

Python PCAP 31-03 Exam Practice Questions List (Difficult or Reminder)

by [naufalnashif](#)

Harga Sertifikasi PCAP 31: \$295 (+- Rp. 4.697.432,50) | [link](#)

Courses:

- [Cisco Python Essential 1 \(Course Entry Level PCEP 30 + Discount 20%\)](#)
- [Cisco Python Essential 2 \(Course Associate PCAP 31 + Discount 50%\)](#)
- [Udemy PCAP 31-03](#)

Exam Practice:

- [Exam Practice Without Answer](#)
- [Exam Practice With Answer](#)

Note :

Passing grade 70/100 dari 40 soal, atau 28/40 soal. Berarti maksimal salah 12 soal.

Modules & Packages

Soal	Keterangan
Select all valid option(s) about the result of dir() <input type="checkbox"/> A list of filenames inside the directory <input checked="" type="checkbox"/> A list of the module's attribute <input checked="" type="checkbox"/> A list of names of class attributes <input checked="" type="checkbox"/> A list of names of object attributes <input checked="" type="checkbox"/> A list of names of the base class attributes Answer >>>	
math.ceil() or math.floor()	return int
math.factorial(x)	x (int) >= 0 0! = 1
random.choices(populasi, weights, k)	return list
random.sample(populasi, k)	return list

Soal

Keterangan

```
1 import platform  
2  
3 print(platform.platform())  
4 print(platform.platform(alienated = 0, terse = 0))  
5 print(platform.platform(alienated = 1, terse = 1))  
6  
7 print(platform.machine())  
8 print(platform.processor())  
9 print(platform.system())  
10 print(platform.version())  
11 print(type(platform.version()))
```

```
→ Linux-6.1.85+-x86_64-with-glibc2.35  
Linux-6.1.85+-x86_64-with-glibc2.35  
Linux-6.1.85+-x86_64-with-glibc2.35  
x86_64  
x86_64  
Linux  
#1 SMP PREEMPT_DYNAMIC Thu Jun 27 21:05:47 UTC 2024  
<class 'str'>
```

The digraph written as `#!` is used to:

- tell an MS Windows OS how to execute the contents of a Python file
- tell a Unix or Unix-like OS how to execute the contents of a Python file
- make a particular module entity private
- create a docstring

shebang or hashbang

memberi tahu path to interpreter untuk dijalankan di Unix OS

Question 5

When a module is imported, its contents:

- may be executed (explicitly)
 - are executed as many times as they are imported
 - are ignored
 - are executed once (implicitly)
- When a module in Python is imported, **the entire file is executed once implicitly** in order to assure no errors are present in the module.

ketika import module, maka hanya di eksekusi sekali walaupun kita melakukan import berulang ulang

The `sys.stderr` stream is normally associated with:

- the console/terminal
- the screen

Error Handling (*Exceptions*)

Soal

What is the output of the following code?

```
class E (Exception):
    def __init__(self, message):
        self.message = message
    def __str__(self):
        return "Surprise"
```

```
try :
    raise Exception( "Stop" )
except E as e:
    print(e)
else:
    print( "Goodbye" )
```

- Unhandled Exception
() Surprise
() Stop
() Goodbye

Keterangan

Unhandled Exception :
ada `raise` yang tidak di tangkap

Jika tidak ada error di blok `try`, baru `else` akan tereksekusi

```
1  class E(Exception):
2      def __init__(self, message):
3          self.message = message
4      # def __str__(self):
5      #     return "Surprise!"
6
7  try:
8      raise Exception('Erorrr!')
9 except E as e:
10    print(e)
11 else:
12    print('GoodBye!')
```

Exception Traceback (most recent call last)
<ipython-input-4-d4d59fdb0b7e> in <cell line: 7>()
 6
 7 try:
----> 8 raise Exception('Erorrr!')
 9 except E as e:
 10 print(e)

Exception: Erorrr!

```
1  class E(Exception):
2      def __init__(self, message):
3          self.message = message
4      def __str__(self):
5          return 'Suprise!'
6
7  try:
8      raise E('Erorrr!')
9 except E as e:
10    print(e)
11 else:
12    print(['GoodBye!'])
```

Suprise!

`__str__` akan dipanggil ketika `print(e)` alih - alih memanggil `self.message` yang berisi pesan Error!

