새 파일 열기

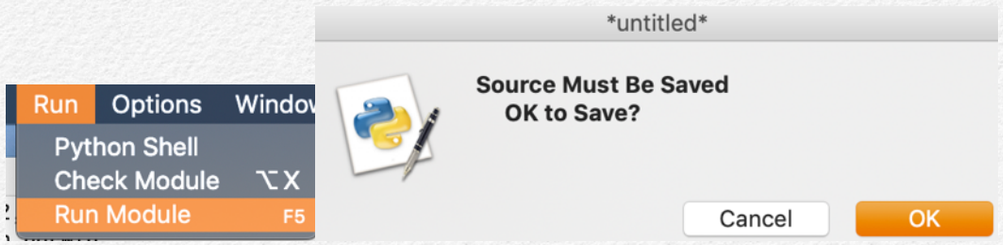


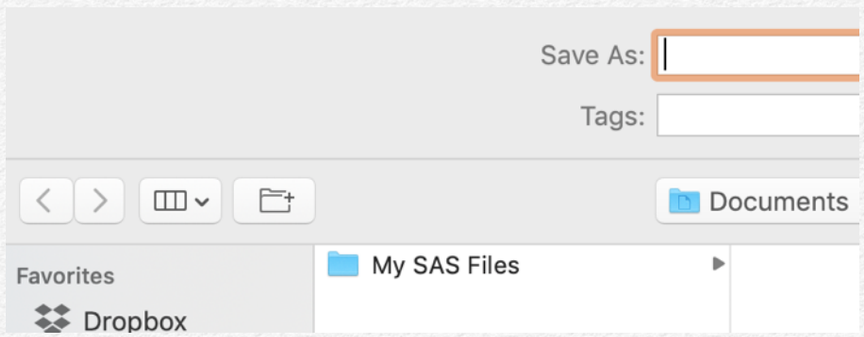
(2)

(3) 주피터 노트북 코드 ctrl+c(복사)하여 파일에 붙어 넣는다.

```
def normplt(m,s):
    import numpy as np
    import matplotlib.pyplot as plt
    from scipy.stats import norm
    x=np.arange(m-3*s,m+3*s,0.1)
    plt.plot(x, norm.pdf(x,m,s),'b')
    plt.show()
```

