Pemrograman Berorientasi Objek

muhamad.soleh@iti.ac.id

Konsep Atribut dan Fungsi

Konsep Kelas & Objek

01 Konsep Atribut

02 Konsep Fungsi

03 Implementasi Konsep Atribut dan Fungsi

04 Studi Kasus Konsep Atribut dan Fungsi

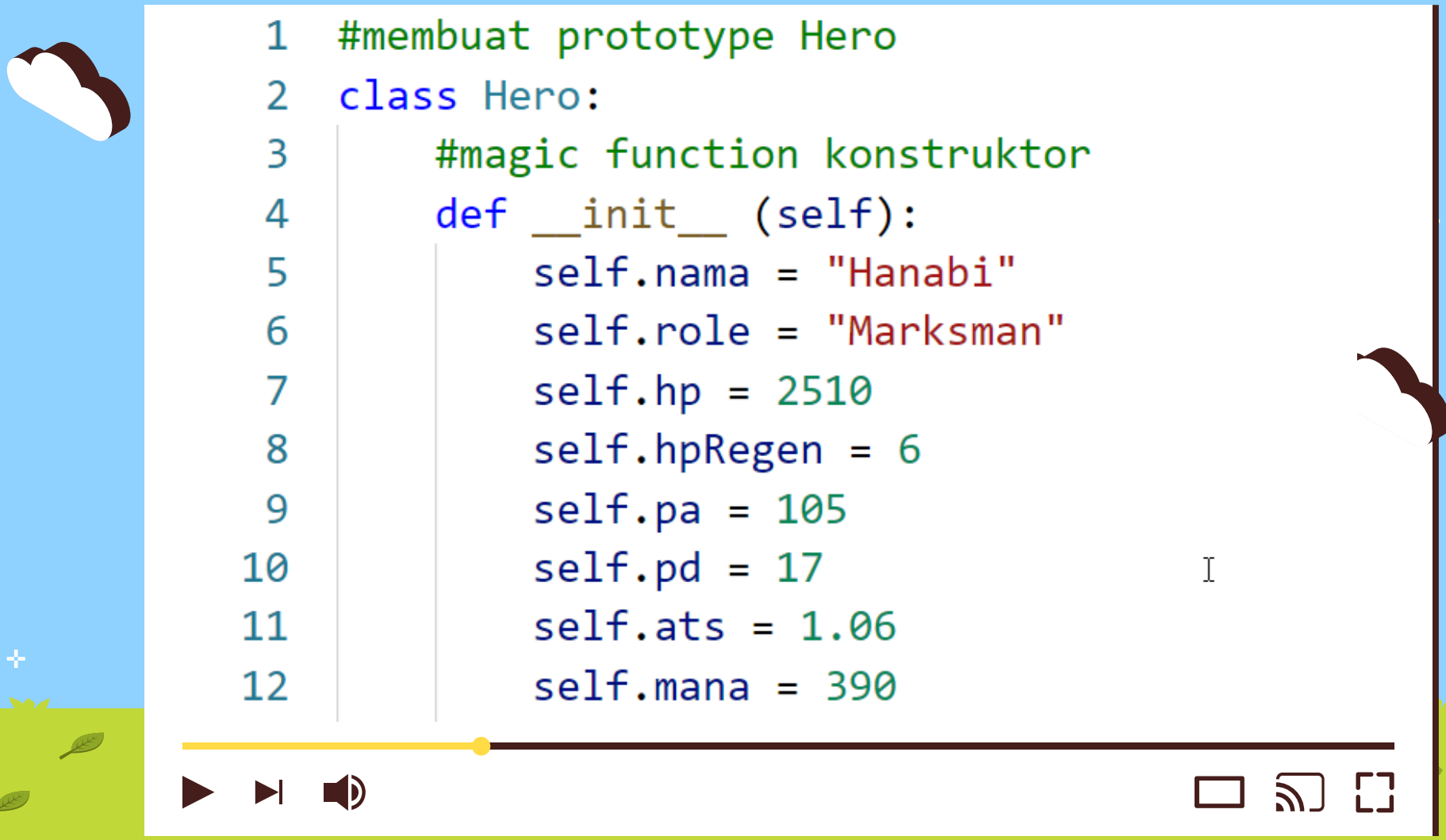
01

Konsep Atribut

Pemrograman Berorientasi Objek

Home Game Tools Facts Team

Konsep Atribut


#membuat prototype Hero
class Hero:
#magic function konstruktor
def __init__ (self):
self.nama = "Hanabi"
self.role = "Marksman"
self.hp = 2510
self.hpRegen = 6
self.pa = 105
self.pd = 17
self.ats = 1.06
self.mana = 390

```
        self.mana = 390
        self.manaRegen = 15
        self.mp = 0
        self.md = 15
        self.ms = 245
#membuat objek dari kelas Hero
hanabi = Hero()
print(hanabi.hp) #2510
hanabi.skin = "Rekshesha"
hanabi.cv = "Mochizuki Yu"
hanabi.specialty = "Regen / Reap"
print(hanabi.skin) #Rekshesha
```

Konsep Atribut

- Atribut merupakan *Ciri / datafield / variable / data member* yang melekat pada suatu objek.
- Atribut merupakan karakteristik yang membedakan antara satu objek dengan objek yang lainnya.
- Atribut memiliki 2 komponen, yaitu nama atribut dan data atribut.
- 2 cara membangkitkan Atribut:
  - Menggunakan magic function kontruktor pada kelas
  - Menggunakan Operator dot (.) / titik pada objek

Detail Objek Hanabi

Detail Objek Hanabi

```python
#membuat prototype Hero
class Hero:
    #magic function konstruktor
    def __init__ (self, n, r, h, hr, pa, pd, \
        a, m, mr, mp, md, ms):
        #("Hanabi","Marksman" ,2510 , 6, 105, 17,
        # 1.06, 390, 15, 0, 15, 245)