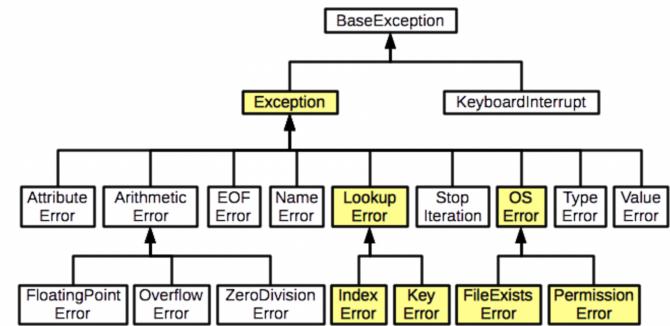
```
1  class E(Exception):
2      def __init__(self, message):
3          self.message = message
4      # def __str__(self):
5      #     return 'Suprise!'
6
7  try:
8      raise E('Erorrr!')
9 except E as e:
10    print(e)
11 else:
12    print('GoodBye!')
```

Erorrr!

Soal	Keterangan
<p>What is the output of the following code?</p> <pre data-bbox="112 114 451 332"> try : raise Exception except: Default Except must be last print("Spam" , end= " ") except BaseException: print("Ham" , end= " ") except Exception: print("Eggs") </pre>	Default except must be last
<pre data-bbox="96 377 467 595"> class SpamException (Exception): def __init__(self, message): <<< INSERT CODE HERE >>> self.message = message raise SpamException("Spam") [X] super().__init__(message) [!] Exception.__init__(self, message) [X] super(SpamException, self).__init__(message) [!] super.__init__(message) </pre>	<p>pemanggilan attr pada kelas induk</p> <ol style="list-style-type: none"> 1. <code>super().__init__(message)</code> 2. <code>super(SpamException, self).__init__(message)</code> 3. <code>ParentClass.__init__(self, message)</code>
<p>What is the result of the following code?</p> <pre data-bbox="112 699 451 827"> >>> assert (False , 'Trigger Assertion') [N] No output [!] Trigger Assertion [!] SyntaxError: invalid syntax [!] Assertion is always true </pre>	<p>tuple yang memiliki isi dianggap <code>True</code></p> <p><code>assert False, 'Message'</code> akan selalu <code>False</code></p>
<p>Which of the following are examples of built-in concrete Python exceptions? (Select two answers)</p> <p><input type="checkbox"/> <code>BaseException</code></p> <p><input type="checkbox"/> <code>ArithmeticError</code></p> <p><input checked="" type="checkbox"/> <code>ImportError</code></p> <p><input type="checkbox"/> <code>IndexError</code></p>	build in konkret (error plaing dasar/spesifik)
<pre data-bbox="112 1171 632 1546"> 1 try: 2 raise Exception 3 except BaseException: 4 print("a") 5 except Exception: 6 print("b") 7 except: 8 print("c") </pre> <p>a</p>	BaseException merupakan exception paling luas sehingga exception di bawahnya tidak akan ter eksekusi
<pre data-bbox="112 1710 616 2086"> try: pass except Exception: pass except BaseException: pass except: pass </pre>	sebaiknya dalam membuat urutan exception dari exception yang paling spesifik baru ke yang paling luas dan untuk <code>except</code> harus di akhir

Soal

Keterangan



```
1 try :  
2 |     raise Exception ('Your Message!')  
3 except Exception as e :  
4 |     print(e)  
5 |     print(e.args)
```

Output:
Your Message!
(Your Message!,)

```
1 for subclass in Exception.__subclasses__():  
2 |     print(subclass.__name__)
```

TypeError
StopAsyncIteration
StopIteration
ImportError
OSError
EOFError
RuntimeError
NameError

```
1 class MyEx(Exception):  
2 |     def __init__(self, msg):  
3 |         Exception.__init__(self, msg+msg)  
4 |         self.args = (msg,)  
5 |  
6 try:  
7 |     raise MyEx('wrong!')  
8 except MyEx as e:  
9 |     print(e)
```

wrong!

```
1 try:  
2 |     val = int(input('Provide a number: '))  
3 |     print(val/val)  
4 except TypeError:  
5 |     print('a')  
6 except ValueError:  
7 |     print('b')  
8 except ZeroDivisionError:  
9 |     print('c')  
10 except:  
11 |     print(['d'])
```

Provide a number: a
b

e.args -> tuple

menghasilkan hierarchy error.

