~\algorithem project #3.py

```
def print_fruits_info(person_name, fruits):
    person_name = 'Max'
    fruits.pop()
    for fruit in fruits:
     print(f'{person_name} likes {fruit}')

my_name = 'Paul'
favorite_fruits = ['oranges', 'apples', 'bananas']

print_fruits_info(my_name, favorite_fruits)

print(favorite_fruits)
print(id(favorite_fruits))
print(my_name)

print('abc', 10 ,True)

def my_fn(first, second=True):
    print(first)
    print(second)

my_fn(100, None,)
my_fn(None)

def print_persons_background_info(person, race):
    persons_name = 'Josh'
    persons_race = ('white', 'slavic')
    person.copy = persons_name
    for persons_name in race :
        print(f'{person} is {persons_race}zip(persons_name, persons_race')
    print({'persons_name', 'persons_race'})
    return print(persons_name, persons_race)

def sort_nums(*args):
    args = sorted(args)
    for arg in args:
      return sorted(args)


sort_nums(10, 3, 15, 246, 23)

sorted_nums = sort_nums(10, 3, 15, 246, 23)
print(sorted_nums)

def comments_info(comments_qty, day):
    print(f'{comments_qty} were posted on {day}')


```

```
comments_info(comments_qty=50, day='Monday')

def product_price_info(**product):
    product_title = product['product_title']
    product_price = product['product_price']
    print(product)
    # print(f'{title} costs {price}$')

product_price_info(product_title = 'bottle of water', product_price=6)
def supply_info(supplies, supply_qty):

   print(f'{supplies}  {supply_qty}')
   my_fn = print([40, 10],[supplies] * [supply_qty])
   (args for args in my_fn**my_fn)
   def my_fn(supplies, supply_qty):
    #print(f'{supplies}  {supply_qty}')
    return print(supplies, supply_qty)

   def my_fn_info_dict(value, quantity): {[value, quantity]}
   print({'value : quantity'})
   my_fn = print({'value : quantity'})
   print(my_fn)

   res = my_fn_info_dict [20_30 : 40_50 * 60_70]
   print(res)
   return(res) == True

update_car_info_list = ('model, key = value', 'model, year, date') == ['Tesla Model 3, 2022, 10/10/2022']
print(update_car_info_list)

def send_email(to, subject, *args, **kwargs):
    print(f'Sending email to: with subject: {to}')
    print(f'Email subject: {subject}')

    for arg in args:
        print(arg)
    for kwarg in kwargs:
        print(kwargs)
    if arg:
        print("additonal recipients:")
        for recipient in arg:
            print(recipient)

    if kwargs:
        print('additional keywords:')
        for recipient in kwargs:
            print(recipient)
            for key in list(kwargs):
                print(f"{key}: {kwargs[key]}")
```

```
98              print({kwarg} for kwarg in kwargs)
99
100
101 send_email('test@test.com', 'Hows it going?' 'other@test.com' , 'someone@gmail.com' ,
    bcc='paul@gmail.com' , img = 'img.png')
102 from datetime import datetime
103
104
105 def get_weekday():
106             return datetime.today().strftime('%A')
107
108
109 def create_new_post(post, weekday= get_weekday()):
110             post_copy = post.copy()
111             post_copy['create_on_weekday'] = weekday
112             return post_copy
113
114
115 initial_post = {
116              'id': 243,
117              'author': 'Paul',
118         }
119
120 post_with_weekday = create_new_post(initial_post)
121
122 print(post_with_weekday)
123
124 def print_number_info(num):
125             """
126             Returns info regarding num, wheather num is even or odd
127
128             :param num: number to be evaluated
129             :returns: returns string which indicates if num is even or odd
130             """
131             if num % 2 == 0:
132                 return "numb is even"
133             else:
134                 return "numb is odd"
135
136
137             print(print_number_info(50))
138 def send_email(to, subject, *args, **kwargs):
139                 """
140                 Sends an email to different recipients
141
142
143                 :param str to: Recipient address
144                 :param str subject: Email Subject
145                 :param str args: Additional recipients
146                 :param str kwargs: Additional details
```

```
147                 :return None:
148                 """
149 
150                 print(f"Email subject: {subject}")
151 
152                 if args:
153                     print("additonal recipients:")
154                     for recipient in args:
155                         print(recipient)
156 
157                 if kwargs:
158                     print("additional keywords:")
159                     for recipient in kwargs:
160                         print(recipient)
161 
162                 print("Sending email to:", to)
163 
164 send_email('test@test.com', 'hello there!')
165 print('_______')
166 send_email('test@test.com', 'hello there!')
