8_functions

- Define a function
  - function with specified information
- Passing Arguments
  - positional arguments
  - keyword arguments
  - default value
- return
  - return a simple value
  - optional argument
  - returns a dictionary
  - optional dictionary item
- passing a list
  - modify list
  - preventing function from modifying lists
- passing arbitrary number of arguments
  - mixing positional and arbitrary arguments
  - arbitrary keyword argument
- storing function in module
  - import entire module
  - import specific module
  - function nickname
  - import all functions in a module

functions are named blocks designed to do one specific job. we can call the function when we need to perform that particular task.

Define a function

we can define our own function for further usage

```
def hello_printer():
        """it prints welcoming messages to users"""
        user_name=input('pleae log in with your username')
        print(f'{user_name}, welcome!')
hello_printer()
# the first line defines the name of the function and you could specify the
```

```
information required in the parentheses.in the second line, we use triple quotes to enclose the docstring, which is the introduction of the function. Then we have the code of the function.
```

function with specified information

```
def hello_printer(user_name):
        """it prints greeting messages to users"""
        print(f'Hi,{user_name}')
hello_printer('Marcos')
#Hi,Marcos
# user_name in the parentheses is the parameter and the hello_printer('Marocs')is the argument
```

Passing Arguments

you have many ways to form you arguments to include the parameter and therefore call the function

positional arguments

```
# one way is to match the order in the parameter
def favorite_color(name, color):
        """it prints the person and its favorite color"""
        print(f"{name}'s favorite color is {color} " )
favorite_color('marcos','purple')
```

keyword arguments

```
#or we can just type out the keyword
def favorite_color(name, color):
        """it prints the person and its favorite color"""
        print(f"{name}'s favorite color is {color} " )
favorite_color(name='marcos',color='purple')
```

default value

```
#one can set up the default value for parameter so we can skip it when using the function
def petnamer(type='dog',name):
        """ it prints the pet's name"""
```

```
        print(f"my {pet}'s name is {name}")
petnamer(name='billy')
```

return

we don't have to print out the value directly. instead we could use return to deliver a value for our function

return a simple value

```
def namegenerator(first_name,last_name):
        """generate your full name for you"""
        fullname=f'{first_name} {last_name}'
        return fullname.title()
print(namegenerator(first_name='marcos',last_name='gao'))
#Marcos Gao
```

optional argument

```
def get_formatted_name(first_name, last_name, middle_name=''):
        """Return a full name, neatly formatted."""
        if middle_name:
                full_name = f"{first_name} {middle_name} {last_name}"
        else:
                full_name=f"{first_name} {last_name}"
        return full_name.title()

musician = get_formatted_name('john', 'hooker')
print(musician)
#John  Hooker
```

returns a dictionary

```
def people_data(name,age,gender):
        """it generates list for people"""
        people={'name':name,'age':age,'gender':gender}
        return people
print(people_data('marcos','20','male'))
```

optional dictionary item

```
def build_person(first_name, last_name, age=None):
        """Return a dictionary of information about a person."""
        person = {'first': first_name, 'last': last_name}
        if age:
                person['age'] = age
        return person
musician = build_person('jimi', 'hendrix', 27)
print(musician)
```

passing a list

```
def favorite_animal(animal):
        """returns favorite animal"""
        for creature in animal:
                print(f'one vote for{creature}')
vote_result=['cat','cat','alpaca','dog','panda','panda']
favorite_animal(vote_result)
```

modify list

```
def print_models(models_unprinted,models_printed):
        """move models unprinted to models printed."""
        while models_unprinted:
                model_printed=models_unprinted.pop()
                print(f'the {model_printed} is printing')
                models_unprinted.append(model_printed)
models_unprinted=['car','tank','bridge','clock']
models_printed=[]
print_models(models_unprinted,models_printed)
```

preventing function from modifying lists

```
# sometimes you may want to keep a copy of the list while functioning it.
def print_models(models_unprinted,models_printed):
        """move models unprinted to models printed."""
        while models_unprinted:
                model_printed=models_unprinted.pop()
                print(f'the {model_printed} is printing')
                models_printed.append(model_printed)
unprinted=['car','tank','bridge','clock']
printed=[]
```

```
print_models(unprinted[:],printed)
#we add a [:] in the last line so that all the manipulation are on the copy of the
list.
```

passing arbitrary number of arguments

```
#sometimes we don't know the exact number of arguments in advance
def coming_guests(*guests):
        """prints every guest's name"""
        print(f'The following guests will come:\n')
        for guest in guests:
                print(guest)
coming_guests('marcos','guao','niao','chan')
#The following guests will come:
#Marcos
#Guao
#Niao
#Chan
```

mixing positional and arbitrary arguments

```
# when mixing positional arguments and an arbitrary arguments, the later must be
placed last in the function to avoid confusion
def party_info(location,*guests):
        """prints the location and the coming guests"""
        print(f'this party is held in {location}, the coming guests includes:')
        for guest in guests:
                print(guest.title())
party_info('starbucks','marcos','ban','shan')
```

arbitrary keyword argument

```
#sometimes we don't know how many information, and also, what information is
required for the function

def party_info(first,last,**guests_info):
#'**'allows you to create an empty dicionary and modify it
        """introduce the information of the guests"""
        guests_info['first_name']=first
        guests_info['last_name']=last
        return guests_info
print(party_info('Marcos','Gao',nationality='china',mbti='ENTP'))
```

storing function in module

you don't have to list all your function and codes on the same page. instead, you could store functions on a separate page called module and import the module whenever you need. this helps you focus on the main logic of your work.

import entire module

first, you need to create a file ends in .py for your function.
then use import to tell python to copy the code from the module.

```
#pizza.py
def make_pizza(size, *toppings):
    """Summarize the pizza we are about to make."""
    print(f"\nMaking a {size}-inch pizza with the following toppings:")
    for topping in toppings:
        print(f"- {topping}")
```

```
#makepizza.py
import pizza

pizza.make_pizza(16, 'pepperoni')
pizza.make_pizza(12, 'mushrooms', 'green peppers', 'extra cheese')
```

import specific module

```
from pizza import make_pizza
```

function nickname

```
from pizza import make_pizza as mp
```

import all functions in a module

```
from pizza import *
```