Cara buat exception hanya melakukan inheritance dari error paling bawah

self.args akan mengoverride msg di instance

type sudah benar str hanya value yang salah

Strings

Soal

Keterangan

What is the output of the following code?

```
print("C:\Program Files\Microsoft\Windows NT", end= "")  
print( "")  
( ) Syntax Error  
( ) C:\Program Files\Microsoft\Windows NT  
( ) Replace escaped characters with "?" e.g. C:\?rogram Files\?icrosoft?  
indows NT?  
( ) Ignore escaped characters e.g. C: rogram Filesicrosoftindows NT
```

Answer :

- Syntax Error

* ASCII, UNICODE, UTF-8, codepoints, escape sequences
Select all valid option(s) below about string
 string.ascii_letters is a concatenation of ascii_lowercase and ascii_uppercase
 string.ascii_letters is a concatenation of ascii_lowercase, ascii_uppercase and digits
 string.ascii_letters are all printable characters found in the keyboard
 string.ascii_lowercase contains 'abcdefghijklmnopqrstuvwxyz'
 string.ascii_uppercase contains 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

Strings : upper + lower

bukan digit dan any char

What is the output of the following code?

```
spam = chr('a')  
ham = ord(spam)  
print(spam, ham)  
( ) 97 a  
( ) TypeError: an integer is required (got type str)  
( ) TypeError: chr() takes exactly two arguments (1 given)  
( ) TypeError: ord() takes exactly two arguments (1 given)  
( ) Syntax Error
```

Answer : b

chr(int) not chr('a')

spam = 'FuBar'
ham = spam[:]
 spam == ham
 id(spam) == id(ham)
 spam.startswith(ham)
 spam.endswith(ham)
 spam.equals(ham)

Jika *immutable*, python ga akan membuat memori baru
id(spam) == id(ham)

What is the output of the following code?

```
>>> None * 2  
( ) 0  
( ) None  
( ) NoneNone  
( ) TypeError: unsupported operand type(s) for *  
Answer >>>
```

not support NoneType

What is the output of the following code?

```
>>> sorted([ 5 , "1" , 100 , "34" ])  
( ) ["1", 5, "34", 100]  
( ) [5, "1", "34", 100]  
( ) ["1", "100", "34", "5"]  
( ) [1, 5, 34, 100]  
( ) TypeError: 'less than or equal to' not supported between instances of 'str' and 'int'
```

cannot sort int and str

What is the output of the following code?

```
spam.txt  
spam ham eggs  
spam.py  
f = open('spam.txt', 'r')  
if 'eggs' in f:  
    print('Eggs found')  
else:  
    print('Eggs not found')  
( ) Eggs found  
( ) Eggs not found  
( ) TypeError: argument type TextIOWrapper not iterable  
( ) SyntaxError: invalid syntax
```

open(file, 'r') return file object not file content

content = f.read()

Which of the calls below are valid String function calls and will return True?

```
'abc123'.isalnum()  
'abc'.isalpha()  
'123abc'.isidentifier()  
'123'.islower()  
'123'.isdigit()  
'Abe'.istitle()
```

isalnum() -> isalpha() + isnumeric()

isnumeric -> termasuk isdigit(0-9)

isidentifier() -> cek apakah valid sebagai variable

What is the output of the following code?

```
>>> sorted(['banana', 'pear', 'grapes', 'apple'], key= lambda x:  
x[::-1])  
( ) 'apple' 'grapes' 'pear' 'banana'
```

sorted(iter, key)

