Python 3 エンジニア認定基礎試験用模擬試験

Python 3 エンジニア認定基礎試験の模擬試験です。 試験については、下記を参照してください。

- https://www.pythonic-exam.com/exam/basic

1. Pythonの特徴として誤っているものを選択してください（1つ選択）

A. 変数や引数の宣言が不要である

B. 複数の文のまとまりをカッコで表現する

C. コンパイルが不要なインタープリター言語である

D. 配列や集合やディクショナリなど汎用的なデータ型を組み込みで使える

2. Pythonのインタープリターの特徴として正しいものを選択してください（1つ選択）

A. `python -c モジュール名`でモジュールのソースファイル名を完全な形で指定したかのように実行できる

B. インタープリターの引数は、sysモジュールのargs属性で取得できる

C. インタープリターを対話モードで起動すると、プロンプトが`>>`のように表示される

D. インタープリターを終了するには、ファイル終端キャラクタ（WindowsではCtrl+Z）を入力するか`quit()`を入力する

3. 文字列の定義として誤っているものを選択してください（1つ選択）

A. `s = "It's OK."`

B. `s = '''1. one\n2. two'''`

C. `s = 'one' 'two'`

D. `s = ""Hi!""`

4. 次のコードを実行した結果として表示されるものを選択してください（1つ選択）

```
it = "telescope"
print(it[6:] + it[2:5])
```

A. telescles

B. copeeles

C. opeles

D. scopetel

5. `name[-3:-1]`の実行結果が`"al"`になるとき、`name`の作成方法として正しいものを選択してください（1つ選択）

A. `name = "small"`

B. `name = "calorie"`

C. `name = "aluminum"`

D. `name = "oriental"`

6. 次のコードを実行した結果として表示されるものを選択してください（1つ選択）

```
ch1 = "1"
ch2 = "3"
print(ch1 * 2 + ch2)
```

A. 5

B. 23

C. 113

D. 123

7. 文字列やリストの特徴として正しいものを選択してください（1つ選択）

A. 文字列は可変体で、リストも可変体である

B. 文字列は可変体で、リストは不変体である

C. 文字列は不変体で、リストは可変体である

D. 文字列は不変体で、リストも不変体である

8. 次のコードを実行した結果として表示されるものを選択してください（1つ選択）

```
month, day = 12, 24
a, b = day + 1, (month + 1) % 12
print(a, b)
```

A. 13 1

B. 13 25

C. 25 1

D. 25 13

9. 次のコードを実行した結果として表示されるものを選択してください（1つ選択）

```
types = ["float", "int", "str"]
for t in types:
    if len(t) == 3:
        print(t)
```

A.

int str

B.

float int str

C.

int
str

D.

float
int
str

10. 次のコードを実行して期待する結果が表示されるとき、空欄①に入る記述として正しいものを選択してください（1つ選択）

```
numbers_list = [[1, 3, 5], [1, 2, 3]]
for numbers in numbers_list:
    print(numbers, end=" ")
    ①
```

【期待する結果】

```
[1, 3, 5] All numbers are odd
[1, 2, 3] Even numbers found
```

A.

```
for number in numbers:
    if number % 2 == 0:
        print("Even numbers found")
    else:
        print("All numbers are odd")
```

B.

```
for number in numbers:
    if number % 2 == 0:
        print("Even numbers found")
        break
    else:
        print("All numbers are odd")
```

C.

```
for number in numbers:
    if number % 2 == 0:
        print("Even numbers found")
else:
    print("All numbers are odd")
```

D.

```
for number in numbers:
    if number % 2 == 0:
        print("Even numbers found")
        break
else:
    print("All numbers are odd")
```

11. 次のコードを実行した結果として表示されるものを選択してください（1つ選択）

```
def calc(value, x=2):
    return value ** x

print(calc(2, 3) * calc(3))
```

A. 9

B. 18

C. 36

D. 72

12. 次のコードを実行して`Hello! Good bye!` が表示されるとき、空欄①、②に入る記述の組み合わせとして正しいものを選択してください（1つ選択）

```
def greeting(①, ②):
    for message in messages:
        print(message, **options)

greeting("Hello", "Good bye", end="! ")
```