167 print('_______')
168 send_email('test@test.com', 'hello there!', bcc='paul@gmail.com' , img = 'img,png')
169 print('_______')
170 send_email('test@test.com', 'hello there!', 'other@test.com' , 'someone@gmail.com' ,
    bcc='paul@gmail.com' , img = 'img.png')
171 
172 c = True
173 
174 def mult(a, b):
175      a = 10
176      b = 20
177      c = a * b
178      return c
179 
180 
181 print(mult(100, 30))
182 
183 counter = 0
184 
185 
186 def inc_counter(value=1):
187     global counter
188     counter += 1
189 
190 
191     def inc_counter(value=1):
192         global counter
193         counter -= 1
194     def inc_counter(value=1):
195         """
```

```
        decrements value

        :param int value: increment counter by value
        """

    inc_counter(counter)
    print(inc_counter)


    inc_counter(5)
    print(counter)
    inc_counter(2)
    print(counter)
    inc_counter(7)
    print(counter)

    my_num = True

    print(my_num)
    print(+my_num)

    print(my_name)
    print(not not my_name)
    print(bool(my_name))

    print(bool(10))
    print(bool(3.6))
    print(bool(3j))

    print(bool(None))
    print(bool(False))

    print(bool(True))

    print(bool([1, 2]))
    print(bool({'a': 'abc'}))
    print(bool((3, 5)))
    print(bool({10, 20}))
    print(bool(range(100)))

    my_dict = {'name': 'John', 'age': 30}

    if len(my_dict):
        print("dict is not empty")
    if not my_dict:
        print("dict is empty")

    if my_dict['b']:
        print("key 'b' is in dict")

```

```
print('apple' or 'banana' or 'strawberry')

print('' or print('CALLED') or 'banana' or 'strawberry')

print('' or 0 or [] or {} or print('CALLED'))

print(0 and 'banana' and 'strawberry')

print(('apple' or 'banana') and 'strawberry')

my_list = [1,2]
other_list = ['a', 'b']

print(bool(my_list or other_list))

my_list = [1]

if my_list:
   print("list is not empty")
   print(...)

my_list and print("list is not empty")

a = [1, 2]
b = [1, 2]

print(id(a))
print(id(b))

print(a > b)
#a.__gt__
print(a >= b)
#a.__ge__
print(a > b)
#a.__lt__
print(a >= b)
#a.__le__
print(a < b)
print(a == b)
#a.__eq__
print(a != b)

name = 'Paul'

if len(name) > 4:
   print("name is longer than 4")

print(bool(name))

my_nums = [10, 3, 5, 20]
```

```
296
297 if my_nums == sorted(my_nums):
298    print("list is alread sorted!")
299
300 print(sorted(my_nums))
301
302 others_nums = [3.5, 5.10, 7.75, 10.05]
303
304 if others_nums == sorted(others_nums):
305    print("list is alread sorted!")
306
307 print(sorted(others_nums))
308
309 students = [
310     {'name': 'John', 'age': 50},
311     {'name': 'Paul', 'age': 30},
312     {'name': 'Jane', 'age': 20},
313 ]
314
315 students = True
316 print(students)
317
318 # def sort_by_score(x):
319     # return x['score']
320
321
322
323 # students = sorted(students, key=sort_by_score)
324 print(students)
325
326 # students := [30,20,50]
327
328 # sorted_students = students.sort()
329 # print(sorted_students)
330
331 my_nums = [3, 4, 5, 20, 15, 10, 15, 21]
332
333
334 print(list(filter(lambda num: num > 10, my_nums)))
335
336 print(list(filter(lambda num: num % 10 == 0, my_nums)))
337
338 print(10 % 2)
339 print(9 % 2)
340
341 odd_nums = list(filter(lambda num: num % 2, my_nums))
342 print('0dd numbers:', odd_nums)
343 #filter(nun: lambda, x: x > 10)
344 try:
345    salary = int(input("Enter Salary amount:"))
```

```
    days_qty = int(input("Ente days quality:"))
    salary_per_day = salary / days_qty
    print(salary_per_day)

except (ValueError, ZeroDivisionError) as e:
    if type(e) == ValueError:
        print(e)
        print("cannot convert value to int")
    if type(e) == ZeroDivisionError:
        print(e)
        print("cannot divide by zero")
    try:
        salary = int(input("Enter Salary amount:"))
        days_qty = int(input("Ente days quality:"))

        salary_per_day = salary / days_qty
        print(salary_per_day)

    except ValueError as e:
        print(e)
        print("cannot convert value to int")

    except ZeroDivisionError as e:
        print(e)
        print("cannot divide by zero")
    except Exception as e:
        print(type(e))
        print(e)
        print(isinstance(e, Exception))
        print(isinstance(e, ValueError))
        print(isinstance(e, ZeroDivisionError))
    try:
        salary = int(input("Enter Salary amount:"))
        days_qty = int(input("Ente days quality:"))
        salary_per_day = salary / days_qty
        print(salary_per_day)

    except ValueError as e:
        print(e)
        print("cannot convert value to int")

    except ZeroDivisionError as e:
        print(e)
        print("cannot divide by zero")
    except Exception as e:
        print(type(e))
        print(e)
        print(isinstance(e, Exception))
        print(isinstance(e, ValueError))
        print(isinstance(e, ZeroDivisionError))
```

```
396
397     try:
398         salary = int(input("Enter Salary amount:"))
399         days_qty = int(input("Enter days quality:"))
400         salary_per_day = salary / days_qty