key = fungsi

key = lambda x : x -1 -> reverse setiap element

Soal	Keterangan
<pre>* .sort(), sorted(), .index(), .find(), .rfind() What is the output of the following code? >>> "Spam Ham Eggs" .index('Spam' , 1) ()</pre> <p>() Spam () 0 () 1 ↗ ValueError: substring not found () TypeError: index() takes 1 argument (2 given)</p>	<pre>.index(str, start, end) 'Spam' ada di index 0 bukan 1 jadi ValueError = not found</pre>
<pre>* .sort(), sorted(), .index(), .find(), .rfind() What is the output of the following code? t = "Spam Ham" print(t.find("Ham" , 0)== t.index("Ham" , 0)) print(t.find("Eggs" , 0)== t.index("Eggs" , 0)) ()</pre> <p>() True True () True False ↗ True will be printed followed by ValueError: substring not found () True will be printed followed by TypeError: find() takes 1 argument (2 given) Answer >>></p> <pre>* .sort(), sorted(), .index(), .find(), .rfind() What is the output of the following code? t = "Spam Ham" print(t.rfind("am") == t.find("am")) print(t.rfind("am" , 3) == t.find("am" , 3)) print(t.rfind("am" , -3)== t.find("am" , -3)) ()</pre> <p>() False False False ↗ False True True () True True True () True will be printed followed by TypeError: rfind takes 1 argument (2 given) Answer >>> </p>	<pre>.index() -> jika tidak ditemukan maka ValueError .find() & .rfind() -> return -1</pre>
<p>UTF-8 is:</p> <ul style="list-style-type: none"> <input type="radio"/> a synonym for byte <input checked="" type="radio"/> a form of encoding Unicode code points <input type="radio"/> the 9th version of the UTF standard <input type="radio"/> a Python version name 	<p>UTF – 8 is a form of encoding Unicode code points</p>
<pre>1 print('Anre' > "Anrea") 2 3</pre> <p>False</p> <pre>1 print('2Anre' > "1Anrea") 2 3</pre> <p>True</p> <pre>1 'z' < 'aaaa'</pre> <p>False</p>	<p>pembandingan string berdasarkan nilai UNICODE karakter demi karakter dari kiri, ketika menemukan perbedaan, python tidak melanjutkan pembandingan karakter berikutnya</p> <p>- karena <code>ord('2') > ord('1')</code> ini langsung memunculkan output True tanpa membandingkan karakter berikutnya</p> <p>- <code>ord('z') > ord('a')</code></p>

Soal

What will be printed in the following code?

```
spam = "!!!!!!"
ham = "!!!"
!!!!
print(spam, ham)
()
Syntax Error
()
Two empty strings
()
An empty string and a new line character
()
Two new line character
Answer >>>
```

Keterangan

```
1 ham = "!!!"
2 !!!
3
4 len(ham)
```

1

s = 'Python'

- () print(s[0] + s[-1])
- () print(s[:5])
- print(s[::-5])
- () print(s[:-1][:-5])

Answer >>>

step -5 berarti membaca dari index terakhir dengan step 5 -> nP

sedangkan step 5 -> Pn

```
1 '' in 'AnyString'
```

True

string kosong itu termasuk kedalam string apapun

```
1 with open('file.txt', 'w') as f:
2     f.write('spam ham eggs')
3
4 with open('file.txt', 'r') as f:
5
6     if 'eggs' in f:
7         print('eggs found')
8     else:
9         print('eggs not found')
10
11     print(f.read(), 'tidak ada file karena pointer di akhir')
12     f.seek(0)
13     print(f.read())
14     print(f)

eggs not found
tidak ada file karena pointer di akhir
spam ham eggs
<_io.TextIOWrapper name='file.txt' mode='r' encoding='UTF-8'>
```

open() as f ini mengembalikan object bukan content , jadi tidak bisa diiterasi

untuk melakukan iterasi lakukan read dahulu

What is the output of the following code?

```
>>> "/" .join({ "Month" : "12", "Day" : "25" , "Year" : "2021" })
()
12/25/2021
()
Month/Day/Year
()
Month/12/Day/25/Year/2021
()
TypeError: can only join an iterable
Answer >>>
```

join pada dict hanya akan mengambil key

What is the output of the following code?

```
>>> "XYZ" .join( "123" )
()
XYZ123
()
123XYZ
()
1XYZ2XYZ3
()
X123Y123Z
()
TypeError: can only join an iterable
Answer >>>
```

menghasilkan string kosong dan panjang list sebanyak /

```
1 '/spam/ham/eggs/'.split('/')
['', 'spam', 'ham', 'eggs', '']
```