(4) Run Module 혹은 F5를 눌러 모듈로 저장한다. nplt.py로 저장

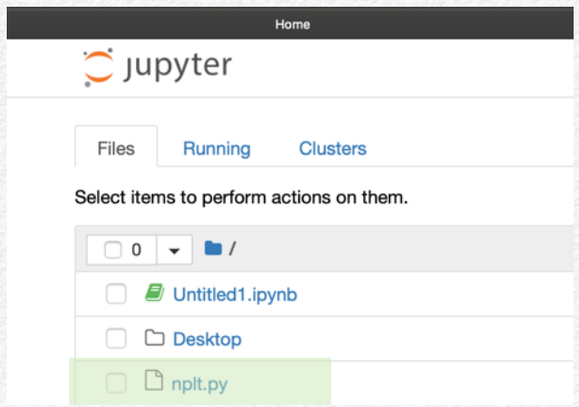




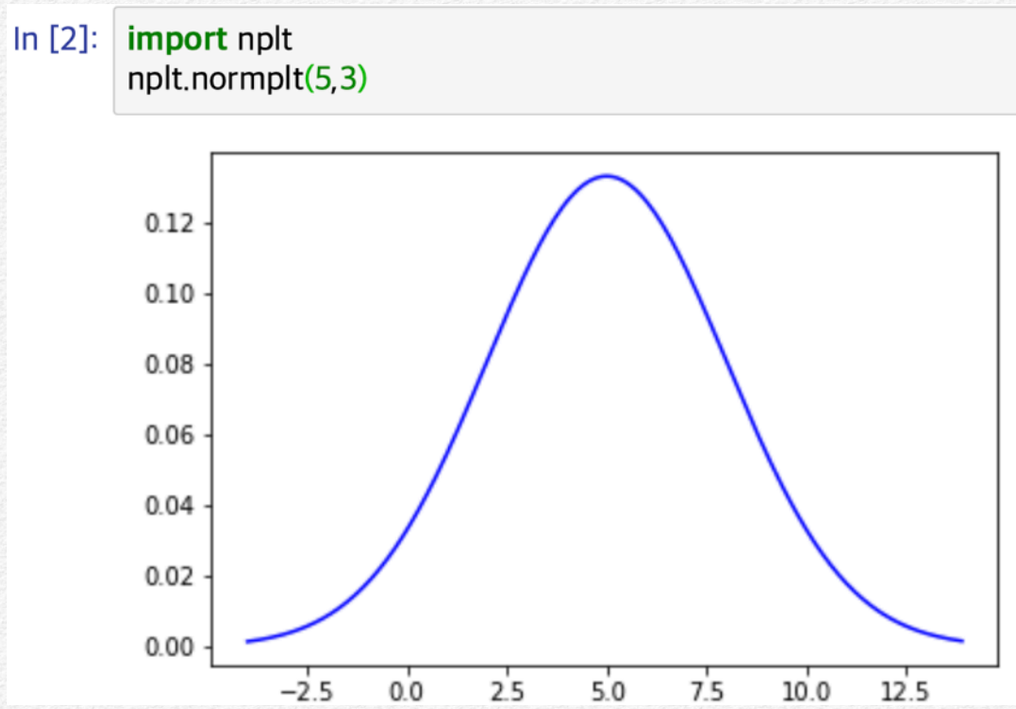
(5) 주피터 시작 폴더(Home 탭)에 nplt.ny로 저장하였다.

사용하기

(1) 주피터 노트북을 실행한다. 당연히 Home 탭에서 만들어 놓은 nplt.py 모듈이 있는지 확인된다.



(2) 사용자 정의 모듈 nplt import 하고 바로 함수 실행하면 결과를 얻을 수 있다.



모듈

개념

프로그램이 길어짐에 따라 디버깅과 유지를 쉽게 하려고 여러 개의 파일로 나누어 코딩하거나 각 프로그램에서 썼던 편리한 함수를 각 프로그램에 정의를 복사하지 않고도 사용하고 싶을 때 사용한다.

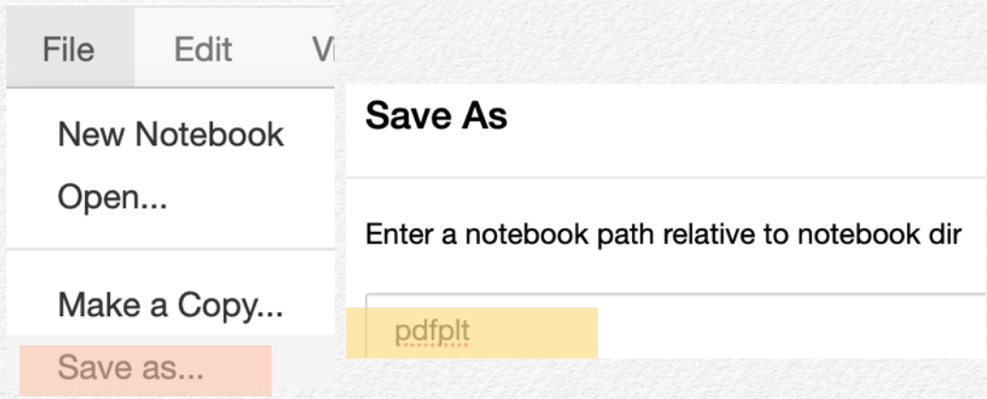
파이썬은 정의들을 파일에 넣고 스크립트나 인터프리터의 대화형 모드에서 사용할 수 있는 방법을 제공하는데 그런 파일을 모듈이라고 한다. 모듈은 파이썬 정의와 문장들을 담고 있는 파일로 확장자 .py 를 붙인다.