401         print(salary_per_day)
402
403     except ValueError as e:
404         print(e)
405         print("cannot convert value to int")
406
407     except ZeroDivisionError as e:
408         print(e)
409         print("cannot divide by zero")
410     else:
411         print("Result, salary per day is: ", salary_per_day)
412
413     finally:
414         print('print salary operation calculation complete')
415     try:
416         file = open("file.txt", "r")
417
418     except: FileNotFoundError
419     print("file not found")
420
421 else:
422         print("file is ready found")
423 finally:
424         print("file operation complete")
425
426
427 print("file is ready for reading")
428 try:
429         file = open("file.txt", "r")
430
431 except: FileNotFoundError
432 print("file not found")
433
434
435 employee_info = ("Paul Gulko", 20, "Web Developer")
436
437 employee_name, employee_age, employee_position = employee_info
438
439 # employee_name = empplyee_info[0]
440 # employee_age = empplyee_info[1]
441 # employee_position = empplyee_info[2]
442
443 print(employee_name, employee_age, employee_position)
444
445 color = (225, 128, 10)
```

```
446
447 red, green, blue = color
448 print(red)
449 print(green)
450 print(blue)
451
452 red = 100
453 blue = 2000
454
455 color =(red, green, blue)
456 print(color)
457
458 user_credentials = [
459     ('admin1', '12345'),
460     ('user1', 'password'),
461     ('guest1', 'qwerty')
462 ]
463
464 admin1, user1, guest1 = user_credentials
465 print(user1)
466 print(guest1)
467 print(admin1)
468
469 user1_username = guest1_password = admin1
470 admin1_username = user1_password = guest1
471 guest1_username = guest1_password = user1
472
473 print(user1_username, user1_password)
474 print(guest1_username, guest1_password)
475 print(admin1_username, user1_password)
476
477 person = ('bob, 20')
478 name = ('name', 'age')
479 bob_20 = ('person + name', 'age')
480 print(bob_20)
481
482 school_grades = (80, 75, 35, 20, 90)
483
484 first, middle, *remaining = school_grades
485
486 print(first)
487 print(middle)
488 print(remaining)
489
490 comment = ("This is a great course", 'bob_202, 120, 4.7')
491
492 def calculate_rectangle_area(width, height):
493     return width * height
494
495 print(calculate_rectangle_area(10, 5))
```

```python

dimensions = [100, 20]

#area = calculate_rectangle_area(*dimensions[0], dimensions[1])
# print(area)

area = calculate_rectangle_area(*dimensions)
print(area)


def calculate_rectangle_area(width, height):
    return width * height

def setup_database_connection(hostname, port, username, password, database):
    print(hostname, port, username, password, database)
    return {'hostname': hostname, 'port': port, 'username': username, 'password': password, 'database': database}

connection_data = {
    'hostname': 'localhost',
    'port': 5432,
    'username': 'postgres',
    'password': 'password',
    'database': 'postgres'}




def create_user(username, password, email):
    # create user
    """
    Creates a user with the given username, password and email.

    Args:
        username (str): The username for the user
        password (str): The password for the user
        email (str): The email for the user

    Returns:
        dict: A dictionary containing the user details
    """
    return {'username': username, 'password': password, 'email': email}

user_details = {
    'username': 'bob_202',
    'email': 'u2F3c@example.com',
    'password': '12345'
}

created_user = create_user(**user_details)
```

```python
print(created_user)

person = {'name': 'john', 'age': 26}

#other_person = person.copy()
#other_person['age'] = 27

other_person = {
**person,
'age': 30
}

print(person)
print(other_person)

default_setting = {'theme': 'light', 'font_size': 16}
user_settings = {'theme': 'dark', 'font_size': 24}

merged_settings = {**default_setting, **user_settings}
print(merged_settings)

# merged_settings = {
#     **default_setting,
#     **user_settings,
#}

merge_setting = default_setting | user_settings
print(merged_settings)

for example in ("enter the persons: ID and name"):
    print(example)
    ID = 123435
    name = "[{Paul Gulko}]"

    user1 = ("ID, name: 123435", "Paul:Gulko")
    merged = True, 123435, "ID: name" # account("ID: name", "Paul:Gulko")

    print(merged)
print(True, 123435, "ID: name")
dict({"ID": 123435, "name": "Paul:Gulko"})

print(True, 123435, "ID: name") # value (True 123435 "ID: name")

weight = 10.5

def check_shipping_fee(weight):
    if weight <= 0:
       print("invalid weight. Weight must be greater than zero")

    if 0 < weight <= 5:
```

```
595         print("The shipping fee is 5$")
596 
597     elif weight <= 15:
598         print("The shipping fee is 15$")
599 
600     elif weight <= 10:
601         print("The shipping fee is 20$")
602     else:
603         print("The shipping fee is 5$")
604 
605 
606 check_shipping_fee(5)
607 
608 num = 20
609 def check_divisibility(num):
610  if num% 3 == 0 and num % 5 == 0:
611     print("number is divisible by both 3 and 5")