OOP

Soal	Keterangan
What is the output of the following code? <pre>class Foo : bar = 'spam' f1 = Foo() f2 = Foo() f2.bar = 'ham' Foo.bar = 'eggs' print(f1.bar, f2.bar, Foo.bar)</pre> <p>() spam ham eggs X eggs ham eggs () eggs eggs eggs</p>	Jika instance tidak melakukan rewrite bar maka bar akan mengambil attr kelas
<pre>issubclass(classA, classA) will return True</pre> <pre>isinheritance(classA, classA) will return True</pre>	
<pre>1 str_one = 'hello' 2 str_two = 'hell' 3 str_two += 'o' 4 5 print(str_one is str_two)</pre> False	because when modify string, python will make a different reff memory is used to compare 2 object memory
 <pre>1 class Void: 2 pass 3 4 a = Void() 5 b = Void() 6 7 a is b</pre>  False	dua instance pasti memiliki beda referensi memori
JANGAN GUNAKAN assert dalam production	dengan perintah -O ketika run program di CLI maka semua assert akan unactive
<pre>sum = lambda x , y : x + y sum(10, 1)</pre>	lambda bisa di assign ke variable

Soal

```
def count_occurrences(file_name, word):
    word_counts = {}

    try :
        with open(file_name, 'r') as f:
            words = f.read().replace(',', '').replace('.', '').split()
            for w in words:
                w = w.lower()
                if w in word_counts:
                    word_counts[w] += 1
                else:
                    word_counts[w] = 1
    except:
        print('Something error!')
        raise

    return word_counts[word]
    return word_counts.get(word.lower(), 0)
```

Keterangan

pendekatan terbaik untuk return menggunakan get(), bukan menggunakan index.

jika menggunakan index, ketika key tidak ada maka akan error, sedangkan jika dengan get() itu akan mengembalikan nilai default yaitu 0

```
1 class A():
2     attrClass = ''
3     def __init__(self):
4         self.name = 'empty'
5
6     print(A.__dict__)
7     print(len(A.__dict__))
8     print(A().__dict__)
9     print(len(A().__dict__))

{'__module__': '__main__', 'attrClass':
6
{'name': 'empty'}
1
```

A -> kelas
A() -> instance

```
1 class Dog():
2     def __init__(self, name = 'none'):
3         self.name = name
4
5     def set(self, name = 'any'):
6         self.name = name
7
8 obj1 = Dog()
9 obj2 = obj1
10 obj2.set()
11
12 print(obj1.name)

any
```

obj2 merubah value obj1 karena merujuk pada memori yang sama

```
1 class A():
2     def __init__(self):
3         self.c = 'any'
4
5 hasattr(A(), 'c')

True
```

A() adalah instance jadi memiliki c

Cara baca:

Apakah A() memiliki attr 'c'

Soal

```
1 def a():
2     hey = 5
3
4     def b():
5         print(hey)
6
7     return b
8
9 x = a()
10 print(x())
```

5
None

```
1 def myFunc():
2     print('hello')
3
4 print(myFunc())
```

hello
None

isinstance(obj, cls)

hanya berlaku di instance (obj)

Mengecek apakah suatu **objek** adalah instance dari suatu kelas atau subclass.

issubclass(cls1, cls2)

hanya berlaku di kelas

Mengecek apakah suatu kelas adalah subclass dari kelas lain.

What is the output of the following code?

```
class Ham :
    def __init__(self):
        self.v1 = 1
class Spam(Ham):
    def __init__(self):
        self.v2 = 2
s = Spam()
print(s.v1,s.v2)
()
```

0 2
1 2
Invalid Syntax
AttributeError: 'Spam' object has no attribute 'v1'

A 

ketika mendeklarasikan `__init__` sendiri maka untuk memunculkan attr pada kelas induk harus menggunakan `super().__init__()` atau `Ham.__init__(self)`

Soal



```
1 def foo(self, p):
2     print('Hello', p)
3
4 class Spam:
5     bar = foo
6
7 s = Spam()
8 s.bar('World!')
```

→ Hello World!

Keterangan

foo disimpan ke bar tanpa di eksekusi, hanya menyimpan referensi

sedangkan jika foo() akan error karena ketika menyimpan ke bar, foo() akan ter eksekusi

```
1 class X : pass
2 class Y : pass
3 class Z (X, Y): pass
4
5 issubclass(Z, (list, X, Y))
```

True

Apakah Z adalah salah satu subclass dari tuple(list, X, Y)

