

平成28年度 成果報告書



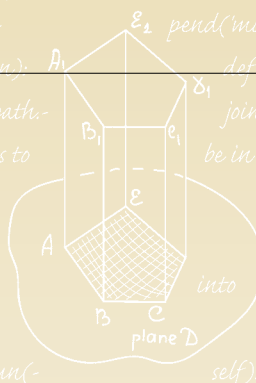
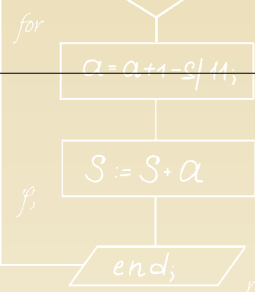
【文部科学省】
情報技術人材育成のための実践教育ネットワーク形成事業

分野・地域を越えた実践的情報教育協働ネットワーク



成長分野を支える情報技術人材の育成拠点の形成





**発行: 大阪大学大学院情報科学研究科
enPiT事務局**

〒565-0871 大阪府吹田市山田丘1-5
TEL ▶ 06-6879-4395
FAX ▶ 06-6879-4649
URL ▶ <http://www.enpit.jp/>
E-mail ▶ enpit-info@ist.osaka-u.ac.jp



enPiT1運営委員会 (分野・地域を越えた実践的情報教育協働ネットワーク)

大阪大学、東北大学、筑波大学、東京大学、東京工業大学、名古屋大学、神戸大学、九州大学、九州工業大学、北陸先端科学技術大学院大学、奈良先端科学技術大学院大学、公立はこだて未来大学、産業技術大学院大学、慶應義塾大学、情報セキュリティ大学院大学

enPiT2運営委員会 (成長分野を支える情報技術人材の育成拠点の形成)

大阪大学、東京大学、東京工業大学、お茶の水女子大学、電気通信大学、神戸大学、和歌山大学、九州工業大学、東北大学、北海道大学、北陸先端科学技術大学院大学、奈良先端科学技術大学院大学、和歌山大学、岡山大学、九州大学、慶應義塾大学、東京電機大学、情報セキュリティ大学院大学、名古屋大学、岩手大学、徳島大学、東海大学、南山大学、筑波大学、室蘭工業大学、埼玉大学、愛媛大学、琉球大学、公立はこだて未来大学、岩手県立大学、会津大学、産業技術大学院大学、山口大学

we need the files to be in a specific directory hierarchy for -I <include_dir>

```

python2 REQUIRED_PACKAGES.append('numpy')
class BinaryDistribution(Distribution):
    def install_headers(self):
        self.install_headers = os.path.join(self.install_purelib, 'include')
        return self.install_headers
    def copy_file(self, header):
        install_dir = os.path.join(self.install_purelib, 'include')
        self.copy_file(header, install_dir)
    def find_files(self, pattern, root):
        """Return all the files matching pattern in root"""
        return [os.path.join(root, f) for f in os.listdir(root) if f.startswith(pattern)]
    def get_outputs(self):
        return self.outfiles
    def run(self):
        self.run_command(self.install_headers)
        self.run_command(self.copy_file)
        self.run_command(self.find_files)
        self.run_command(self.get_outputs)
    def __init__(self, *args, **kwargs):
        super(BinaryDistribution, self).__init__(*args, **kwargs)
        self.install_headers = os.path.join(self.install_purelib, 'include')
        self.copy_file = self.copy_file
        self.find_files = self.find_files
        self.get_outputs = self.get_outputs
        self.run = self.run
        self.run_command = self.run_command
    def __str__(self):
        return "BinaryDistribution(%s)" % self.install_headers

```

```

def run(self):
    self.run_command(self.install_headers)
    self.run_command(self.copy_file)
    self.run_command(self.find_files)
    self.run_command(self.get_outputs)
    self.run_command(self.run)
    self.run_command(self.run_command)

```

```

class BinaryDistribution(Distribution):
    def install_headers(self):
        self.install_headers = os.path.join(self.install_purelib, 'include')
        return self.install_headers
    def copy_file(self, header):
        install_dir = os.path.join(self.install_purelib, 'include')
        self.copy_file(header, install_dir)
    def find_files(self, pattern, root):
        """Return all the files matching pattern in root"""
        return [os.path.join(root, f) for f in os.listdir(root) if f.startswith(pattern)]
    def get_outputs(self):
        return self.outfiles
    def run(self):
        self.run_command(self.install_headers)
        self.run_command(self.copy_file)
        self.run_command(self.find_files)
        self.run_command(self.get_outputs)
    def __init__(self, *args, **kwargs):
        super(BinaryDistribution, self).__init__(*args, **kwargs)
        self.install_headers = os.path.join(self.install_purelib, 'include')
        self.copy_file = self.copy_file
        self.find_files = self.find_files
        self.get_outputs = self.get_outputs
        self.run = self.run
        self.run_command = self.run_command
    def __str__(self):
        return "BinaryDistribution(%s)" % self.install_headers

```

```

def run(self):
    self.run_command(self.install_headers)
    self.run_command(self.copy_file)
    self.run_command(self.find_files)
    self.run_command(self.get_outputs)
    self.run_command(self.run)
    self.run_command(self.run_command)

```

```

class BinaryDistribution(Distribution):
    def install_headers(self):
        self.install_headers = os.path.join(self.install_purelib, 'include')
        return self.install_headers
    def copy_file(self, header):
        install_dir = os.path.join(self.install_purelib, 'include')
        self.copy_file(header, install_dir)
    def find_files(self, pattern, root):
        """Return all the files matching pattern in root"""
        return [os.path.join(root, f) for f in os.listdir(root) if f.startswith(pattern)]
    def get_outputs(self):
        return self.outfiles
    def run(self):
        self.run_command(self.install_headers)
        self.run_command(self.copy_file)
        self.run_command(self.find_files)
        self.run_command(self.get_outputs)
    def __init__(self, *args, **kwargs):
        super(BinaryDistribution, self).__init__(*args, **kwargs)
        self.install_headers = os.path.join(self.install_purelib, 'include')
        self.copy_file = self.copy_file
        self.find_files = self.find_files
        self.get_outputs = self.get_outputs
        self.run = self.run
        self.run_command = self.run_command
    def __str__(self):
        return "BinaryDistribution(%s)" % self.install_headers

```

```

class BinaryDistribution(Distribution):
    def install_headers(self):
        self.install_headers = os.path.join(self.install_purelib, 'include')
        return self.install_headers
    def copy_file(self, header):
        install_dir = os.path.join(self.install_purelib, 'include')
        self.copy_file(header, install_dir)
    def find_files(self, pattern, root):
        """Return all the files matching pattern in root"""
        return [os.path.join(root, f) for f in os.listdir(root) if f.startswith(pattern)]
    def get_outputs(self):
        return self.outfiles
    def run(self):
        self.run_command(self.install_headers)
        self.run_command(self.copy_file)
        self.run_command(self.find_files)
        self.run_command(self.get_outputs)
    def __init__(self, *args, **kwargs):
        super(BinaryDistribution, self).__init__(*args, **kwargs)
        self.install_headers = os.path.join(self.install_purelib, 'include')
        self.copy_file = self.copy_file
        self.find_files = self.find_files
        self.get_outputs = self.get_outputs
        self.run = self.run
        self.run_command = self.run_command
    def __str__(self):
        return "BinaryDistribution(%s)" % self.install_headers

```

$$\ln x := x - 1$$

$$\ln y := y + 1$$

$$x := (x + y) / 2$$

$$\ln x := x - 1$$