612  elif num % 3 == 0:
613     print("number is divisible only by 3")
614  elif num % 5 == 0:
615     print("number is divisible only by 5")
616  else:
617     print("number is not divisible by 3 or by 5")
618 check_divisibility(15) # num % {divider2} == 0 and num % {divider1}
619 check_divisibility(15)
620 
621 def calc_discounted_price(price: float, is_member: bool):
622     if is_member:
623         return price -price * 0.1 # 10% discount
624     else:
625         return price -price * 0.05 # 5% discount
626     #if is_member:
627       #discount = price * 0.1 # 10% discount
628     #else:
629       #discount = price * 0.05 # 5% discount
630     #return price - discount
631 
632 
633 res_price = calc_discounted_price(100.5, True)
634 print(res_price)
635 
636 def get_letter_grade(grade):
637     if grade >= 90:
638         return "A"
639     elif grade >= 80:
640         return "B"
641     elif grade >= 70:
642         return "C"
643     elif grade >= 60:
644         return "D"
```

```
    else:
        return "F"

from sys import getsizeof

squares_gen = (num * num for num in range(1, 100))

print(getsizeof(squares_gen))

print(type(squares_gen)) # <class 'generator'>

squares_list = [num * num for num in range(1, 100)]

print(type(squares_list)) # <class 'list'>

print(getsizeof(squares_list))

print(squares_list)

print(type(squares_list[0])) # <class 'int'>

print(getsizeof(squares_list[0]))

squares_gen = (num * num for num in range(100_000_000))

for num in squares_gen:
        print(num)
        if num == 100:
            break
print("START SECOND ITERATION")

def decorator(func):
    def wrapper(*args, **kwargs):
        print("START DECORATOR")
        func(*args, **kwargs)
        print("END DECORATOR")
    return wrapper@decorator

def check_user_auth(fn):
    def wrapper(*args, **kwargs):
        print("START WRAPPER")
        fn(*args, **kwargs)
        print("END WRAPPER")
    return wrapper
def do_sensitive_job():
    # Peform action only if user is authenticated
    pass

@check_user_auth
def do_sensitive_job():
```

```
    # Peform action only if user is authenticated
    pass

def log_function_call(fn):
    def wrapper(*args, **kwargs):
        print(f"Calling function {fn.__name__}")
        return fn(*args, **kwargs)
    return wrapper
print(f"Function arguments: {log_function_call(do_sensitive_job)}")

def mult_numbers(a,b):
    return a*b

print(mult_numbers(10,20))
@log_function_call
def mult_numbers(a,b):
    return a*b

print(mult_numbers(10,20))

@log_function_call
def add_numbers(a,b):
    return a + b

print(add_numbers(10,20))
print('')
print(add_numbers(a=10, b=20))
print('')
print(add_numbers(b=20, a=10))

def sum(a,b):
    return a + b

print(sum(5,2))

@log_function_call
def sum(a,b):
    return a + b

def validate_args(fn):
    def wrapper(*args, **kwargs):
        print("START WRAPPER")
        fn(*args, **kwargs)
        print("END WRAPPER")
    return wrapper

@validate_args
def sum(a,b):
    return a + b

```

```python
print(sum(5,2))

class Car:
    def move(self):
        print("The car is moving")
        return "The car is moving"

    def stop(self):
        print("The car is stopped")
        return "The car is stopped"


my_car = Car()
print(type(my_car))

my_car = Car()
print(type(my_car))
print(isinstance(my_car, Car))
print(isinstance(my_car, object))
print(dir(my_car))
print(my_car.__dict__)

my_car.move()
my_car.stop()

class User:
    def info(self, username, email):
        print(f"user {username} has email {email}")

first_user = User()
first_user.info("john", "u2F3c@example.com")

print(first_user.__dict__)

second_user = User()

class User:
    def info(self):
        print(dir(self))
        print(f"user has email")


    first_user = User()

    first_user.username = "john"
    first_user.email = "u2F3c@example.com"

class Post:
    def __init__(self, title):
        self.title = title
```

```python

first_post = Post("My first post")
second_post = Post("My second post")
same_post = first_post
first_post is same_post == first_post == second_post

[1,2,3].__add__([4,5,6])

class Post:
    def __init__(self, title):
        self.title = title

first_post = Post("My first post")
second_post = Post("My second post")
print(first_post.title)
print(second_post.title)
print(first_post.__dict__)

class Admin(User):
    def __init__(self, username, email):
        super().__init__(username, email)
        self.role = "admin"
        self.is_admin = True
        self.is_staff = True

admin = Admin("john", "u2F3c@example.com")
print(admin.__dict__)

# Encapsulation

class Email:
    def __init__(self, email):
        self.email = email

    @property
    def email(self):
        return self._email

    @email.setter
    def email(self, email):
        if "@" in email:
            self._email = email
        else:
            raise ValueError("Email is not valid")

    @email.deleter
    def email(self):
        self._email = None

email = Email("u2F3c@example.com")
```

```
845
846 class Email:
847     def __init__(self, sender, recipient, subject, body):
848         self.sender = sender
849         self.recipient = recipient
850         self.subject = subject
851         self.body = body
852
853     def send_email(self):
854         #print(f"Sending email to {self.recipient}")
855         pass
856
857     def read_email(self):
858         #print(f"Reading email from {self.sender}")
859         pass
860
861     def delete_email(self):
862         #print(f"Deleting email from {self.sender}")
863         pass
864 email = Email("u2F3c@example.com", "u2F3c@example.com", "subject", "body")
865 email.send_email()
866 email.read_email()
867 email.delete_email()
868
869 forum = Forum()
870
871 forum.register_user("john", "u2F3c@example.com")
872
873 print(form.users)
874
875 forum.create_post("My first post", "Post content", Paul)
876
877 print(Paul.posts)
878 print(forum.posts)
879 print(forum.posts[0].title)
880 print(forum.posts[0].content)
881 print(forum.posts[0].author)
882 print(forum.posts[0].author.username)
883 print(forum.posts[0].author.email)
884 # Composition
885 # Aggregation
886 # Abstraction)
887 # Inheritance
888 # Polymorphism
889
890 class Shape:
891     def __init__(self, width, height):
892         self.width = width
893         self.height = height
894
```

```python
    def area(self):
        def cal_area(width, height):
            return width * height
        return cal_area(self.width, self.height)
    pass

class Rectangle(Shape):
    def __init__(self, width, height):
        super().__init__(width, height)

    def area(self):
        return super().area()

    def perimeter(self):
        return 2 * (self.width + self.height)

rectangle = Rectangle(10, 20)
print(rectangle.area())
print(rectangle.perimeter())

class Circle(Shape):
    def __init__(self, radius):
        super().__init__(radius, radius)

    def area(self):
        return super().area() * 3.14

circle = Circle(10)
print(circle.area())

#abstraction

class payment:
    def process(self):
        pass

class CreditCardPayment(payment):
    pass

class CashPayment(payment):
    pass

class BankTransferPayment(payment):
    pass

class PaymentProcessor:
    def __init__(self, payment):
        self.payment = payment
    def process_payment(self):
        pass
```

```
945         self.payment.process()
946         pass
947 
948 payment_processor = PaymentProcessor(CashPayment())
949 
950 import utils
951 
952 print(utils)
953 print(type(utils))
954 print(dir(utils))
955 
956 print(utils.add(1, 2))
957 print(u.hello(utils.my_name))
958 
959 from utils import add
960 print(add(1, 2))
961 
962 from utils import *
963 print(add(1, 2))
964 print(my_name)
965 hello('paul')
966 
967 import utils
968 print(utils.add(1, 2))
969 print(utils.my_name)
970 print(utils.hello('paul'))
971 
972 from utils import add
973 print(add(1, 2))
974 print(my_name)
975 print(hello('paul'))
976 
977 import utils as u
978 print(u.add(1, 2))
979 print(u.my_name)
980 print(u.hello('paul'))
981 
982 def sum(a, b):
983     return a + b
984 
985 def mult(a, b):
986     return a * b
987 
988 def sub(a, b):
989     return a - b
990 
991 def div(a, b):
992     return a / b
993 
994 def mod(a, b):
```

```
995     return a % b
996 
997 def pow(a, b):
998     return a ** b
999 
1000 from math import pi
1001 print(math.pow(2, 3))
1002 
1003 print(dir(math))
1004 
1005 print(math.pi)
1006 print(math.e)
1007 
1008 print(type(__name__))
1009 
1010 __name__ == '__main__'
1011 
1012 if __name__ == '__main__':
1013     print('hello')
1014     print('hello')
1015     print('hello')
1016     print('hello')
1017 print("MODULE")
1018 
1019 # print(dir(math))
1020 # print(math.pi)
1021 # print(math.e)
1022 # print(math.pow(2, 3))
1023 
1024 print('MAIN', __name__)
1025 print('MAIN', __name__=='__main__')
1026 
1027 import json
1028 
1029 my_nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
1030 print(json.dumps(my_nums))
1031 json.dumps(my_nums)
1032 
1033 {'1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9, '10': 10}
1034 
1035 my_post_json = '{"title": "My first post", "content": "Post content", "author": "Paul"}'
1036 
1037 {'title': 'My first post', 'content': 'Post content', 'author': 'Paul'}
1038 
1039 print(json.loads(my_post_json))
1040 my_post_dict = json.loads(my_post_json)
1041 print(my_post_dict)
1042 print(type(my_post_dict))
1043 
1044 import json
```

```
1045
1046 my_dict = {'title': 'My first post', 'content': 'Post content', 'author': 'Paul'}
1047 print(json.dumps(my_dict))
1048
1049 my_dict = {
1050     'title': 'My first post',
1051     'content': 'Post content',
1052     'author': 'Paul',
1053     'views': 100
1054 }
1055
1056 print(json.dumps(my_dict))
1057
1058 my_dict = {
1059     'title': 'My first post',
1060     'content': 'Post content',
1061     'author': 'Paul',
1062     'views': 100
1063 }
1064
1065 import os
1066
1067 directory_path = "my_test_directory"
1068 os.mkdir(directory_path)
1069
1070 os.rmdir(directory_path)
1071
1072 import os
1073
1074 directory_path = "my_test_directory"