        self.nama = n
        self.role = r
        self.hp = h
        self.hpRegen = hr
        self.phat = pa
        self.phdef = pd
        self.ats = a
        self.mana = m
```

```
        self.mana = m
        self.manaRegen = mr
        self.map = mp
        self.mad = md
        self.mos = ms

#membuat objek pertama dari kelas Hero
hanabi = Hero("Hanabi","Marksman" ,2510 , 6, 105, 17, \
    1.06, 390, 15, 0, 15, 245)
print("HP mula-mula hanabi adalah: " , hanabi.hp) #2510

#membuat objek kedua dari kelas Hero
uranus = Hero("Uranus","Tank" ,2489 , 6.4, 115, 14, \
    1.04, 455, 12, 0, 15, 260)
print("Role heru uranus adalah : " , uranus.role) #Tank
```


8
07:54
7
15
10
0
0
2
62
Computer
Computer
Computer
8
8
7
Aster Demon
9
35
BUFF
Recall
Regen
Retribution
AoE

02

Konsep Fungsi

Pemrograman Berorientasi Objek

Home Game Tools Facts Team

Konsep Fungsi

Konsep Fungsi

- **Fungsi** merupakan suatu Aksi atau kelakuan */ method /* *behavior* yang dilakukan oleh objek
- **Fungsi** adalah grup/blok program untuk melakukan tugas tertentu yang dapat dilakukan secara berulang.
- **Fungsi** membuat kode program menjadi *reusable*, artinya hanya di definisikan sekali saja, dan kemudian bisa digunakan berulang kali dari tempat lain di dalam program.

Konsep Fungsi

- **Fungsi** terdiri dari 2 jenis:
  - Fungsi Return: digunakan **untuk mengembalikan suatu nilai dari fungsi.**
  - Fungsi Void: Disebut void karena fungsi tersebut tidak boleh memiliki return value
    - atau tidak mengembalikan suatu nilai keluaran yang didapat dari hasil proses fungsi tersebut.