- bukan subclass dari list
- subclass dari X
- subclass dari Y

can't duplicate

```
1 class A: pass
2 class B (A, A): pass

TypeError: Traceback (most recent call last)
<ipython-input-7-dda9e50ea591> in <cell line: 2>()
      1 class A: pass
----> 2 class B (A, A): pass

TypeError: duplicate base class A
```

setiap instance memiliki attr sendiri, ketika tidak mengoverwritte attr kelas maka akan selalu menggunakan value pada attr kelas

```
1 class MyClass:
2     foo = 100
3     def __init__(self):
4         self.bar = []
5
6     def add(self, p):
7         self.bar.append(p)
8
9 obj1, obj2 = MyClass(), MyClass()
10 obj1.add('Spam')
11 obj2.add('Ham')
12
13 obj2.foo = 200 #obj2 memiliki attr foo sendiri
#obj1 masih menggunakan attr foo MyClass
14
15 MyClass.foo = 300
16
17 print(obj1.bar, obj1.foo, obj2.bar, obj2.foo)
18
19
['Spam'] 300 ['Ham'] 200
```

vars(Class) dan Class.__dict__ memberikan output yang sama, hanya pendekatan vars() lebih aman

```
+ Kod
1 class Person :
2     name = "John"
3     age = 36
4     country = "USA"
5 p = Person()
6
7 print(vars(p))
8 print(vars(Person))
9 print(p.__dict__)
10 print(Person.__dict__)

{}{'__module__': '__main__', 'name': 'John', 'age': 36, 'country': 'USA'}
{}{'__module__': '__main__', 'name': 'John', 'age': 36, 'country': 'USA'}
```

Soal

```
1 class Ham :
2     def __init__(self):
3         print(type(self))
4         print(type(self).__name__ + '.__init__()')
5
6     def update(self):
7         print(type(self))
8         print(type(self).__name__ + '.update()')
9
10 Ham().update()

<class '__main__.Ham'>
Ham.__init__()
<class '__main__.Ham'>
Ham.update()
```

Keterangan

type(self) -> menghasilkan representasi kelas

- __name__ menghasilkan nama kelas

```
1 class Spam:
2     __ham = 0
3     __ham2 = 0
4     def __eggs(self):
5         __ham = 100
6         self.__ham2 = 100
7         return self.__ham2
8
9
10 obj = Spam()
11
12 print(obj._Spam__eggs())
13 print(obj._Spam__ham)
14 print(obj._Spam__ham2)

100
0
100
```

Kenapa obj._Spam__ham mengakses attr kelas __ham alih alih mengambil __ham di method __eggs() :

```
python
def __eggs(self):
    __ham = 100 # Variabel lokal
    return __ham

• __ham = 100 adalah variabel lokal di dalam metode __eggs .
• Variabel lokal ini tidak memengaruhi atribut kelas atau atribut instans.
• Ketika metode selesai dieksekusi, variabel lokal __ham hilang.
```

Gunakan self.__ham agar menjadi attr instance dan bisa dipanggil oleh object

```
1 class A:
2     def __init__(self):
3         print(f'{self.v} + __init__')
4
5     def update(self):
6         print(self.d)
7
8     v = 10
9     d = 12
10
11 print[A()]
12 print(A().update())

10 + __init__
<__main__.A object at 0x7971a1e70d90>
10 + __init__
12
None
```

print(A())
- method di dalam kelas bisa mengambil attr walaupun deklarasinya setelah method

- print A() akan menghasilkan yang ada di __init__ dan output kelas <__main__.A object at 0x7971a1e719f0>

print(A().update())

- memanggil kontructor __init__

- memberikan output self.d

- print method tanpa return secara default memberikan output None

```
1 class Ham :
2     def __init__(self):
3         print(type(self).__name__ + '.__init__()' , end= ' ')
4         self.update()
5     def update(self):
6         print(type(self).__name__ + '.update()' , end= ' ')
7
8 class Spam(Ham):
9     def update(self):
10        print(type(self).__name__ + '.update()' , end= ' ')
11
12 print(Ham())
13 print(Spam())

Ham.__init__() Ham.update() <__main__.Ham object at 0x7cd80e532f20>
Spam.__init__() Spam.update() <__main__.Spam object at 0x7cd80e531ab0>
```

Soal

Keterangan

Spam().update() membutuhkan argumen 'param' karena __init__ pada kelas Spam memanggil update(self, param) dikelasnya sendiri

Spam().update() akan mengambil dari kelas Ham ketika __init__ dijalankan, sehingga tidak membutuhkan argumen param.