1075 os.mkdir(directory_path)
1076 os.rmdir(directory_path)
1077 if not os.path.exists(directory_path):
1078     os.mkdir(directory_path)
1079     print("Directory created")
1080 else:
1081     print("Directory already exists")
1082
1083 from zipfile import ZipFile
1084 with ZipFile('my_file.zip', 'w') as my_zip:
1085     my_zip.write('hello.txt')
1086
1087 with ZipFile('my_file.zip', 'r') as my_zip:
1088     with open('file from zip', 'w') as f:
1089         f.write(my_zip.read('hello.txt'))
1090
1091 with ZipFile('my_file.zip', 'r') as my_zip:
1092     with open('my-file.txt', 'w') as f:
1093         f.write(my_zip.read('hello.txt'))
1094
```

```
1095 import csv
1096
1097 with open('my_file.csv', 'w') as my_file:
1098     writer = csv.writer(my_file)
1099     writer.writerow(['Title', 'Content', 'Author'])
1100     writer.writerow(['My first post', 'Post content', 'Paul'])
1101     writer.writerow(['My second post', 'Post content', 'Paul'])
1102
1103 with open('my_file.csv', 'r') as my_file:
1104     reader = csv.reader(my_file)
1105     for row in reader:
1106         print(row)
1107
1108 with open('my_file.csv', 'r') as my_file:
1109     reader = csv.DictReader(my_file)
1110     for row in reader:
1111         print(row)
1112
1113 with open('my_file.csv', 'r') as my_file:
1114     reader = csv.DictReader(my_file)
1115     for row in reader:
1116         print(row['Title'])
1117         print(row['Content'])
1118         print(row['Author'])
1119
1120 from datetime import datetime
1121
1122 future_datetime = datetime(2023, 1, 1, 0, 0, 0)
1123 current_datetime = datetime.now()
1124
1125 print(future_datetime - current_datetime)
1126 print("future date and time:", future_datetime)
1127 print("current date and time:", current_datetime)
1128
1129 import random
1130
1131 print(random.randint(0, 10))
1132 print(random.randint(0, 10))
1133 print(random.randint(0, 10))
1134 print(random.randint(0, 10))
1135
1136 #float random
1137 print(random.random())
1138
1139 #int random
1140 print(random.randint(0, 10))
1141
1142 #shuffle
1143 my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
1144 random.shuffle(my_list)
```

```
print(my_list)
random.shuffle(my_list)
print(my_list)
random.shuffle(my_list)

import math

print(math.pi)
print(math.e)
print(math.pow(2, 3))
print(math.sqrt(9))

def cal_factorial(num: int):
    if num == 1:
        return 1
    return num * cal_factorial(num - 1)

print(cal_factorial(5))
print(cal_factorial(10))
print(cal_factorial(20))

def cal_recursive_factorial(num: int):
    if num == 1:
        return 1
    return num * cal_recursive_factorial(num - 1)

print(cal_recursive_factorial(5))
print(cal_recursive_factorial(10))
print(cal_recursive_factorial(20))

import re

my_string = "My name is Paul"
print(re.findall("Paul", my_string))
print(re.search("Paul", my_string))
print(re.search("Paul", my_string).group())
print(re.search("Paul", my_string).span())

res = re.search("Paul", my_string)
print(res.group())
print(res.span())

my_string = "My name is Paul"
print(re.sub("Paul", "Gulko", my_string))
print(re.sub("Paul", "Gulko", my_string, 1))

import sqlite3

conn = sqlite3.connect('my_database.db')
cursor = conn.cursor()
```

```
cursor.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, username TEXT, email TEXT, password TEXT)")
conn.commit()
conn.close()

DB_name = 'sqlite.db'
conn = sqlite3.connect(DB_name)
cursor = conn.cursor()
cursor.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, username TEXT, email TEXT, password TEXT)")
conn.commit()
conn.close()

from array import array

array('i', [1, 2, 3, 4, 5])

my_int_array = array('i', [1, 2, 3, 4, 5])
print(my_int_array)
my_int_array.append(6)
print(my_int_array)

my_int_array = array('i', [1, 2, 3, 4, 5])
print(my_int_array)

verify_ssl = False
if verify_ssl:
          ssl._create_default_https_context = ssl._create_unverified_context
def get_python_version():
    python_version = "3.9"
    python_full_version = "3.9.6"

    # print(f"Python version: {python_version}")
    # print(f"Python full version: {python_full_version}")

    #SECURITY WARNING: don't run with debug turned on in production!
    DEBUG = True
    print(f"DEBUG: {DEBUG}")
    print(f"verify_ssl: {verify_ssl}")
    print(f"get_python_version: {get_python_version()}")
    # application definition

    INSTALLED_APPS = [
        'django.contrib.admin',
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
        'django.contrib.staticfiles',
    ]
```

```
1243
1244    MIDDLEWARE = [
1245        'django.middleware.security.SecurityMiddleware',
1246        'django.contrib.sessions.middleware.SessionMiddleware',
1247        'django.middleware.common.CommonMiddleware',
1248        'django.middleware.csrf.CsrfViewMiddleware',
1249        'django.contrib.auth.middleware.AuthenticationMiddleware',
1250        'django.contrib.messages.middleware.MessageMiddleware',
1251        'django.middleware.clickjacking.XFrameOptionsMiddleware',
1252    ]
1253
1254 from django.db import models
1255
1256 # Create your models here.