Sintaks membuat fungsi return dan void dalam PBO

Pemrograman Berorientasi Objek

Home Game Tools Facts Team

Membuat fungsi return OOP menggunakan python

Detail Objek Hanabi

❖ Fungsi:

❖ Skill 1 : Petal Barrage

Cooldown: 0

Mana cost: 20

Hanabi menggunakan "*mana*" miliknya untuk melepaskan kekuatan "*flower demon*" yang terkunci dalam senjata terlarang Higanbana. Hanabi akan menembakkan "*petal blade*" ke arah lawannya.

Setiap *basic attack*, *blade* atau mata pisau baru akan muncul dan memberikan 30 persen *physical damage* kepada lawan. Hanabi menembakkan tiga *petal* atau daun bunga yang menyerang secara area (damage yang dihasilkan saat *petal* bergerak balik akan berkurang).

Pasif: Hanabi akan mendapatkan 7 persen lifesteal yang bisa bertambah sebesar 3 persen setiap kali level dari skill ini di-upgrade.

```python
#membuat prototype Hero
class Hero:
    #magic function konstruktor
    def __init__ (self, n, r, h, hr, pa, pd, \
        a, m, mr, mp, md, ms):
        self.nama = n
        self.role = r
        self.hp = h
        self.hpRegen = hr
        self.phat = pa
        self.phdef = pd
        self.ats = a
```

```
13          self.mana = m
14          self.manaRegen = mr
15          self.map = mp
16          self.mad = md
17          self.mos = ms
18
19      #contoh fungsi return
20      def skill1 (self):
21          print(self.nama , "menggunakan skill 1")
22          mana = 20
23          sisaMana = self.mana - mana
24          return sisaMana
```

```
25
26  #membuat objek pertama dari kelas Hero
27  hanabi = Hero("Hanabi","Marksman" ,2510 , 6, 105, 17, \
28      1.06, 390, 15, 0, 15, 245)
29
30  #membuat objek kedua dari kelas Hero
31  uranus = Hero("Uranus","Tank" ,2489 , 6.4, 115, 14, \
32      1.04, 455, 12, 0, 15, 260)
33
34  efekSisaMana = hanabi.skill1()
35  print("Sisa Mana Hanabi adalah:" , efekSisaMana)
36
```


11
04:33
1
65ms
0
0
1
488
Hendar13
6
Alzis
5
40
86
253
5
Recall
Regen
Flicker

Membuat fungsi return berargumen menggunakan OOP python

```python
#Syntaks fungsi return berargumen
def namaFungsi (self , argumen):
    #isiFungsi
    nilaiFungsi = argumen
    return (nilaiFungsi)
```

Detail Objek Hanabi

- **Fungsi:**
  - **Skill 2 : Soul Scroll**

    Cooldown: 12

    Mana cost: 35

Hanabi menembakkan kunai ke arah tertentu, melekat pada “scroll” atau gulungan yang dikunci dengan energi “demonic flower“. Lawan yang terserang dalam jalurnya akan terkena 300 physical damage (+80 persen dari total physical attack).

Selain itu, serangan Hanabi akan mengurangi movement speed musuh sebesar 60 persen selama 2 detik. Hanabi juga akan mendapatkan kembali 140 poin mana setiap kali ada hero yang terkena skill ini.

```python
#membuat prototype Hero
class Hero:
    #magic function konstruktor
    def __init__ (self, n, r, h, hr, pa, pd, \
        a, m, mr, mp, md, ms):
        #("Hanabi","Marksman" ,2510 , 6, 105, 17,
        # 1.06, 390, 15, 0, 15, 245)
        self.nama = n
        self.role = r
        self.hp = h
        self.hpRegen = hr
        self.phat = pa
```

```
13          self.phdef = pd
14          self.ats = a
15          self.mana = m
16          self.manaRegen = mr
17          self.map = mp
18          self.mad = md
19          self.mos = ms
20
21      #contoh fungsi return berargumen
22      def skill2 (self, movementSpeed):
23          print(self.nama , "menggunakan skill 2")
24          movementSpeedMinus = int(60 * movementSpeed / 100)
25          return movementSpeedMinus
```

```
27  hanabi = Hero("Hanabi","Marksman" ,2510 , 6, 105, 17, \
28      1.06, 390, 15, 0, 15, 245)
29
30  uranus = Hero("Uranus","Tank" ,2489 , 6.4, 115, 14, \
31      1.04, 455, 12, 0, 15, 260)
32
33  #contoh fungsi return berargumen
34  print("Movement Speed",uranus.nama,"semula",uranus.mos)
35  efekMovementSpeed = hanabi.skill2(uranus.mos)
36  movementSpeedUranus = uranus.mos - efekMovementSpeed
37  uranus.mos = movementSpeedUranus
38  print("Movement Speed",uranus.nama,\
39      "setelah skill 2 hanabi",uranus.mos)
```


