Examination in *Programming in Python* BB1000

Suggested solutions

2018-09-29 14:00-19:00

Grading:

- E: Part 1 >= 75%
- D: Part 1 >= 75% and Part 2 >= 25%
- C: Part 1 >= 75% and Part 2 >= 75%
- B: Part 1 >= 75% and Part 2 >= 75% and Part 3 >= 25%
- A: Part 1 >= 75% and Part 2 >= 75% and Part 3 >= 75%

Each correctly answered question yields one point.

Note: Part 2 will only be graded if Part 1 has been passed. Part 3 will only be graded if both parts 1 and 2 both has been passed.

Part 1

1. Name three built-in numerical data types in Python.

   int, float, bool, complex

2. What is the difference between `=` and `==` in Python

   The first is an assignment operator the second an equality operator

3. The two most common container types for an ordered sequence of objects are `list` and `tuple`. What is the main difference between them.

   A list is mutable, a tuple immutable (can not be changed)

4. In order to write data to a file or to the screen it has to be converted to the string type `str`. Write an f-string which incorporates the value of a variable v into a string, as in

```
>>> v = 42
>>> text = f"The value of v is {v}."  # HIDE
>>> print(text)
The value of v is 42.
```

5. A dictionary is a mapping of key-value pairs. Keys have a restriction that makes one of the following statements pass

   - key = ()
   - key = []
   - key = {}

```
>>> d = {}
>>> key = ()  # HIDE
>>> d[key] = 'somevalue'
```

Which value of key will pass and why?

() is immutable

6. The accumulator pattern is a common design pattern using a for-loop for generating a single value from a given sequence of values. Outline a Python function that calculates the factorial of a non-negative integer n

```
>>> def f(n):
...     p = 1
...     for i in range(1, n + 1):
...         p = p * i
...     return p
>>> f(0)
1
>>> f(1)
1
>>> f(4)
24
```

7. How do you use git to include new changes in a remote repository into your own local repository

   git pull

8. The `id` function takes an object as input and returns a unique number for that object, its identity, as output. What is the output of the last line, True of False? This question is about the nature of assignment in Python so think carefully.

```
>>> a = 42
>>> b = 42
>>> id(a) == id(b)
True
```

Part 2

9. In a class definition, what is the name of the class method where instance variables defined?

`__init__`

10. What is the output of the append operation

```
>>> l1 = [1, 2]; l2 = [3, 4]
>>> l1.append(l2)
```

```
>>> print(l1)
[1, 2, [3, 4]]
```

11. A class `Foo` containing the following definition of `hi` could potentially be called the following way.

```
>>> class Foo:
...     def hi(self):
...         print('hi')
>>> foo = Foo()
>>> Foo.hi(foo)
hi
```

    What does a normal class method call look like in this case?

```
>>> foo.hi()
hi
```

12. Outline a function that takes arbitrary number of input arguments (positional) and returns a list with arguments in reverse order

```
>>> def f(*args):
...     return list(reversed(args))
>>> f(1)
[1]
>>> f(1, 2)
[2, 1]
```

13. Write a test function for the output examples in problem 12., that can be processed by pytest

```
def test_f():
    assert f(1) == [1]
    assert f(1, 2) == [2, 1]
```

14. The following does a scalar product of lists of numbers

```
>>> def sp(l1, l2):
...     sum = 0.0
...     for e1, e2 in zip(l1, l2):
...         sum += e1*e2
...     return sum
>>> sp([1, 2], [3, 4])
11.0
```

How would you use the numpy library for the same functionality and why in general?

`numpy.dot`, *Explicit looping in Python is slow*

15. The following function takes a Google spread-sheet as input. What is the data type of the return value?

```
>>> import pandas as pd
>>> def read_gsp(gid):
...     return pd.read_csv(
...         f'https://docs.google.com/spreadsheets/d/{gid}/export?gid=0&format=csv'
...     )
>>> retval = read_gsp('1bUPG9DoB1AvhuDts0XaamtAZCdtIw-XVLFR_uLnKn2E')
>>> print(type(retval).__name__)
DataFrame
```

16. Two objects `objа` and `objb`, are instances of a class `Cl` with a dunder-add methods. It can be called in three ways, two are

```
obja.__add__(objb)
type(obja).__add__(obja, objb)
```

what is the third?

```
obja + objb
```

Part 3

17. A general function definition has starred arguments

```
>>> def f(*args, **kwargs):
...     print(type(args).__name__)
...     print(type(kwargs).__name__)
```

What is the data type of `args` and `kwargs` respectively?

```
>>> f()
tuple
dict
```

18. How do you write and apply an identity decorator, which does not modify the function it is applied to?

```
def identity(func):
    return func
```

19. How would you write a decorator that makes a function run twice?

```
>>> def dotwice(func):
...     def wrapper(*args, **kwargs):
...         func(*args, **kwargs)
...         func(*args, **kwargs)
...     return wrapper
```

such that

```
>>> @dotwice
... def hello():
```

```
...    print("Hello")
>>> hello()
Hello
Hello
```

20. The decorator syntax is handy when you apply it to your own code, *e.g.*

```
@timer
def myfunc():
   ...
```

Another syntax must be use if you want to apply it to a library function without direct access to the code. What would you write to apply the math.sqrt function?

```
math.sqrt = timer(math.sqrt) #HIDE
```