A. ① `*messages` ② `*options`

B. ① `*messages` ② `**options`

C. ① `**messages` ② `*options`

D. ① `**messages` ② `**options`

13. 次のコードの空欄①、②、③に入る引数の説明の組み合わせとして正しいものを選択してください（1つ選択）

```
def greeting(①, /, ②, *, ③):
    pass
```

A. ① 位置のみ ② 位置またはキーワード ③ キーワードのみ

B. ① 位置のみ ② キーワードのみ ③ 位置またはキーワード

C. ① キーワードのみ ② 位置のみ ③ 位置またはキーワード

D. ① 位置またはキーワード ② 位置のみ ③ キーワードのみ

14. 次のコードの関数の使い方として誤っているものを選択してください（1つ選択）

```
def greeting(*messages, to=None):
    pass
```

A. `greeting("Hello")`

B. `greeting("Hello", to="Taro")`

C. `greeting(to="Taro", "Hello")`

D. `greeting("Hello", "Good bye")`

15. 次のコードを実行して`carrot,tomato`が表示されるとき、空欄①に入る記述として正しいものを選択してください（1つ選択）

```
vegetables = ["carrot", "tomato"]
options = {"sep": ","}
print(①)
```

A. `vegetables, options`

B. `vegetables, *options`

C. `*vegetables, *options`

D. `*vegetables, **options`

16. 次のコードを実行した結果として表示されるものを選択してください（1つ選択）

```
address = ["hokkaido", "mie", "akita"]
address.sort(key=lambda s: len(s))
print(address)
```

A. `['akita', 'hokkaido', 'mie']`

B. `['hokkaido', 'akita', 'mie']`

C. `['mie', 'akita', 'hokkaido']`

D. `['mie', 'hokkaido', 'akita']`

17. ドキュメンテーション文字列の慣習として誤っているものを選択してください（1つ選択）

A. 1行目に簡潔な説明などを書く

B. 1行目は大文字ではじめて、ピリオドで終わらせる

C. 続きを書く場合、2行目に詳細な説明を書く

D. 詳細な説明として、呼び出し方法や副作用などを書く

18. 次のコードを実行した結果として表示されるものを選択してください（1つ選択）

```
vegetables = ["tomato"]
vegetables.append("carrot")
vegetables.append("potato")
vegetables.reverse()
vegetables.pop()
vegetables.pop()
print(vegetables)
```

A. []

B. ['carrot']

C. ['potato']

D. ['tomato']

19. 次のコードを実行した結果として表示されるものを選択してください（1つ選択）

```
print([[3 * i + j + 1 for j in range(3)] for i in range(2)])
```

A. [[1, 2, 3, 4, 5, 6]]

B. [[1, 2, 3], [4, 5, 6]]

C. [[1, 2], [3, 4], [5, 6]]

D. [[1], [2], [3], [4], [5], [6]]

20. 次のコードを実行して`{2} {2, 3, 5, 7, 9, 11}`が表示されるとき、空欄①、②に入る記述の組み合わせとして正しいものを選択してください（1つ選択）

```
odds = {3, 5, 7, 9, 11}
primes = {2, 3, 5, 7, 11}
print(①, ②)
```

A. ① `primes ^ odds` ② `odds & primes`

B. ① `primes ^ odds` ② `odds | primes`

C. ① `primes - odds` ② `odds & primes`

D. ① `primes - odds` ② `odds | primes`

21. ディクショナリの作成方法として誤っているものを選択してください（1つ選択）

A. `{"carrot": 80, "tomato": 100}`

B. `dict("carrot"=80, "tomato"=100)`

C. `dict([("carrot", 80), ("tomato", 100)])`

D. `{k: v for k, v in [("carrot", 80), ("tomato", 100)]}`

22. 次のコードを実行して期待する結果が表示されるとき、空欄①に入る記述として正しいものを選択してください（1つ選択）

```
vegetables = {0: "carrot", 1: "potato", 2: "tomato"}

for kv in ①:
    print(kv)
```