11
05:07
2
0
0
1
47
[Computer] Lavla
[Computer] Vexana
Batal
Cast
Alzis
Recall
Regen
Flicker

Membuat fungsi void OOP menggunakan python

Detail Objek Hanabi

- **Fungsi:**
  - **Skill 3 / Ultimate : Higanbana**

    Cooldown: 32

    Mana cost: 50

Saat bunga mekar, daun akan terlupakan. Hanabi membuka kunci Higanbana dan menembakkan ke arah musuh. Setelah berhasil menyerang musuh, Higanbana akan mekar dan memberikan 400 physical damage (+100 persen dari total physical attack) dan immobilize atau melumpuhkan musuh selama 2 detik, kemudian terpencar ke arah musuh terdekat.

Setelah beberapa waktu, bila musuh masih berada dalam jangkauan Higanbana, bunga baru akan muncul, menimbulkan damage serta efek immobilize yang sama. Efek terpencarnya bunga hanya berfungsi satu kali untuk satu hero.

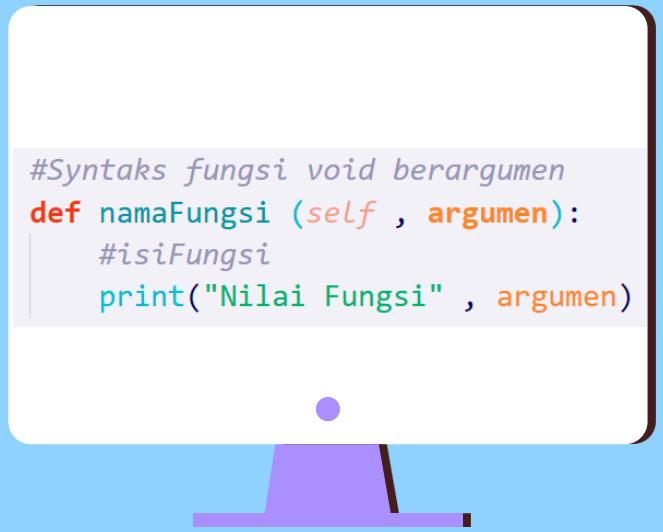
```python
#membuat prototype Hero
class Hero:
    #magic function konstruktor
    def __init__ (self, n, r, h, hr, pa, pd, \
        a, m, mr, mp, md, ms):
        #("Hanabi","Marksman" ,2510 , 6, 105, 17,
        # 1.06, 390, 15, 0, 15, 245)
        self.nama = n
        self.role = r
        self.hp = h
        self.hpRegen = hr
        self.phat = pa
```

```
13          self.phdef = pd
14          self.ats = a
15          self.mana = m
16          self.manaRegen = mr
17          self.map = mp
18          self.mad = md
19          self.mos = ms
20
21      #contoh fungsi void
22      def skillUltimate(self):
23          print(self.nama, "menggunakan skill Ultimate")
24          print(self.nama, "membuka kunci Higanbana")
```

```
25
26  #membuat objek pertama dari kelas Hero
27  hanabi = Hero("Hanabi","Marksman" ,2510 , 6, 105, 17, \
28      1.06, 390, 15, 0, 15, 245)
29
30  #membuat objek kedua dari kelas Hero
31  uranus = Hero("Uranus","Tank" ,2489 , 6.4, 115, 14, \
32      1.04, 455, 12, 0, 15, 260)
33
34  #contoh fungsi void
35  hanabi.skillUltimate()
36
```