Karena __update = update dan self.__update()

perhatikan bahwa __update = update bukan __update = update(), karena jika ada kurung () method akan tereksekusi

__name__; __module__; __bases__

- __name__ : mengembalikan nama

print(__name__) -> __main__

- __module__ : mengembalikan module __main__ atau module lain jika diimport

- __bases__ : untuk melihat hierarchy pewarisan

Tabel Perbandingan:

Atribut	Dari Kelas	Dari Instance	Cara Akses dari Instance
__name__	✓	✗ (akses melalui __class__)	obj.__class__.__name__
__module__	✓	✓	obj.__module__
__bases__	✓	✗ (akses melalui __class__)	obj.__class__.__bases__

Penjelasan type sebagai pembuat kelas

Fungsi type biasanya digunakan untuk mengetahui tipe suatu objek, misalnya:

```
python
print(type(5)) # Output: <class 'int'>
```

Namun, type juga bisa digunakan untuk membuat kelas secara dinamis. Formatnya adalah:

```
python
type(class_name, bases, attributes)
```

- class_name : Nama kelas yang ingin Anda buat.
- bases : Tuple berisi kelas-kelas yang menjadi parent (base class).
- attributes : Dictionary berisi atribut atau metode untuk kelas.

Soal

```
1  0%
```

```
-----  
ZeroDivisionError Traceback (most recent call last)  
<ipython-input-46-5d2de8fe0392> in <cell line: 1>()  
----> 1 0%  
  
ZeroDivisionError: integer division or modulo by zero
```

Keterangan

mod 0 -> error

Miscellaneous

Soal

```
1 import calendar  
2  
3 c = calendar.Calendar()  
4  
5 for weekday in c.iterweekdays():  
6     print(weekday, end=" ")  
7  
8  
0 1 2 3 4 5 6
```

Keterangan

akan menghasilkan dari 0

```
1 from datetime import date, datetime  
2  
3 date_1 = date(1992, 1, 16)  
4 date_2 = date(1991, 2, 5)  
5  
6 print(date_1 - date_2)  
7  
8 datetime1 = datetime(1992, 1, 16, 11, 11, 11)  
9 datetime2 = datetime(1991, 2, 5)  
10  
11 print(datetime1 - datetime2)  
  
345 days, 0:00:00  
345 days, 11:11:11
```

jika hari = 0, maka tidak muncul

```
1 import os  
2  
3 os.mkdir('thumbnails')  
4 os.chdir('thumbnails')  
5  
6 sizes = ['small', 'medium', 'large']  
7  
8 for size in sizes:  
9     os.mkdir(size)  
10  
11 print(os.listdir())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
(.venv) naufalnashif@MacBook-Air-Naufal python-  
/Users/naufalnashif/Desktop/python-dev/main.py  
['large', 'medium', 'small']  
(.venv) naufalnashif@MacBook-Air-Naufal python-
```

bakal diurutkan sesuai abjad

```
['large', 'medium', 'small']
```

dengan urutan abjad :

```
a, b, ..., l, m, ..., s, ..., z
```

Soal

```
1 import os
2
3 os.mkdir('pictures')
4 os.chdir('pictures')
5 os.mkdir('thumbnails')
6 os.chdir('thumbnails')
7 os.mkdir('tmp')
8 os.chdir('../')
9
10 print(os.getcwd())
```

/content/pictures

Keterangan

sudah kembali ke path /pictures dengan perintah os.chdir(..)

Select the true statements. (Select two answers)

The `lambda` function can accept a **maximum of two** arguments

The `lambda` function can accept **any number** of arguments

The `lambda` function can evaluate **multiple** expressions

The `lambda` function can evaluate **only one** expression

lambda arguments (bisa banyak) :
expression (only one)

`lambda x , y, z : x + y + z`

```
1 with open('file.txt', 'w') as f:
2     f.write("Hello\n")
3     f.write("World!")
4
5 print('with for:')
6 for x in open('file.txt', 'rt'):
7     print(x)
8 print()
9 print('-----')
10 print('with read:')
11 with open('file.txt', 'r') as f:
12     print(f.read(1))
```

menggunakan `for` akan membaca line by line
sama seperti `.readline`

sedangkan menggunakan `.read` akan
membaca karakter demi karakter

- `.readline()` : Membaca line demi line hingga karakter newline atau EOF.
- `.read(1)` : Membaca karakter demi karakter.
- `.readlines()` : Membaca semua baris sekaligus, mengembalikannya sebagai list of strings.

with for:
Hello

World!

with read:
H

What information can be read using the `uname` function provided by the `os` module? (Select two answers)

Last login date

Hardware identifier

Operating system name

Current path

`os.uname` hanya ada di unix/linux

Soal

Keterangan

Ya, data biner yang diberikan (Hexadecimal): `0x01 0x02 0x03 0x04` terdiri dari 4 byte.