1257 class User(models.Model):
1258     username = models.CharField(max_length=100)
1259     email = models.EmailField()
1260     password = models.CharField(max_length=100)
1261
1262 from djano.shortcuts import render
1263
1264 def index(request):
1265     return render(request, 'index.html')
1266 from django.shortcuts import render
1267 from .models import User
1268 from django.http import HttpResponse
1269
1270 def index(request):
1271
1272     users = User.objects.all()
1273     return render(request, 'index.html', {'users': users})
1274
1275 <!DOCTYPE html>
1276 <html>language="en">
1277 <head>
1278     <meta charset="UTF-8">
1279     <title>Users</title>
1280 </head>
1281 <body>
1282     <h1>Users</h1>
1283     <ul>
1284         {% for user in users %}
1285         <li>{{ user.username }} - {{ user.email }}</li>
1286         {% endfor %}
1287     </ul>
1288 </body>
1289 </html>
1290
1291 from django.shortcuts import render
1292 from .models import User
```

```
from django.http import HttpResponse

def index(request):
    users = User.objects.all()
    return render(request, 'index.html', {'users': users})
urlpatterns = [
    path('', index),
    path('users/', index),
    path('users/<int:user_id>/', index),
]

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'tastypie',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

from django.db import models

class User(models.Model):
    username = models.CharField(max_length=100)
    email = models.EmailField()
    password = models.CharField(max_length=100)
    api = api(api_name='api')

    class Meta:
        app_label = 'tastypie'
    api.register(categoryresource(User))
    api.register(useresource(User))

    # /api/users/
    def get_list(self, request, **kwargs):
    # /api/cources/1/
    def get_object(self, request, **kwargs):
```

```
    # /api/useresource/1/
    def get_object(self, request, **kwargs):
    # /api/useresource/1/
    def get_object(self, request, **kwargs):
    # /api/categoryresource/1/
    def get_object(self, request, **kwargs):
    # /api/categoryresource/1/
    def get_object(self, request, **kwargs):
    # /api/useresource/1/
    def get_object(self, request, **kwargs):
    # /api/users/2/
    def get_object(self, request, **kwargs):

    import pygame

    pygame.init()
    pygame.mixer.init()
    pygame.display.init()
    pygame.font.init()
    pygame.image.init()
    pygame.mixer.init()
    pygame.time.init()
    pygame.display.init()
    pygame.event.init()
    pygame.mouse.init()
    pygame.key.init()
    pygame.display.set_mode((800, 600)

from django.db import models
from django.db.models import Q
from django.shortcuts import get_object_or_404
from django.shortcuts import render

def index(request):
    users = User.objects.all()
    return render(request, 'index.html', {'users': users})

class Fighter:

    def __init__(self, x, y, image, speed):
        self.image = pygame.image.load(image)
        self.x = x
        self.y = y
        self.speed = speed

    def draw(self, window):
        window.blit(self.image, (self.x, self.y))

    def move_right(self):
        self.x += self.speed
```

```
1393
1394        import numpy as np
1395
1396        def move_left(self):
1397            self.x -= self.speed
1398
1399        def move_up(self):
1400            self.y -= self.speed
1401
1402        def move_down(self):
1403            self.y += self.speed
1404
1405        def collide(self, other):
1406            return self.x == other.x and self.y == other.y
1407
1408            second = other
1409            if self.collide(second):
1410                return True
1411            else:
1412                return False
1413
1414            array('i', [1, 2, 3, 4, 5])
1415
1416            print(array('i', [1, 2, 3, 4, 5]))
1417
1418            first = array('i', [1, 2, 3, 4, 5])
1419            second = array('i', [1, 2, 3, 4, 5])
1420            if first == second:
1421                return True
1422            else:
1423                return False
1424
1425            first = array('i', [1, 2, 3, 4, 5])
1426            second = array('i', [1, 2, 3, 4, 5])
1427            if first == second:
1428                return True
1429            else:
1430                return False
1431
1432                prices = np.array([1, 2, 3, 4, 5])
1433                prices[0] = 10
1434                print(prices)
1435
1436                prices = np.array([1, 2, 3, 4, 5])
1437                print(prices[0])
1438                qualities = np.array([1, 2, 3, 4, 5])
1439                print(qualities[0])
1440                prices * qualities
1441                prices + qualities
1442                prices - qualities
```

```
            prices / qualities
            prices // qualities
            prices ** qualities
            prices % qualities
            np.sqrt(prices)
            np.log(prices)
            np.exp(prices)
            ###examples
            np.array([1, 2, 3, 4, 5]) * 2
            np.array([1, 2, 3, 4, 5]) + 2
            np.array([1, 2, 3, 4, 5]) - 2
            np.array([1, 2, 3, 4, 5]) / 2
            np.array([1, 2, 3, 4, 5]) // 2
            np.array([1, 2, 3, 4, 5]) ** 2
            np.array([1, 2, 3, 4, 5]) % 2
            np.array([1, 2, 3, 4, 5]).sqrt()