25
09:28
8
95ms
500
Alzis
9
Absorb
32
Recall
Regen
Flicker

Membuat fungsi void berargumen menggunakan OOP Python



Detail Objek Hanabi

❖ Fungsi:

❖ Skill Pasif : Equinox

Sebuah teknik rahasia “scarlet shadow“, di mana saat HP Hanabi dalam kondisi penuh, 50 persen dari HP yang diterima dari lifesteal akan diubah menjadi shield.

Perisai tersebut bisa menahan sampai 20 persen dari maksimal HP-nya. Ketika perisai aktif, Hanabi akan immune atau kebal terhadap efek control apapun dari musuh.

```python
#membuat prototype Hero
class Hero:
    #magic function konstruktor
    def __init__ (self, n, r, h, hr, pa, pd, \
        a, m, mr, mp, md, ms):
        self.nama = n
        self.role = r
        self.hp = h
        self.hpRegen = hr
        self.phat = pa
        self.phdef = pd
        self.ats = a
```

```
13          self.mana = m
14          self.manaRegen = mr
15          self.map = mp
16          self.mad = md
17          self.mos = ms
18
19      def skill1 (self):
20          print(self.nama , "menggunakan skill 1")
21          mana = 20
22          sisaMana = self.mana - mana
23          lifesteal = int(7 * 173 / 100)
24          return sisaMana , lifesteal
```

```
27  def skillPasif(self,hp):
28      print("Skill Pasif", self.nama , "non-Aktif" )
29      if self.hp > 2510:
30          print("Skill Pasif", self.nama , "Aktif" )
31          shield = int(50 * hp / 100)
32          print(self.nama, "mendapatkan shild " ,shield)
33      elif self.hp < 2510:
34          self.hp = self.hp + hp
35          if self.hp > 2510:
36              print("Skill Pasif", self.nama , "Aktif" )
37              shield = int(50 * hp / 100)
38              print(self.nama,"mendapatkan shild" ,shield)
```

```
39            else:
40                print("Skill Pasif", self.nama , "Aktif" )
41                shield = int(50 * hp / 100)
42                print(self.nama,"mendapatkan shild ",shield)
43
44  hanabi = Hero("Hanabi","Marksman" ,2510 , 6, 105, 17, \
45      1.06, 390, 15, 0, 15, 245)
46
47  uranus = Hero("Uranus","Tank" ,2489 , 6.4, 115, 14, \
48      1.04, 455, 12, 0, 15, 260)
49
```

```
49
50  returnSkill1 = hanabi.skill1()
51  print(type(returnSkill1))
52  efekSisaMana = returnSkill1[0]
53  print("Sisa Mana Hanabi adalah:" , efekSisaMana)
54
55  #contoh fungsi void berargumen
56  hp = returnSkill1[1]
57  hanabi.skillPasif(hp)
58
59
60
```


1
01:11
0
0
0
1
158
Alzis
2
173
Recall
Regen
Flicker

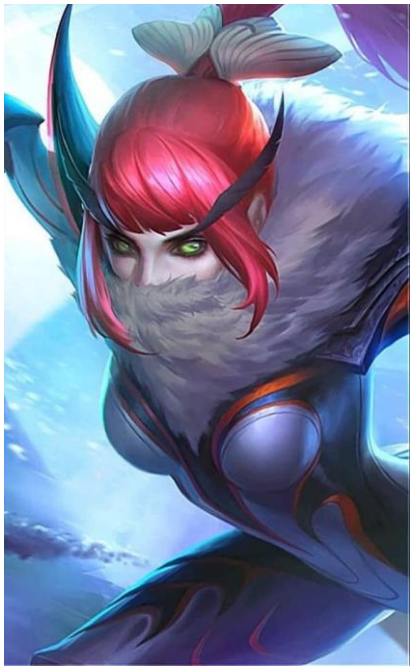
03

Implementasi

Pemrograman Berorientasi Objek

Home Game Tools Facts Team

Implementasi atribut dan fungsi



- Kelas
- Objek
- Magic Function Constructor
- Atribut
- Fungsi
  - Return Function
  - Return Function dengan Argumen
  - Void Function
  - Void Function dengan Argumen