예제 : 2개 함수, 정규분포, 감마분포 PDF 그리는 함수

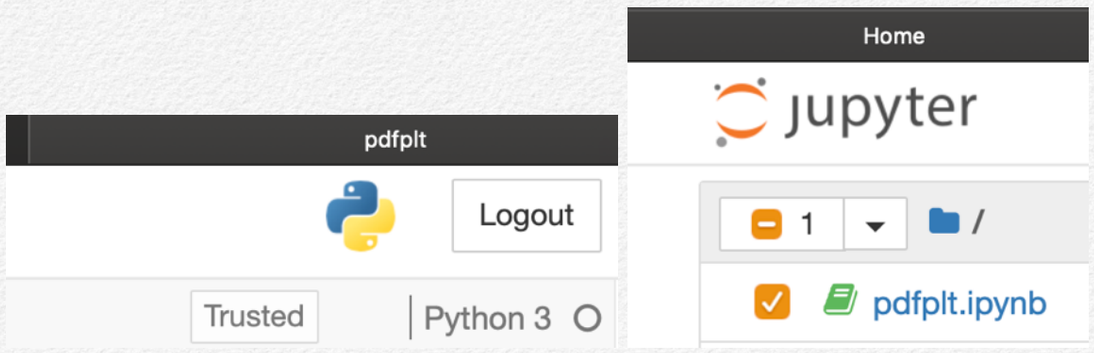
In [34]:
```
def normplt(m,s):
    import numpy as np
    import matplotlib.pyplot as plt
    from scipy.stats import norm
    x=np.arange(m-3*s,m+3*s,0.1)
    plt.plot(x, norm.pdf(x,m,s),'b')
    plt.title('Normal (Mean='  + str(m) + ', STD=' + str(s) +') PDF')
    plt.show()

def gamplt(a,b):
    import numpy as np
    import matplotlib.pyplot as plt
    from scipy.stats import gamma
    x=np.arange(0,a*b+6*a*b*b,0.1)
    plt.plot(x, gamma.pdf(x,a,b),'b')
    plt.title('Gamma (Alpha='  + str(a) + ',Beta=' + str(b) +') PDF')
    plt.show()
```

모듈 만들기

(1) 주피터 노트북에서 File => Save as 메뉴를 선택하여 폴더 지정 없이 바로 파일 이름을 적는다.

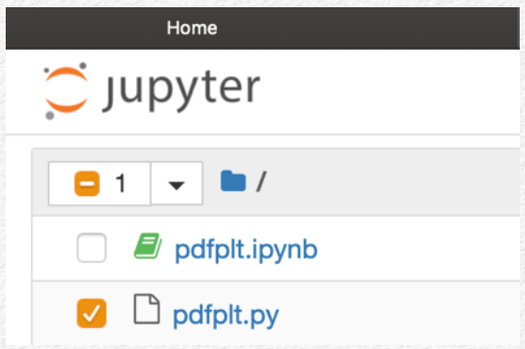


(2) 탭이름이 untitled=>에서 저장된 이름(pdfplt)으로 바뀌고 Home 폴더에는 pdfplt.ipynb 파일이 저장된다. (확장자 *.ipynb는 자동으로 부여된다)



(3) 주피터 코드 실행 후 In 라인에 !jupyter nbconvert --to python pdfplt.ipynb 입력하면 Home 폴더에 pdfplt.py 모듈이 만들어진다.

```
    plt.title('Gamma (Alpha='  + str(a) + ',Beta=' + str(b) +') PDF')
    plt.show()

In [37]: !jupyter nbconvert --to python pdfplt.ipynb

    [NbConvertApp] Converting notebook pdfplt.ipynb to python
    [NbConvertApp] Writing 590 bytes to pdfplt.py
```



(4) Home 폴더에 만들어진 모듈 pdfplt.py은 향후 계속 사용할 수 있다.

모듈 사용하기

from 모듈이름 import 함수이름1, 함수이름2, 이용하여 불러온다.

import matplotlib.pyplot as plt 는 모듈 내에 import하지만 모듈 내 함수 사용 전에 필요한 plt.subplt()이 있는 모듈이므로 사전에 불러와야 한다.

```
In [1]: from pdfplt import gamplt, normplt

In [7]: import matplotlib.pyplot as plt
        plt.subplot(2,1,1)
        gamplt(2,1)
        plt.subplot(2,1,1)
        normplt(2,1)
```