            import pandas as pd

            prices = pd.Series([1, 2, 3, 4, 5])
            print(prices)

            forum_users = pd.Series([1, 2, 3, 4, 5])
            print(forum_users)

            forum_users = pd.Series([1, 2, 3, 4, 5])
            print(forum_users[0])

            import numpy as np
            prices = np.array([1, 2, 3, 4, 5])
            print(prices[0])

            prices = np.array([1, 2, 3, 4, 5])
            print(prices[0])

            import pandas as pd
            prices = pd.Series([1, 2, 3, 4, 5])
            print(prices[0])

            prices = pd.Series([1, 2, 3, 4, 5])
            print(prices[0])

            prices = pd.Series([1, 2, 3, 4, 5])
            print(prices[0])

            df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6], 'C': [7, 8, 9]})
            print(df)
            df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6], 'C': [7, 8, 9]})
            print(df['A'])
            df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6], 'C': [7, 8, 9]})
```

```
1493
1494            forum_users = pd.Series([1, 2, 3, 4, 5])
1495            print(forum_users[0])
1496            forum_users = pd.Series([1, 2, 3, 4, 5])
1497            print(forum_users[0])
1498
1499            df.dtypes
1500            df.head(10)
1501            df.tail(10)
1502            df.describe()
1503            df['A'].describe()
1504            df['A'].head(10)
1505            df['A'].tail(10)
1506            df['A'].describe()
1507            df.to_excel('output.xlsx')
1508            df.to_csv('output.csv')
1509            df.to_json('output.json')
1510            df.to_html('output.html')
1511
1512            df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6], 'C': [7, 8, 9]})
1513            df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6], 'C': [7, 8, 9]})
1514
1515            forum_users = pd.Series([1, 2, 3, 4, 5])
1516            print(forum_users[0])
1517            forum_users = pd.Series([1, 2, 3, 4, 5])
1518            print(forum_users[0])
1519
1520            df.dtypes
1521            df.head(10)
1522            df.tail(10)
1523            df.describe()
1524            df['A'].describe()
1525            df['A'].head(10)
1526            df['A'].tail(10)
1527            df['A'].describe()
1528            df.to_excel('output.xlsx')
1529            df.to_csv('output.csv')
1530            df.to_json('output.json')
1531            df.to_html('output.html')
1532
1533            plt.title('Simple Plot')
1534            plt.xlabel('X axis')
1535            plt.ylabel('Y axis')
1536            plt.plot([1, 2, 3, 4, 5], [1, 4, 9, 16, 25])
1537            plt.show()
1538            plt.title('Simple Plot')
1539
1540            import pandas as pd
1541            df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6], 'C': [7, 8, 9]})
1542
```

```
            forum_users = pd.Series([1, 2, 3, 4, 5])
            print(forum_users[0])
            forum_users = pd.Series([1, 2, 3, 4, 5])
            print(forum_users[0])

            df.dtypes
            df.head(10)
            df.tail(10)
            df.describe()
            df['A'].describe()
            df['A'].head(10)
            df['A'].tail(10)
            df['A'].describe()
            df.to_excel('output.xlsx')
            df.to_csv('output.csv')
            df.to_json('output.json')
            df.to_html('output.html')

            plt.title('Simple Plot')
            plt.xlabel('X axis')
            plt.ylabel('Y axis')
            plt.plot([1, 2, 3, 4, 5], [1, 4, 9, 16, 25])
            plt.show()
            plt.title('Simple Plot')

            import pandas as pd
            df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6], 'C': [7, 8, 9]})

            forum_users = pd.Series([1, 2, 3, 4, 5])
            print(forum_users[0])
            forum_users = pd.Series([1, 2, 3, 4, 5])
            print(forum_users[0])

            df.dtypes
            df.head(10)
            df.tail(10)
            df.describe()
            df['A'].describe()
            df['A'].head(10)
            df['A'].tail(10)
            df['A'].describe()
            df.to_excel('output.xlsx')
            df.to_csv('output.csv')
            df.to_json('output.json')
            df.to_html('output.html')

            plt.title('Simple Plot')
            plt.xlabel('X axis')
            plt.ylabel('Y axis')

```

```
1593            import mathplotlib.pyplot as plt
1594            plt.title('Simple Plot')
1595            plt.xlabel('X axis')
1596            plt.ylabel('Y axis')
1597            plt.plot([1, 2, 3, 4, 5], [1, 4, 9, 16, 25])
1598            plt.show()
1599
1600            import seaborn as sns
1601            sns.set(style="darkgrid")
1602            tips = sns.load_dataset("tips")
1603            print(tips.head())
1604
1605            df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6], 'C': [7, 8, 9]})
1606
1607            forum_users = pd.Series([1, 2, 3, 4, 5])
1608            print(forum_users[0])
1609            forum_users = pd.Series([1, 2, 3, 4, 5])
1610            print(forum_users[0])
1611
1612            df.dtypes
1613            df.head(10)
1614            df.tail(10)
1615            df.describe()
```