【期待する結果】

```
(0, 'carrot')
(1, 'potato')
(2, 'tomato')
```

A. `vegetables`

B. `zip(vegetables)`

C. `vegetables.items()`

D. `enumerate(vegetables)`

23. 次のコードを実行して`True`が表示されるとき、空欄①に入る記述として正しいものを選択してください（1つ選択）

```
temp = 22
print(①)
```

A. `10 <= temp < 20`

B. `temp >= 10 or temp < 20`

C. `temp >= 10 and temp < 20`

D. `temp >= 10 and not temp >= 20 or temp < 20`

24. 次のコードを実行した結果として表示されるものを選択してください（1つ選択）

```
compare1 = "carrot" < "tomato" < "potato"
compare2 = [3, 2] < [3, 1, 4]
compare3 = (3, 1) < (3, 1, -1)

print(compare1, compare2, compare3)
```

A. False False False

B. False False True

C. False True False

D. True False False

25. 次のコードを正常に実行できるとき、空欄①に入る記述として誤っているものを選択してください（1つ選択）

```
import math
from math import pi
from math import pi as pai

print(①)
```

A. `pi`

B. `pai`

C. `math.pi`

D. `math.pai`

26. 下記のディレクトリ構成のtodo_appパッケージを使います。baseサブモジュールをインポートする方法として正しいものを選択してください（1つ選択）

【ディレクトリ構成】

```
todo_app/
    __init__.py
    model/
        __init__.py
        base.py
```

A. `import base`

B. `import model.base`

C. `import todo_app.base`

D. `import todo_app.model.base`

27. open関数のモード引数（第2引数）の特徴として誤っているものを選択してください（1つ選択）

A. モード引数は省略できない

B. `"rb"`は、バイナリモードの読み込みの指定である

C. `"r+"`は、テキストモードの読み書きの指定である

D. `"a"`は、テキストモードの追加書き込みの指定である

28. 次のコードを実行した結果として表示されるエラーを選択してください（1つ選択）

```
print(python_version)
```

A. NameError

B. TypeError

C. ValueError

D. SyntaxError

29. 次のコードを実行した結果として表示されるものを選択してください（1つ選択）

```
numerator = 1
denominator = 0
try:
    result = numerator / denominator
except ZeroDivisionError:
    print("division by zero")
else:
    print("executed else")
finally:
    print("executed finally")
```

A.

executed else
executed finally

B.

division by zero
executed else

C.

division by zero
executed finally

D.

division by zero

30. 次のコードを正常に実行できるとき、空欄①に入る記述として正しいものを選択してください（1つ選択）

```
def divide(numerator, denominator):
    try:
        result = numerator / denominator
    except ZeroDivisionError as e:
        raise ValueError("Illegal argument") ① e
    else:
        return result

try:
    result = divide(1, 0)
except ValueError as e:
    print(e)
```

A. `as`

B. `from`

C. `in`

D. `with`

31. 次のコードで例外時でもデータベースの終了処理を実行したいとき、空欄①に入る記述として正しいものを選択してください（1つ選択）

```
# データベースの接続処理
...  # 略
print("Open DB")
try:
    # データベースの処理
```

```
    ...  # 略
①:
    # データベースの終了処理
    ...  # 略
    print("Close DB")
```

A. close

B. ending

C. finally

D. terminal

32. 次のコードを実行した結果として表示されるものを選択してください（1つ選択）

```
class Path1:
    def __init__(self, dirs):
        self.dirs = dirs
        self.sep = "/"
    def join(self):
        return self.sep.join(self.dirs)

class Path2(Path1):
    def __init__(self, dirs):
        super().__init__(dirs)
        self.sep = "\\"

pth1 = Path1(["home", "taro"])
pth2 = Path2(["users", "taro"])

print(pth1.join(), pth2.join())
```