Whoa!

Ternyata kita sudah membahas banyak hal terkait pemrograman berorientasi objek. Yeay !

Mari kita rangkai keseluruhan pembahasan menjadi satu program

Pemrograman Berorientasi Objek

Home Game Tools Facts Team

```python
#membuat prototype Hero
class Hero:
    #magic function konstruktor
    def __init__ (self, n, r, h, hr, pa, pd, \
        a, m, mr, mp, md, ms):
        #("Hanabi","Marksman" ,2510 , 6, 105, 17,
        # 1.06, 390, 15, 0, 15, 245)
        self.nama = n
        self.role = r
        self.hp = h
        self.hpRegen = hr
        self.phat = pa
```

```
13          self.phdef = pd
14          self.ats = a
15          self.mana = m
16          self.manaRegen = mr
17          self.map = mp
18          self.mad = md
19          self.mos = ms
20
21      #contoh fungsi return
22      def skill1 (self):
23          print(self.nama , "menggunakan skill 1")
24          mana = 20
```

```
25          sisaMana = self.mana - mana
26          lifesteal = int(7 * 250 / 100)
27          return sisaMana , lifesteal
28
29      #contoh fungsi return berargumen
30      def skill2 (self, movementSpeed):
31          print(self.nama , "menggunakan skill 2")
32          movementSpeedMinus = int(60 * movementSpeed / 100)
33          return movementSpeedMinus
34
35      #contoh fungsi void
36      def skillUltimate(self):
```

```
36  def skillUltimate(self):
37      print(self.nama, "menggunakan skill Ultimate")
38      print(self.nama, "membuka kunci Higanbana")
39
40  #contoh fungsi void berargumren
41  def skillPasif(self,hp):
42      print("Skill Pasif", self.nama , "non-Aktif" )
43      if self.hp > 2510:
44          print("Skill Pasif", self.nama , "Aktif" )
45          shield = int(50 * hp / 100)
46          print(self.nama, "mendapatkan shild" , shield)
47      elif self.hp < 2510:
```

```
47      elif self.hp < 2510:
48          self.hp = self.hp + hp
49          if self.hp > 2510:
50              print("Skill Pasif", self.nama , "Aktif" )
51              shield = int(50 * hp / 100)
52              print(self.nama,"mendapatkan shild" ,shield)
53      else:
54          print("Skill Pasif", self.nama , "Aktif" )
55          shield = int(50 * hp / 100)
56          print(self.nama, "mendapatkan shild" ,shield)
57
```

```
57
58  #membuat objek pertama dari kelas Hero
59  hanabi = Hero("Hanabi","Marksman" ,2510 , 6, 105, 17, \
60      1.06, 390, 15, 0, 15, 245)
61  print("HP mula-mula hanabi adalah: " , hanabi.hp) #2510
62
63  #membuat objek kedua dari kelas Hero
64  uranus = Hero("Uranus","Tank" ,2489 , 6.4, 115, 14, \
65      1.04, 455, 12, 0, 15, 260)
66  print("Role heru uranus adalah : " , uranus.role) #Tank
67
```

```
returnSkill1 = hanabi.skill1()
print(type(returnSkill1))
efekSisaMana = returnSkill1[0]
print("Sisa Mana Hanabi adalah:" , efekSisaMana)
print("Sisa Mana Hanabi adalah:" , hanabi.mana - 20)

#Hubungan antar dua objek
print("Movement Speed",uranus.nama,"semula",uranus.mos)
efekMovementSpeed = hanabi.skill2(uranus.mos)
movementSpeedUranus = uranus.mos - efekMovementSpeed
uranus.mos = movementSpeedUranus
print("Movement Speed",uranus.nama,"setelah",uranus.mos)
```

```
80
81  #contoh fungsi void
82  hanabi.skillUltimate()
83
84  #contoh fungsi void berargumen
85  hp = returnSkill1[1]
86  hanabi.skillPasif(hp)
87
88
```