Penjelasan:

- Setiap nilai dalam format hexadecimal seperti `0x01`, `0x02`, `0x03`, dan `0x04` mewakili 1 byte.
- Total ada 4 nilai, sehingga ukurannya adalah 4 byte.
- Dalam Python, Anda dapat membuat buffer yang sesuai dengan ukuran ini menggunakan `bytearray(4)`.

`bytearray(4)` -> membuat array dengan panjang 4 byte

```
1 x = 10
2 y = 15
3
4 f = lambda i : i + x + y
5
6 f(2)
```

27

deklarasi lambda bisa seperti ini

```
python
import sys
temp = sys.stdout # Simpan referensi stdout asli
sys.stdout = open('spam.txt', 'w') # Ubah stdout ke file 'spam.txt'
print("Hello World") # Output ini ditulis ke 'spam.txt'
sys.stdout.close() # Tutup file 'spam.txt'
sys.stdout = temp # Kembalikan stdout ke konsol asli
print("Good Bye") # Output ini muncul di konsol
```

`sys.stdout` secara default diarahkan ke konsol, sehingga `print('string')` akan muncul di konsol.

Namun ketika diarahkan ke file `spam.txt` `print('string')` bukan muncul di konsol melainkan di `spam.txt`

What is the output of the following code?

a, b = 10, 20
`print(a < b and a or b)`

- 10
 20
 True
 Invalid Syntax
- (True and 10) or b
10 or b > 10

and -> 10 ;
or -> jika pertama sudah True maka tidak cek argument berikutnya

True and 10 -> mengembalikan operand ke 2 karena operan pertama True

Jika `or`, jika operan pertama True maka akan langsung di kembalikan

Select the values considered false.

None

False

zero of any numeric type (0, 0L, 0.0, 0j)

empty sequence ("", (), [])

class with a nonzero () definition

class with len () that returns integer 0

Select valid integer assignment for the variable spam?

spam = 1e0

spam = 0b1

spam = 0o1

spam = 0x1

spam = \u0031 #The Unicode of 1 is U+0031

[Answer >>>](#)

Soal**Keterangan**

```
1 print(['spam', 'ham']* 3)
2 print('spam',)*3)
```

['spam', 'ham', 'spam', 'ham', 'spam', 'ham'
('spam', 'spam', 'spam')

-1//2 -> floor(-1 / 2 = -0.5) sehingga -1

```
1 print(1. /( 3. % 2.))
2 print(-1//2)
```

1.0
-1

Select all keyword argument of print()

- sep
- end
- file
- flush
- format

print(sep, end, file, flush)

What is the output of the following code?

```
1 print(int( 10.10 ), end= " " )
2 print(int( "10" , 10 ), end= " " )
3 print(int( "10" , base= 10 ), end= " " )
4 print(int( 0o12 ), end= " " )
5 print(int( 10 ))
```

10 10 10 10 10
() TypeError: int() takes at most 1 argument (2 given)
() TypeError: 'base' is an invalid keyword argument for int()
() Invalid syntax on Line 4 (0o12)

What is the output of the code?

```
1 print(float( '+1.23' ), end= " " )
2 print(float( '-12345\n' ), end= " " )
3 print(float( '1e-003' ), end= " " )
4 print(float( '+1E6' ), end= " " )
5 print(float( '-Infinity' ))
```

1.23 -12345.0 0.001 1000000.0 -inf

What is the output of the following code?

```
>>> 5 ** 0 ** 0
```

- SyntaxError: invalid syntax
- 1
- 5
- 0

dalam python $0^{**0} = 1$

walaupun dalam matematika $0^0 = 0$

Soal**Keterangan**

```
1 x = 10
2 ham = 1 if x%2 != 0 else 0
3
4 print(ham)
```

penulisan if else bisa seperti itu

0

Thank You !!