A. home/taro users/taro

B. home/taro users\taro

C. home\taro users/taro

D. home\taro users\taro

33. 次のコードを実行して0 1 1 2 3 5 8が表示されるとき、空欄①に入る記述として正しいものを選択してください（1つ選択）

```
def fibonacci(limit):
    i, j = 0, 1
    while i <= limit:
        ① i
        i, j = j, i + j
```

```
for i in fibonacci(10):
    print(i, end=" ")
```

A. `push`

B. `return`

C. `send`

D. `yield`

34. osモジュールの特徴として誤っているものを選択してください（1つ選択）

A. `os.chdir(移動先)`でカレントディレクトリを変更できる

B. `os.pwd()`でカレントディレクトリを取得できる

C. `dir(os)`でosモジュールの全属性名を取得できる

D. `help(os)`でosモジュールのヘルプを確認できる

35. 次のファイル`file.py`を実行して`['buy', 'milk'] 2/3`が表示されるとき、実行方法の記述として正しいものを選択してください（1つ選択）

```
import argparse

parser = argparse.ArgumentParser()
parser.add_argument("task", nargs="+")
parser.add_argument("--date")
args = parser.parse_args()
print(args.task, args.date)
```

A. `python file.py buy milk 2/3`

B. `python file.py 2/3 buy milk`

C. `python file.py --date 2/3 buy milk`

D. `python file.py --date 2/3 "['buy', 'milk']"`

36. 次のファイルを実行したとき処理される内容として正しいものを選択してください（1つ選択）

```
from datetime import date

def date_str(year, month, day):
    """To string from year, month, date.
```

```
    >>> print(date_str(2024, 2, 3))
    2024-02-03
    """
    dt = date(year, month, day)
    return dt.strftime("%Y-%m-%d")

import doctest
doctest.testmod()
```

A. `print(date_str(2024, 2, 3))`を実行せず、何もチェックしない

B. `print(date_str(2024, 2, 3))`を実行せず、docstringがあるかをチェックする

C. `print(date_str(2024, 2, 3))`を実行し、エラーが発生するかどうかだけをチェックする

D. `print(date_str(2024, 2, 3))`を実行し、出力が`2024-02-03`と同じかどうかをチェックする

37. 標準ライブラリの説明として誤っているものを選択してください（1つ選択）

A. csvモジュールは、スプレッドシートなどで使用されているCSV形式のファイルを読み書きできる

B. emailパッケージは、電子メールのデコードやヘッダプロトコルの処理などメッセージの構築ができる

C. jsonパッケージは、JSON形式のデータを安定的にパースできる

D. sqlite1モジュールは、SQLを用いてSQLiteデータベースを更新できる

38. 次のコードを実行した結果として表示されるものを選択してください（1つ選択）

```
import logging

logging.warning("Key not found")
logging.critical("Key not found")
logging.debug("Key not found")
logging.error("Key not found")
logging.info("Key not found")
```

A.

```
CRITICAL:root:Key not found
ERROR:root:Key not found
```

B.

```
CRITICAL:root:Key not found
DEBUG:root:Key not found
ERROR:root:Key not found
```

C.

```
WARNING:root:Key not found
CRITICAL:root:Key not found
ERROR:root:Key not found
```

D.

```
WARNING:root:Key not found
CRITICAL:root:Key not found
DEBUG:root:Key not found
ERROR:root:Key not found
```

39. コマンドとその説明として誤っているものを選択してください（1つ選択）

A. `pip install --file requirements.txt`は、requirements.txtに記述されたパッケージをインストールする

B. `python -m pip install --upgrade pip`は、pipの最新版をインストールする

C. `pip uninstall pyyaml`は、pyyamlをアンインストール（削除）する

D. `pip freeze > requirements.txt`は、インストールされたパッケージ一覧を作成し、requirements.txtに保存する

40. 対話型インタープリターの説明として誤っているものを選択してください（1つ選択）

A. ユーザディレクトリの.python_historyに入力のヒストリが保存される

B. 拡張されたインタープリターとして、bpythonやIPythonがある

C. if文などのブロック内で改行すると、インデントが自動的に挿入される

D. ローカル変数などを途中まで入力してTabキーを押すと、候補が表示されたり補完されたりする