04

Studi Kasus

Pemrograman Berorientasi Objek

Home Game Tools Facts Team

Whoa!

Kita baru tersadar telah membuat 2 buah objek dari satu kelas. Bagaimana menyimpan kedua objek tersebut ?

Studi Kasus: [Hanabi & Uranus]

```python
#membuat prototype Hero
class Hero:
    #magic function konstruktor
    def __init__ (self, n, r, h, hr, pa, pd, \
        a, m, mr, mp, md, ms):
        #("Hanabi","Marksman" ,2510 , 6, 105, 17,
        # 1.06, 390, 15, 0, 15, 245)
        self.nama = n
        self.role = r
        self.hp = h
        self.hpRegen = hr
        self.phat = pa
        self.phdef = pd
        self.ats = a
```

```
15          self.mana = m
16          self.manaRegen = mr
17          self.map = mp
18          self.mad = md
19          self.mos = ms
20
21  #membuat objek pertama dari kelas Hero
22  hanabi = Hero("Hanabi","Marksman" ,2510 , 6, 105, 17, \
23      1.06, 390, 15, 0, 15, 245)
24
25  #membuat objek kedua dari kelas Hero
26  uranus = Hero("Uranus","Tank" ,2489 , 6.4, 115, 14, \
27      1.04, 455, 12, 0, 15, 260)
28
```

```
#List tempat menyimpan objek
SaveHiro = [hanabi,uranus]
print(SaveHiro[0])

#Mengakses atribut dari objek dalam list
print(SaveHiro[0].nama)

#Operasi aritmatika pada atribut objek
print(SaveHiro[0].hp + SaveHiro[1].hp )

#menggunakan method array
SaveHiro.reverse()
print(SaveHiro[0].nama)

```

Flash/back

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Studi Kasus: {Hanabi & Uranus}

```python
#membuat prototype Hero
class Hero:
    #magic function konstruktor
    def __init__ (self, n, r, h, hr, pa, pd, \
        a, m, mr, mp, md, ms):
        #("Hanabi","Marksman" ,2510 , 6, 105, 17,
        # 1.06, 390, 15, 0, 15, 245)
        self.nama = n
        self.role = r
        self.hp = h
        self.hpRegen = hr
        self.phat = pa
        self.phdef = pd
        self.ats = a
```

```
15          self.mana = m
16          self.manaRegen = mr
17          self.map = mp
18          self.mad = md
19          self.mos = ms
20
21  #membuat objek pertama dari kelas Hero
22  hanabi = Hero("Hanabi","Marksman" ,2510 , 6, 105, 17, \
23      1.06, 390, 15, 0, 15, 245)
24
25  #membuat objek kedua dari kelas Hero
26  uranus = Hero("Uranus","Tank" ,2489 , 6.4, 115, 14, \
27      1.04, 455, 12, 0, 15, 260)
28
```

```
29  #kamus tempat menyimpan objek
30  SaveHiro = {hanabi.nama:hanabi , uranus.nama:uranus}
31  print(SaveHiro)
32
33  #Mengakses atribut dari objek dalam kamus
34  print(SaveHiro['Uranus'])
35
36  #Operasi aritmatika pada atribut objek
37  print(SaveHiro['Hanabi'].hp + SaveHiro['Uranus'].hp)
38
39  #menggunakan method array
40  print(SaveHiro.keys())
41  print(SaveHiro.values())
42
```

Flash/back

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Terima Kasih